A spectral algorithm for fast *de novo* layout of uncorrected long nanopore reads

Antoine Recanati [1], Thomas Brüls [2,3,4] and Alexandre d'Aspremont [1]

[1]CNRS & D.I., UMR 8548, École Normale Supérieure, Paris, France.
[2]Commissariat à l'Energie Atomique et aux Energies Alternatives, Direction de la Recherche Fondamentale, Institut de Génomique, Genoscope, Evry, Essonne, 91057, France.
[3]UMR 8030 - Génomique Métabolique, Centre National de la Recherche Scientifique, Evry, Essonne, 91057, France.
[4]Université d'Evry-Val-d'Essonne & Université Paris-Saclay, Evry, Essonne, 91000, France.

ABSTRACT

Motivation: New long read sequencers promise to transform sequencing and genome assembly by producing reads tens of kilobases long. However their high error rate significantly complicates assembly and requires expensive correction steps to layout the reads using standard assembly engines.

Results: We present an original and efficient spectral algorithm to layout the uncorrected nanopore reads, and its seamless integration into a straightforward overlap/layout/consensus (OLC) assembly scheme. The method is shown to assemble Oxford Nanopore reads from several bacterial genomes into good quality ($\sim 99\%$ identity to the reference) genome-sized contigs, while yielding more fragmented assemblies from a *Sacharomyces cerevisiae* reference strain.

Availability and implementation:
https://github.com/antrec/spectrassembler

Contact: antoine.recanati@inria.fr

1 INTRODUCTION

De novo whole genome sequencing aims at reconstructing an entire genome from randomly sampled sub-fragments whose order and orientation within the genome is unknown. The genome is oversampled so that all parts are covered multiple times with high probability.

High-throughput sequencing technologies such as Illumina substantially reduced sequencing cost at the expense of read length, which is typically a few hundred base pairs long at best. Yet, *de novo* assembly is challenged by short reads, as genomes contain repeated sequences resulting in layout degeneracies when read length is shorter or on the same order than repeat length (Pop [2004]). In many bacterial genomes for example, the longest repeat sequence is the conserved ribosomal RNA operon which can span about 8 kilobases in length (Koren et al. [2013]), while the rDNA region in the *Saccharomyces cerevisiae* genome encompasses 1 to 2 million bases (Mb).

Recent long read sequencing technologies such as PacBio's SMRT and Oxford Nanopore Technology (ONT) have spurred a renaissance in *de novo* assembly as they produce reads over 10kbp long (Koren and Phillippy [2015]). However, their high error rate ($\sim 15\%$) makes the task of assembly difficult, requiring complex and computationally intensive pipelines.

Most approaches for long read assembly address this problem by correcting the reads prior to using standard assembly engines such as Canu (Koren et al. [2016]) or Celera Assembler, (Myers et al. [2000])). *Hybrid techniques* combine short and long read technologies: the accurate short reads are mapped onto the long reads, enabling a consensus sequence to be derived for each long read and thus providing low-error long reads (see for example (Madoui et al. [2015])). This method was shown to successfully assemble prokaryotic and eukaryotic genomes with PacBio (Koren et al. [2012]) and ONT (Goodwin et al. [2015]) data. *Hierarchical assembly* follows the same mapping and consensus principle but resorts to long read data only, the rationale being that the consensus sequence derived from all erroneous long reads matching a given position of the genome should be accurate provided there is sufficient coverage and sequencing errors are reasonably randomly distributed: for a given base position on the genome, if 8 out of 50 reads are wrong, the majority vote still yields the correct base. Hierarchical methods map long reads against each other and derive, for each read, a consensus sequence based on all the reads that overlap it. Such an approach was implemented in HGAP (Chin et al. [2013]) to assemble PacBio SMRT data, and more recently by Loman et al. [2015], to achieve complete *de novo* assembly of *Escherichia coli* with ONT data exclusively.

Recently, Li [2016] showed that it is possible to efficiently perform *de novo* assembly of noisy long reads in only two steps and without any dedicated correction procedure: all-vs-all raw read mapping (with `minimap`) and assembly (with `miniasm`). The `miniasm` assembler is inspired by the Celera Assembler and produces unitigs through the construction of an assembly graph. Its main limitation is that it produces a draft whose error rate is of the same order as the raw reads.

We present here a new method for computing the layout of raw nanopore reads, resulting in a simple and computationally efficient protocol for assembly. It takes as input the all-vs-all overlap information (*e.g.* from `minimap`, `MHAP` (Berlin et al. [2015]) or `DALIGNER` (Myers [2014])) and outputs a layout of the reads (*i.e.* their position and orientation in the genome). Like `miniasm`, it computes an assembly from the all-vs-all raw read mapping, but can achieve improved quality through a coveraged-based consensus generation process, as in `nanocorrect` (Loman et al. [2015]), although reads are not corrected individually. The method relies on a simple algorithm with deep theoretical underpinnings, that we describe in §2.1. To our knowledge, it has never been successfully applied to *de novo* assembly. In §2.2, we describe an assembler based on this layout method, to which we add a consensus generation step based on `POA` (Lee et al. [2002]), a multi-sequence alignment engine. Finally, we evaluate this pipeline on prokaryotic and eukaryotic genomes in §3, and discuss possible improvements and limitations in §4.

2 METHODS

2.1 Layout computation

We layout the reads in two steps. We first sort them by position, i.e., find a permutation π such that read $\pi(1)$ will positioned before read $\pi(2)$ on the genome. Then, we iteratively assign an exact position (i.e., leftmost basepair coordinate on the genome) to each read by using the previous read's position and the overlap information.

The key step is the first one, which we cast as a seriation problem, i.e. we seek to reconstruct a linear order between n elements using unsorted, pairwise similarity information [Atkins et al., 1998; Fogel et al., 2013]. Here the n elements are the reads, and the similarity information comes from the overlapper (*e.g.* from `minimap`, `MHAP` (Berlin et al. [2015]).

The seriation problem is formulated as follows. Given a pairwise similarity matrix A_{ij}, and assuming the data has a serial structure, *i.e.* that there exists an order π such that $A_{\pi(i)\pi(j)}$ decreases with $|i-j|$, seriation seeks to recover this ordering π (see Figure 1 for an illustration). If such an order π exists, it minimizes the 2-SUM score,

$$\text{2-SUM}(\pi) = \sum_{i,j=1}^{n} A_{ij}\left(\pi(i) - \pi(j)\right)^2, \qquad (1)$$

and the seriation problem can be solved as a minimization over the set of permutation vectors (Fogel et al. [2013]). In other words, the permutation π should be such that if A_{ij} is high (meaning that i and j have a high similarity), then $(\pi(i) - \pi(j))^2$ should be low, meaning that the positions $\pi(i)$ and $\pi(j)$ should be close to each other. Conversely, if $A_{ij} = 0$, the positions of i and j in the new order may be far away without affecting the score.

When using seriation to solve genome assembly problems, the similarity A_{ij} measures the overlap between reads i and j. In an ideal setting with constant read length and no repeated regions, two overlapping reads should have nearby positions on the genome. We therefore expect the order found by seriation to roughly match the sorting of the positions of the reads.

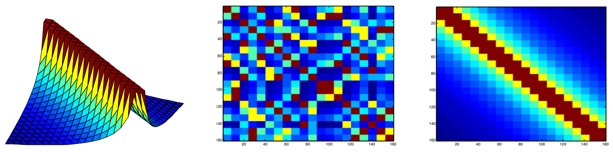


Fig. 1: A similarity matrix reordered with the spectral algorithm. The original matrix (left) has values that decrease when moving away from the diagonal. It is randomly permuted (middle), and the spectral algorithm finds the original ordering (right).

The problem of finding a permutation over n elements is combinatorial, but provided the original data has a serial structure, an exact solution to seriation exists in the noiseless case [Atkins et al., 1998] using spectral clustering, and there exist several convex relaxations allowing explicit constraints on the solution (Fogel et al. [2013]).

The exact solution is directly related to the well-known spectral clustering algorithm. Indeed, for any vector $\mathbf{x}$, the objective in (1) reads

$$\sum_{i,j=1}^{n} A_{ij}\left(x_i - x_j\right)^2 = x^T L_A x, \quad L_A = \mathbf{diag}(A\mathbf{1}) - A$$

where L_A is the Laplacian matrix of A. This means that the 2-SUM problem amounts to

$$\min_{\pi} \pi^T L_A \pi$$

where π is a permutation vector. Roughly speaking, the spectral clustering approach to seriation relaxes the constraint "π is a permutation vector" into "π is a normalized vector of $\mathbb{R}^n$ orthogonal to the constant vector $\mathbf{1} = (1, ..., 1)^T$" with fixed norm. The problem then becomes

$$\min_{\{\mathbf{1}^T\pi = 0,\ \|\pi\|_2 = 1\}} \pi^T L_A \pi$$

This relaxed problem is an eigenvector problem. Finding the minimum over normalized vectors x yields the eigenvector associated to the smallest eigenvalue of L_A, but the smallest eigenvalue, 0, is associated with the eigenvector $\mathbf{1}$, from which we cannot recover any permutation. However, if we restrict x to be orthogonal to $\mathbf{1}$, the solution is the second smallest eigenvector, called the Fiedler vector. A permutation is recovered from this eigenvector by sorting its coefficients: given $\mathbf{x} = (x_1, x_2, ..., x_n)$, the algorithm outputs a permutation π such that $x_{\pi(1)} \leq x_{\pi(2)} \leq ... \leq x_{\pi(n)}$. This procedure is summarized as Algorithm 1.

In fact, [Atkins et al., 1998] showed that under the assumption that A has a serial structure, Algorithm 1 solves the seriation problem exactly, i.e. recovers the order π such that $A_{\pi(i)\pi(j)}$ decreases with $|i-j|$. This means that we solve the read ordering problem by simply solving an extremal eigenvalue problem, which has low complexity (comparable to PCA).

In practice, overlap-based similarity matrices are sparse : a read has overlaps with only a few others (the number of neighbors of a read in the overlap graph is roughly the coverage). There is no sparsity requirement for the algorithm to work, however sparsity lowers RAM usage since we store the $n \times n$ similarity matrix with only $\sim n \times C$ non-zero values, with C the coverage. In such cases, the ordered similarity matrix is band diagonal, as in Figure 3b.

Algorithm 1 Spectral ordering

```
Input: Connected similarity matrix $A \in \mathbb{R}^{n \times n}$
1: Compute Laplacian $L_A = \mathbf{diag}(A\mathbf{1}) - A$
2: Compute second smallest eigenvector of $L_A$, $\mathbf{f}^*$
3: Sort the values of $\mathbf{f}^*$
Output: Permutation $\pi : \mathbf{f}^*_{\pi(1)} \leq \mathbf{f}^*_{\pi(2)} \leq ... \leq \mathbf{f}^*_{\pi(n)}$
```

Once the reads are reordered, we can sequentially compute their exact positions (basepair coordinate of their left end on the genome) and orientation. We assign position 0 and strand "+" to the first read, and use the overlap information (position of the overlap on each read and mutual orientation) to compute the second read's position and orientation, etc. More specifically, when computing the position and orientation of read i, we use the information from reads $i-1, ..., i-c$ to average the result, where c equals the coverage, as this makes the layout more robust to misplaced reads. Note that overlappers relying on hashing, such as `minimap` and `MHAP`, do not generate alignments but still locate the overlaps on the reads, making this positioning step possible.

2.2 Consensus generation

We built a simple assembler using this layout idea and tested its accuracy. It is partly inspired from the `nanocorrect` pipeline from Loman et al. [2015] in which reads are corrected using multiple alignments of all overlapping reads. These multiple alignments are performed with Partial Order Aligner `POA` (Lee et al. [2002]), which computes a consensus sequence from the alignment of multiple sequences using a dynamic programming approach that is efficient when the sequences are quite similar (which is the case if we trim the sequences to align their overlapping parts).

The key point is that we do not need to perform multiple alignment using all reads, since we already have a layout. Instead, we can generate

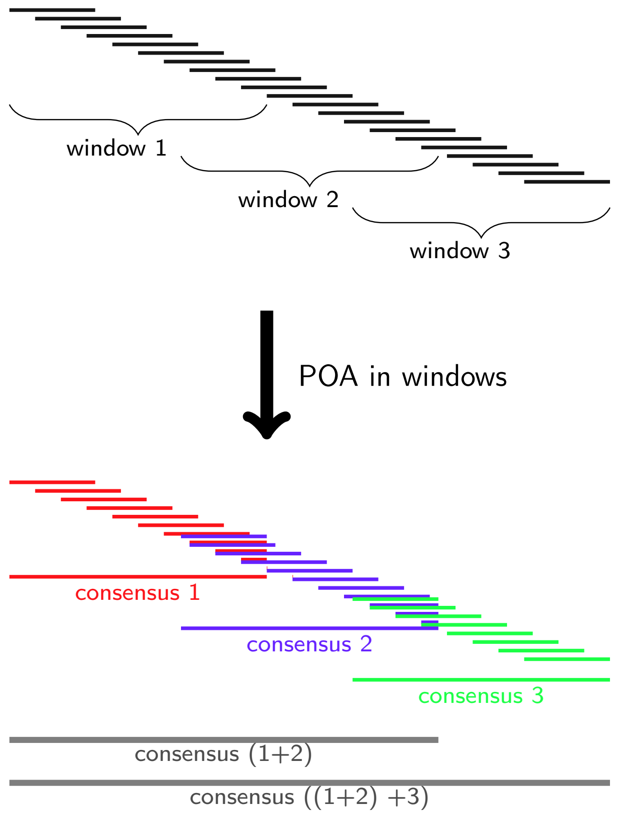


Fig. 2: Consensus generation. Given the layout, the genome is sliced into overlapping windows, and a consensus is computed in each window. The final consensus is then obtained by merging the consensus windows.

a consensus sequence for, say, the first 2000 bp of the genome by aligning the parts of the reads that are included in this window with `POA`, and repeat this step for the reads included in the window comprising the next 2000bp of the genome, etc. In practice, we take consecutive windows that overlap and then merge them to avoid errors at the edges, as shown on Figure 2. The top of the figure displays the layout of the reads broken down into three consecutive overlapping windows, with one consensus sequence generated per window with `POA`. The final assembly is obtained by iteratively merging the window k+1 to the consensus formed by the windows $1, \dots, k$.

The computational complexity for aligning N sequences of length L, with an average divergence between sequences ϵ, with POA is roughly $O(mNL^2)$, with $m \simeq (1 + 2\epsilon)$. With 10% of errors, m is close to 1. If each window of size L_w contains about C sequences, the complexity of building the consensus in a window is $O(mCL_w^2)$. We compute L_g/L_w consensus windows, with L_g the length of the genome (or contig), so the overall complexity of the consensus generation is $O(mCL_gL_w)$. We therefore chose a relatively small window length, but large enough to span a sufficient length so as to prevent mis-assemblies due to noise in the layout. In practice we used $L_w = 2500$bp.

2.3 Outliers removal

In practice, the (reordered) similarity matrix contains outliers outside the main diagonal band (see Figure 3) that corrupt the ordering. These outliers are typically provoked by either repeated subsequences or to sequencing noise. Most of the repeat-induced overlaps can be spotted because reads

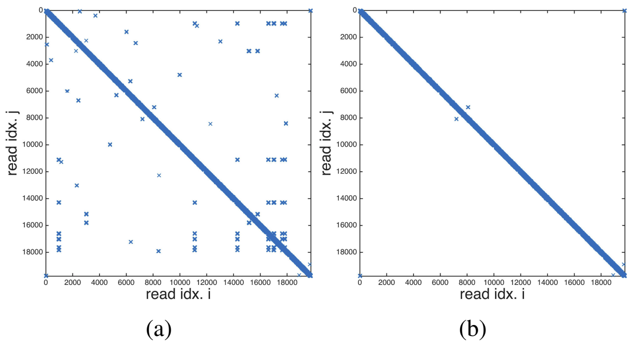


Fig. 3: Similarity matrix for E. coli before (a) and after (b) thresholding and removal of connecting reads. The read indices are sorted by increasing position in the genome.

harboring repeated sequences tend to overlap with a much larger number of reads. By weighting the similarity (for example, the `--weighted` option in `MHAP` adds a tf-idf scaling to the MinHash sketch, making repetitive k-mers less likely to cause a match between two sequences) , the value associated to such overlaps can be significantly reduced, so that a simple threshold on the similarity score can get rid of them. We also applied a second overlap-length threshold, keeping only the overlaps that exceed a minimal length (3500bp in our experiments).

After this filtering step, there remains however a small number of reads that have non-zero similarity with other reads at some distant part of the genome. These reads usually overlap with a first subset of reads at a given position in the genome, and with another distinct subset of reads at another location in the genome, with no overlap between these distinct subsets. We call such reads "connecting reads", and they can be detected from the similarity matrix by computing, for each read (index i), the set of its neighbors in the graph $\mathcal{N}_i = \{j : A_{ij} > 0\}$. The subgraph represented by A restricted to $\mathcal{N}_i$ is either connected (there exists a path between any pair of edges), or split into separate connected components. In the latter case, we label read i as a "connecting read" and discard it.

These filtering steps generally yield a clean similarity matrix, but break the read graph into multiple connected components. The spectral algorithm can find an ordering in each connected component but not for the unconnected full graph A. The consensus phase will therefore be performed separately for each connected component, generating one contig per connected component. There is obviously a tradeoff involved in thresholding the similarity matrix A: aggressive filtering will yield many well ordered contigs, but increases the risk of missing small portions of the genome between the contigs (depending on the coverage). On the other hand, too permissive filtering might result in a single but incorrectly ordered contig. Note that the algorithm will always output an ordering for each connected component, and that there is no simple relation between the connectivity of the graph (number of connected components) and the quality of the ordering within each connected component. If the genome is repeat-free and the sequences noiseless (and with sufficient coverage), the algorithm will find the correct ordering with a single contig (connected component). We performed a Platonic genome experiment with noiseless reads of uniform length 7 kbp, uniformly sampled *in silico* from a 5Mb repeat-free virtual genome at 40X coverage. The resulting similarity matrix contained no outliers, and the ordering recovered by the spectral algorithm matched exactly the true ordering of the reads. Conversely, in the presence of many repeats, sequencing errors and shallower coverage, the preprocessing step necessary to promote well-ordered connected components is likely to break the graph into many connected components. Nevertheless, we can

Table 1. Test data sets

Name	Size	Cov.	N50
Acinetobacter baylyi ADP1	3.6M	28.1	6197
Escherichia coli K-12 MG1655	4.6M	30.2	8080
Saccharomyces cerevisiae S288C	12.4M	67.7	7865

Organism name, genome size, theoretical sequencing coverage and N50 read length. *Acinetobacter baylyi* and *Saccharomyces cerevisiae* data were sequenced at Genoscope under the MinION Access Program (MAP) with the R7.3 and R9 chemistry respectively, while the *Escherichia coli* data was derived from Loman et al. [2015] and kindly made available by the Loman lab (http://bit.ly/loman006 - PCR1 2D pass dataset).

usually find *a posteriori* if a contig is well ordered, as there is a significant jump in bandwidth (maximal distance to the diagonal of non-zeros entries of the matrix) for reordered similarity matrices containing outliers (with no outliers, the reordered matrix is a clean band diagonal). Empirically, we found that well-ordered contigs have a bandwidth smaller than twice the coverage. The optimal threshold can thus be identified by increasing its value until we have contigs with a bandwidth in the reordered matrix not exceeding about twice the coverage. This assembly pipeline is summarized in Algorithm 2.

Algorithm 2 OLC assembly pipeline

```
Input: $n$ long noisy reads
 1: Compute overlaps with an overlapper (e.g. minimap or MHAP)
 2: Construct similarity matrix $S \in \mathbb{R}^{n \times n}$ from the overlaps
 3: Remove outliers from $S$ with a threshold on values $S_{ij}$, on overlap
    length, and removal of connecting reads (as explained in §2.3)
 4: for all Connected component $A$ of $S$ do
 5:     Reorder $A$ with spectral algorithm (Algorithm 1)
 6:     if bandwidth of $A_{reordered} \geq 2 \times coverage$ then
 7:         set higher threshold on $A$ and try again
 8:     end if
 9:     Compute layout from the ordering found and overlaps
10:     Partition the length of the contig into small windows
11:     Compute consensus in each window with POA
12:     Merge consecutive windows with POA
13: end for
Output: Contig consensus sequence
```

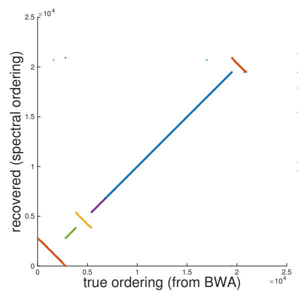
3 RESULTS

3.1 Data

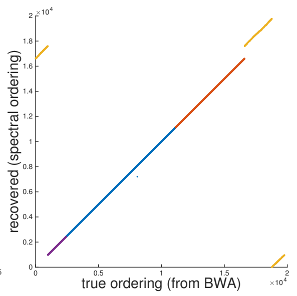
We tested this pipeline on *de novo* assembly of two bacterial (circular) and one eukaryotic yeast genomes. The three datasets (summarized in Table 1) were sequenced with Oxford Nanopore's MinION device (using the R7.3 chemistry for the *A. baylyi ADP1* genome and the R9 chemistry for the *S. cerevisiae S288C* genome), keeping only the reads passing the 2D high quality filtering. Histograms of the reads length are given in Figure 8 in Supplementary Material.

3.2 Layout

3.2.1 Bacterial genomes `minimap` was used to compute overlaps between the raw reads (we also tested the pipeline with `MHAP` and `DALIGNER` and obtained similar results). After thresholding the similarity matrix, keeping only overlaps larger than 3500bp and removing the "connecting reads", the similarity graph was split in 4 and 5 connected components for *E. coli* and *A. baylyi* respectively. The reads were successfully ordered in each of these, as shown in Figure 4.



(a) A. baylyi



(b) E. coli

Fig. 4: Ordering of the reads computed with the spectral algorithm vs true ordering (obtained by mapping the reads to the reference genome with `BWA`). All the contigs are displayed on the same plot for compactness.

3.2.2 Eukaryotic genome For the *S. cerevisiae* genome, the threshold on the similarity matrix had to be set higher than for the bacterial genomes because of a substantially higher number of repetitive regions and false overlaps, leading to 261 connected components. Almost all of them were correctly reordered with the spectral algorithm (see Figure 5).

3.3 Consensus

3.3.1 Bacterial genomes Once the layout was established, the previously described method was used to assemble the contigs and generate a consensus sequence. For the two bacterial genomes, the first round of the layout produced a small number (4 or 5) of connected components, each of them yielding a contig. Sufficient overlap was left between the contig sequences to find their layout in a second iteration of the algorithm and produce a single contig spanning the entire genome. The fact that the first-pass contigs overlap even though they result from breaking the similarity graph into several connected components might seem counter-intuitive at first sight. However, it should be noted that when an edge A_{ij} is cut between two overlapping reads i and j, possibly resulting in the creation of two contigs (one containing i and the other j), the sequence fragment at the origin of the overlap between the two reads is still there to yield an overlap when the two contigs are compared at the time of the second iteration.

The QUAST (Gurevich et al. [2013]) report of these two assemblies can be found in Supplementary Material (Tables 2 and 3). Briefly, the assemblies display about 99% sequence

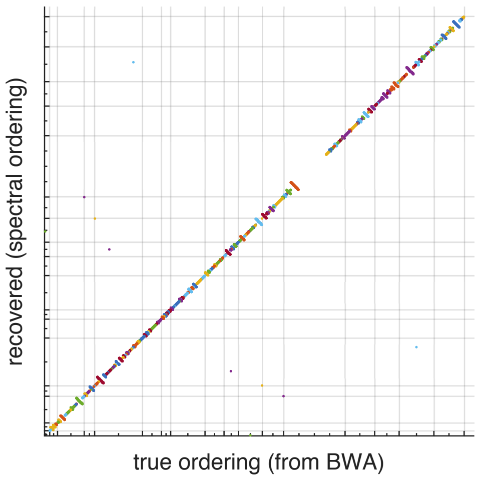


Fig. 5: Ordering of the *Saccharomyces cerevisiae* reads identified with the spectral algorithm vs true ordering (obtained by mapping the reads to the reference genome with `BWA` and concatenating the ordering found in each chromosome). The different chromosomes are separated by the grid; 4 misplaced reads are apparent.

identity to their reference genome, with errors mostly consisting in deletions, likely provoked by (uncorrected) errors in the raw reads. Misassemblies consisted in 4 and 5 relocations for the *A. baylyi* and *E. coli* genomes respectively, provoked by either deletions and/or misplaced reads in the layout.

Figure 6 illustrates the identity of both the final consensus sequences and the raw reads against the reference sequence, making apparent the effect of the sequence correction process.

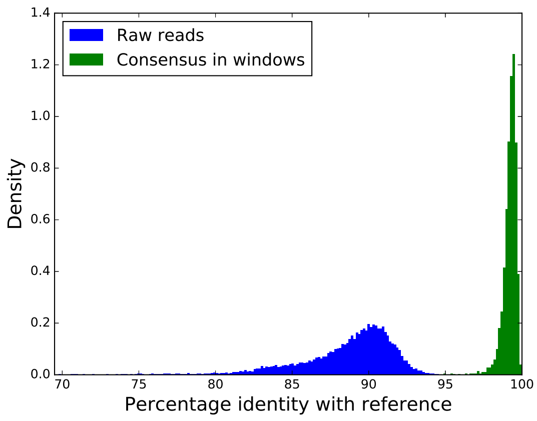


Fig. 6: Sequence identity of consensus windows vs raw reads against the *A. baylyi* reference genome.

3.3.2 Eukaryotic genome The assembly statistics and quality metrics of the 261 contigs are summarized in the QUAST report in Supplementary Table 4. The number of sequence errors is comparable to that obtained for the bacterial genomes, with 98.5% identity against the reference sequence. There are however more misassemblies, including relocations and translocations, as well as some parts of the genome that are missing (94.9% of the genome is recovered, vs 99.5% for the two bacterial genomes). The QUAST quality metrics obtained by separating the assembled contigs by chromosome are provided as Supplementary Table 5. A significant number of undetected overlaps between the first-pass connected components could be detected with `miniasm`, resulting in an assembly with fewer contigs that is shown in the dot plot of Figure 7. In addition to the QUAST evaluation, the long-range consistency of the *S. cerevisiae* assembly was further checked by generating *in silico* restriction maps of whole chromosomes, and aligning the contigs against them. This analysis confirmed that large contigs spanning multiple restriction sites were consistent on a long range, e.g., all the contigs containing ten or more distinct occurences of a given restriction site were correctly aligned to the reference genome (see Supplementary Material, Figure 11).

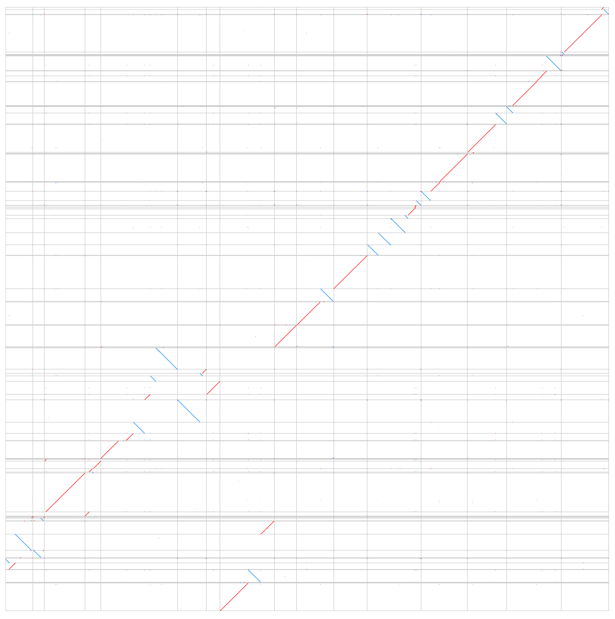


Fig. 7: Dot-plot of the *Sacharomyces cerevisiae* reference genome versus the 84 contigs reassembled with miniasm from the 261 original contigs (see text).

3.3.3 Optical mapping After the first iteration of the bacterial genome assembly pipeline, the overlaps between the first-pass contigs were sufficient to find their layout by a new iteration of the spectral ordering algorithm. It should be anticipated however that not all overlaps might be apparent in some cases, e.g., if too many reads were removed during the preprocessing step. One attractive option is to resort to the optical mapping technique (Aston et al. [1999]) to layout the contigs. We had such an optical map available for the *Acinetobacter baylyi* genome, and implemented the algorithm of Nagarajan et al. [2008] to map the contigs to the restriction map, which led to the same layout as the one identified from our two-round assemblies (data not shown), thus providing a "consistency check" for the layout. As illustrated previously and in the Supplementary Material (Figures 11 and 12) for the *S. cerevisiae* assembly, optical maps could be particularly valuable for the ordering of contigs from the more structurally complex eukaryotic genomes.

4 DISCUSSION

We have shown that seriation based layout algorithms can be successfully applied to *de novo* genome assembly problems, at least for genomes harboring a limited number of repeats.

In a similar vein to the recent report about the `miniasm` assembly engine [Li, 2016], our work confirms that the layout of long reads can be found without prior error correction, using only overlap information generated from raw reads by tools such as `minimap`, `MHAP` or `DALIGNER`. However, unlike `miniasm`, which does not derive a consensus but instead concatenates the reads into a full sequence, we take advantage of the multiple coverage provided by reads to produce contigs with a consensus quality on par with the sequence quality achieved by assembly pipelines executing dedicated error-correction steps. Compared to `miniasm`, our pipeline sacrifices in speed what it gains in sequence quality. Still, compared to approaches implementing error correction steps, we gain significant speed-ups by highly localizing the error correction and consensus generation processes, which is rendered possible by the knowledge of the layout. Because each read is basically corrected by aligning the reads that overlap with it, the speed-up is roughly proportional to the coverage. Our method has also a low memory footprint, as the space requirements for the layout computation mainly arise from the need to store sparse $n \times n$ matrices, with only about $n \times C$ non-zero coefficients (n being the number of reads and C the coverage). On the other hand, the size of the windows in which the consensus is computed bounds the memory used by POA in the multiple sequence alignment computation. In practice, we observed that less than 4Gb of RAM are required for window sizes smaller than 3000bp.

The main limitation of our layout algorithm is its sensitivity to outliers in the similarity matrix, hence the need to remove them in a pre-processing phase. Higher coverage and quality of the input reads, both expected in the near future, would likely improve the robustness of our pipeline. For eukaryotic genomes, we found that some outliers require additional information to be resolved (see Supplementary Material), which could be provided in the future by extracting topological information from the assembly graph.

In the meantime, our pipeline behaves like a draft generating assembler for prokaryotic genomes, and a first-pass unitigger for eukaryotic genomes. Importantly, the modularity of the overall approach provides flexibility to integrate other algorithms in order to increase the robustness of the layout or the quality of the consensus.

Our original contribution here consists in the layout computation. The spectral OLC assembler we built on top of it could be enhanced in many ways. We have shown that the spectral algorithm is suited to find the layout for bacterial genomes, even though there is room left for performance improvements on repeat-rich eukaryotic genomes.

For the latter genomes, it could make sense to use the spectral algorithm jointly with other assembly engines (*e.g.* `miniasm`, `canu`), to check the consistency of the connected components before they are assembled. Our consensus generation method is coarse-grained for now and does not take base-call quality and the statistical properties of ONT sequencing errors into account. Nevertheless, the three components (O, L and C) of the method being independent, an external and more refined consensus generation process could readily be plugged after the overlap and layout computations to further improve results and reach high-end accuracy.

ACKNOWLEDGEMENT

TB would like to thank Genoscope's sequencing (Laboratoire de Séquençage) and bioinformatics (Laboratoire d'Informatique Scientifique) teams for sharing some *Acinetobacter baylyi* ADP1 and *Sacharomyces cerevisiae* S288C MinION data, and is grateful to Oxford Nanopore Technologies Ltd for granting Genoscope access to its MinION device via the MinION Access Programme.

AA and AR would like to acknowledge support from the European Research Council (project SIPA). The authors would also like to acknowledge support from the chaire *Économie des nouvelles données*, the *data science* joint research initiative with the *fonds AXA pour la recherche* and a gift from Société Générale Cross Asset Quantitative Research.

REFERENCES

Aston, C., Mishra, B., and Schwartz, D. C. (1999). Optical mapping and its potential for large-scale sequencing projects. *Trends in Biotechnology*, 17(7):297–302.

Atkins, J. E., Boman, E. G., and Hendrickson, B. (1998). A spectral algorithm for seriation and the consecutive ones problem. *SIAM Journal on Computing*, 28(1):297–310.

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

Chin, C.-S., Alexander, D. H., Marks, P., Klammer, A. A., Drake, J., Heiner, C., Clum, A., Copeland, A., Huddleston, J., and Eichler, E. E. (2013). Nonhybrid, finished microbial genome assemblies from long-read smrt sequencing data. *Nature methods*, 10(6):563–569.

Fogel, F., Jenatton, R., Bach, F., and d'Aspremont, A. (2013). Convex relaxations for permutation problems. pages 1016–1024.

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

Gurevich, A., Saveliev, V., Vyahhi, N., and Tesler, G. (2013). Quast: quality assessment tool for genome assemblies. *Bioinformatics*, 29(8):1072–1075.

Koren, S., Harhay, G. P., Smith, T., Bono, J. L., Harhay, D. M., Mcvey, S. D., Radune, D., Bergman, N. H., and Phillippy, A. M. (2013). Reducing assembly complexity of microbial genomes with single-molecule sequencing. *Genome Biol*, 14(9):R101.

Koren, S. and Phillippy, A. M. (2015). One chromosome, one contig: complete microbial genomes from long-read sequencing and assembly. *Current Opinion in Microbiology*, 23:110–120.

Koren, S., Schatz, M. C., Walenz, B. P., Martin, J., Howard, J. T., Ganapathy, G., Wang, Z., Rasko, D. A., McCombie, W. R., and Jarvis, E. D. (2012). Hybrid error correction and de novo assembly of single-molecule sequencing reads. *Nature biotechnology*, 30(7):693–700.

Koren, S., Walenz, B. P., Berlin, K., Miller, J. R., and Phillippy, A. M. (2016). Canu: scalable and accurate long-read assembly via adaptive k-mer weighting and repeat separation. *bioRxiv*.

Lee, C., Grasso, C., and Sharlow, M. F. (2002). Multiple sequence alignment using partial order graphs. *Bioinformatics*, 18(3):452–464.

Li, H. (2016). Minimap and miniasm: fast mapping and de novo assembly for noisy long sequences. *Bioinformatics*, page btw152.

Loman, N. J., Quick, J., and Simpson, J. T. (2015). A complete bacterial genome assembled de novo using only nanopore sequencing data. *Nat Meth*, 12(8):733–735.

Madoui, M.-A., Engelen, S., Cruaud, C., Belser, C., Bertrand, L., Alberti, A., Lemainque, A., Wincker, P., and Aury, J.-M. (2015). Genome assembly using nanopore-guided long and error-free dna reads. *BMC Genomics*, 16:327.

Myers, E. W., Sutton, G. G., Delcher, A. L., Dew, I. M., Fasulo, D. P., Flanigan, M. J., Kravitz, S. A., Mobarry, C. M., Reinert, K. H., and Remington, K. A. (2000). A whole-genome assembly of drosophila. *Science*, 287(5461):2196–2204.

Myers, G. (2014). *Efficient local alignment discovery amongst noisy long reads*, pages 52–67. Springer.

Nagarajan, N., Read, T. D., and Pop, M. (2008). Scaffolding and validation of bacterial genome assemblies using optical restriction maps. *Bioinformatics*, 24(10):1229–1235.

Pop, M. (2004). Shotgun sequence assembly. *Advances in computers*, 60:193–248.

5 SUPPLEMENTARY MATERIAL

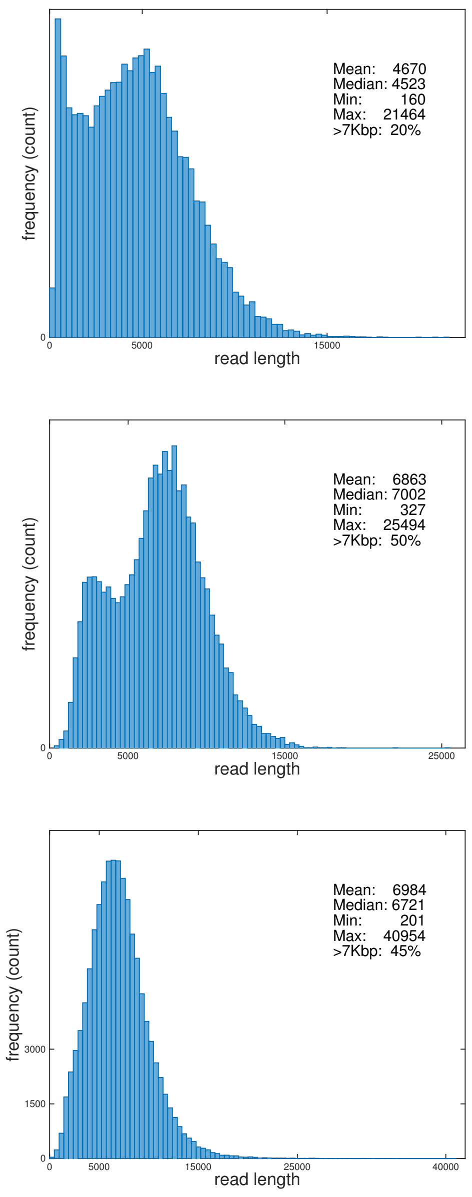


Fig. 8: Histogram of read length for the three datasets (top : *A. baylyi* middle : *E. coli*, bottom : *S. cerevisiae*).

Outliers removal for the *S. cerevisiae* genome

For prokaryotic genomes, we manage to find the layout of the reads without any correction, but for more complex genomes such as S. cerevisiae, our pipeline suffers from the high number of outliers in the similarity matrix. To deal with them, an idea is to identify the subset of reads encompassing repetitive regions and correct this subset of reads before the layout phase. Unfortunately, it appears there are two kinds of outliers, as shown in Figure 9. Some are due to structural repeats that occur many times in the genome; they are easy to identify and can be removed by simply thresholding the similarity matrix. However, outliers remaining after thresholding are more difficult to identify as they don't have significantly more overlaps than reads in non-repetitive regions. Importantly, a small number of such outliers remain (especially at the ends of the chromosomes, cf Figure 10) even with perfectly corrected reads. In order to find one or a few contigs per chromosome, the pipeline would have to be augmented with an algorithm that exploits the topology of the overlap graph, as done in `miniasm` and `canu`, and/or use additional information coming from optical maps for example.

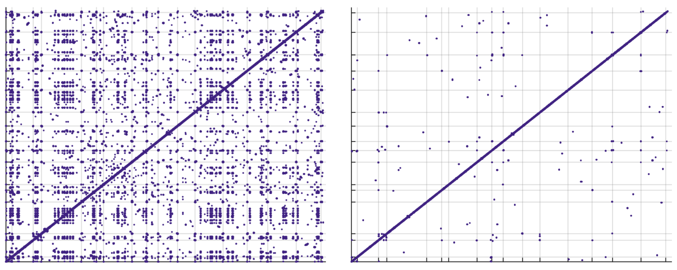


Fig. 9: Similarity matrices of *S. cerevisiae* from `minimap` in the true order (obtained by mapping against the reference genome), with no threshold (Left), and with only the highest 40% values kept (Right). The grid corresponds to the separation between chromosomes. Most of the outliers correspond to structural repeats that are removed with by the thresholding. The outliers that remain on the right subfigure are more challenging to identify *ab initio*.

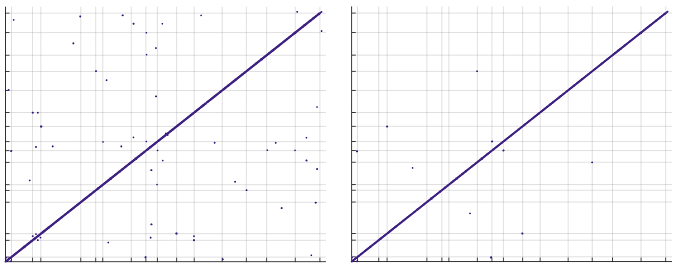


Fig. 10: Similarity matrices of *S. cerevisiae* from `minimap` in the true order (obtained by mapping against the reference genome), after thresholding in order to keep 50 connected components. The grid corresponds to the separation between chromosomes. The left matrix is obtained from the raw (uncorrected) reads, whereas the right one was built from error-free reads. Those simulated error-free reads were generated by extracting the interval from the reference genome to which they align. A few outliers remain even with exact sequences, most of them being located at the edges of the chromosomes, hence could correspond to telomeric sequences.

Table 2. QUAST report : A. baylyi assembly.

Assembly	acino-R2
# contigs ($\geq$ 0 bp)	1
# contigs ($\geq$ 1000 bp)	1
Total length ($\geq$ 0 bp)	3549516
Total length ($\geq$ 1000 bp)	3549516
# contigs	1
Largest contig	3549516
Total length	3549516
Reference length	3598621
GC (%)	40.91
Reference GC (%)	40.43
N50	3549516
NG50	3549516
N75	3549516
NG75	3549516
L50	1
LG50	1
L75	1
LG75	1
# misassemblies	4
# misassembled contigs	1
Misassembled contigs length	3549516
# local misassemblies	2
# unaligned contigs	0 + 0 part
Unaligned length	0
Genome fraction (%)	99.407
Duplication ratio	0.992
# N's per 100 kbp	0.00
# mismatches per 100 kbp	169.71
# indels per 100 kbp	1174.35
Largest alignment	1708619
NA50	859748
NGA50	859748
NA75	580389
NGA75	580389
LA50	2
LGA50	2
LA75	3
LGA75	3

Table 3. QUAST report : E. coli assembly.

Assembly	ecoli-R2
# contigs ($\geq$ 0 bp)	1
# contigs ($\geq$ 1000 bp)	1
Total length ($\geq$ 0 bp)	4661933
Total length ($\geq$ 1000 bp)	4661933
# contigs	1
Largest contig	4661933
Total length	4661933
Reference length	4641652
GC (%)	51.00
Reference GC (%)	50.79
N50	4661933
NG50	4661933
N75	4661933
NG75	4661933
L50	1
LG50	1
L75	1
LG75	1
# misassemblies	5
# misassembled contigs	1
Misassembled contigs length	4661933
# local misassemblies	30
# unaligned contigs	0 + 0 part
Unaligned length	0
Genome fraction (%)	99.458
Duplication ratio	1.010
# N's per 100 kbp	0.00
# mismatches per 100 kbp	114.81
# indels per 100 kbp	758.84
Largest alignment	3352617
NA50	3352617
NGA50	3352617
NA75	768178
NGA75	768178
LA50	1
LGA50	1
LA75	2
LGA75	2

Table 4. QUAST report : S. cerevisiae assembly.

Assembly	euk12M_S288C-R1
# contigs ($\geq$ 0 bp)	261
# contigs ($\geq$ 1000 bp)	261
Total length ($\geq$ 0 bp)	13551327
Total length ($\geq$ 1000 bp)	13551327
# contigs	261
Largest contig	240825
Total length	13551327
Reference length	12157105
GC (%)	38.94
Reference GC (%)	38.15
N50	79993
NG50	85722
N75	46178
NG75	54849
L50	55
LG50	47
L75	112
LG75	91
# misassemblies	9
# misassembled contigs	7
Misassembled contigs length	292829
# local misassemblies	13
# unaligned contigs	39 + 111 part
Unaligned length	825735
Genome fraction (%)	94.964
Duplication ratio	1.102
# N's per 100 kbp	0.00
# mismatches per 100 kbp	233.54
# indels per 100 kbp	1247.78
Largest alignment	237988
NA50	76658
NGA50	82006
NA75	41871
NGA75	52453
LA50	56
LGA50	48
LA75	117
LGA75	94

Table 5. *Sacharomyces cerevisiae* assembly results by chromosome. The assembled contigs were evaluated with QUAST for each chromosome (only a subset of the QUAST descriptive statistics is shown here).

Chr.	Size (kb)	Contigs	PID (%)	Misassmbl.	Genome frac. (%)
I	230	6	98.4	0	83.9
II	813	13	98.6	0	98.2
III	317	12	98.3	1	98.8
IV	1532	33	98.4	2	87.6
V	577	13	98.4	0	97.2
VI	270	8	98.3	0	96.0
VII	1091	23	98.7	2	97.3
VIII	563	17	98.3	1	94.6
IX	440	9	98.7	0	96.4
X	746	13	98.7	0	95.8
XI	667	11	98.7	0	100.0
XII	1078	19	98.5	0	95.1
XIII	924	16	98,6	2	94.1
XIV	784	15	98.5	0	99.1
XV	1091	26	98.6	0	99.1
XVI	948	27	98.4	0	97.8
Chrmt.	67	0	-	-	-

Simulated optical mapping for Sacharomyces cerevisiae

We performed an experiment to evaluate the extent to which optical mapping could improve long-range anchoring of our 261 *S. cerevisiae* contigs and provide an alternative consistency check of the assembly. A restriction map was generated *in silico* from the reference *S. cerevisiae* genome with the BamHI restriction site (GGATCC), yielding one map per chromosome. This simulated optical map represents a best-case scenario since real optical measurements lack some precision and are obtained through an error-prone assembly process. Using the same algorithm as for the *A. baylyi* genome, we obtained the results summarized in Figures 11 and 12. As expected, contig sequences with higher number of occurences of specific restriction sites (RS) are correctly mapped more often than contig sequences with few RS occurences. In particular, contig sequences containing more than 10 distinct RS are always correctly mapped in this experiment (Figure 11). The number of distinct RS depends on the length of the contig sequence. Contig sequences larger than 50kbp are correctly mapped (Figure 12).

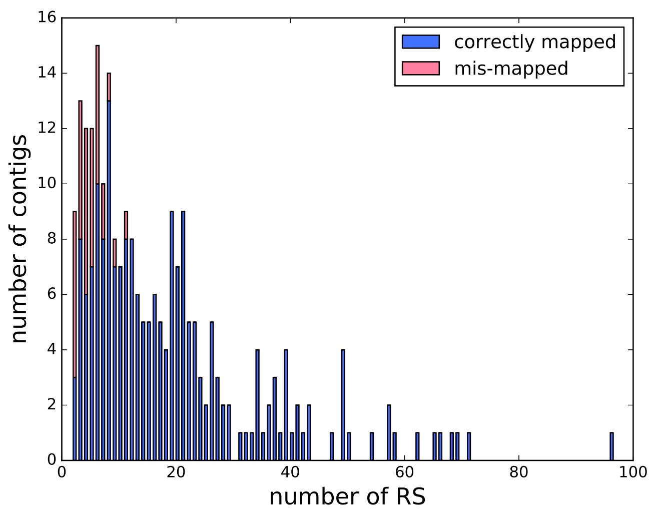


Fig. 11: Bar plot showing the number of contigs as a function of the number of distinct restriction site (RS) occurences in their sequence. For a given number of RS occurrences, the blue part of the bar shows the fraction of contigs correctly aligned to the theoretical restriction map, whereas the red part corresponds to the complementary fraction of unperfectly aligned contigs. All contigs with more than 10 RS occurences are correctly mapped.

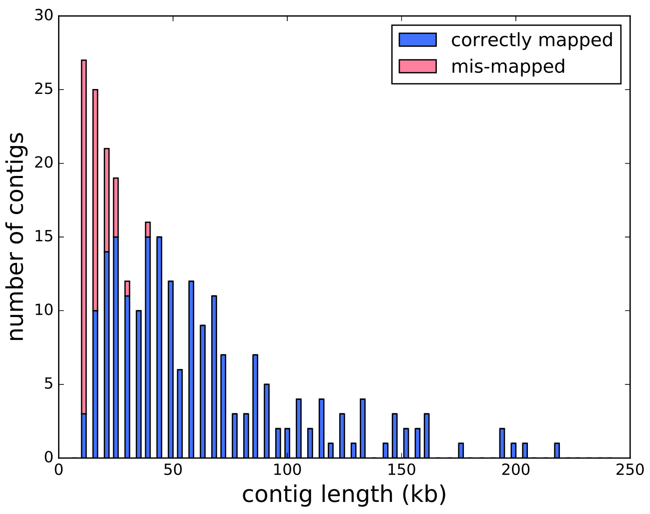


Fig. 12: Bar plot showing the number of contigs as a function of the contig length. For a given contig length, the blue part of the bar shows the fraction of contigs correctly aligned to the theoretical restriction map, whereas the red part corresponds to the complementary fraction of unperfectly aligned contigs. All contigs larger than 50kbp are correctly mapped.