De novo Clustering Nanopore Long Reads of Transcriptomics Data by Gene

Camille Marchet*, Lolita Lecompte*, Corinne Da Silva†, Corinne Cruaud†,
Jean Marc Aury†, Jacques Nicolas*, Pierre Peterlongo*
*IRISA - INRIA - Université Rennes 1
263 avenue du Général Leclerc, 35042 Rennes, France, email: camille.marchet@irisa.fr
†Commissariat à l´ Energie Atomique (CEA)
Institut de Biologie Fran cois-Jacob, Genoscope, F-92057 Evry, France

***Abstract*—This work addresses the problem of assigning a set of long reads issued from a *de novo* transcriptomics study to clusters by genes they originate from. The different transcripts of a gene give long reads sharing similar sequences and our work makes use of this fact to retrieve the right cluster of reads for each gene from the graph of similarity between reads. We propose a method based on the use of the clustering coefficient (CC) and the search of a minimal cut in the graph with a greedy procedure favoring nodes with a high degree and high CC. Our approach compares favorably to state of the art methods. We provide results on the mouse brain transcriptome which show that the approach achieves a high precision level and a good level of recall despite not using any reference genome.**

I. Introduction

The study of gene expression in a cell is commonly made using RNA-seq technology where short reads are generated from the sequences of RNA molecules. Biological processes like alternative splicing in eukaryotes have been investigated through this technique, including *de novo* studies where no reference genome or transcriptome were available. However, short reads do not span the full length of RNA transcripts. Therefore a transcript reconstruction step is needed before analysing an observed population of RNA molecules. Such step shows some limitations in particular when the sequences involve repeats [1]. Whole-length transcript reconstruction without ambiguity is also a computationally hard problem. [2].

Lately, long read sequencing technologies (Pacific Bioscience [3] and Nanopore [4]) have brought the opportunity to sequence full-length RNA molecules. In doing so they relax the previous constraint of transcript reconstruction via assembly or mapping, prior to study complete RNA transcripts. At the same time, this is at the cost of a higher error rate (up to 15%). Such error rate combined with error profiles in favour of insertion/deletion causes new computational problems. Many recent studies described alternative transcripts within eukaryote transcriptomes using long reads, some reporting novel variants discovery [5], [6]. In these works, the studied species already benefited from an assembled reference genome or transcriptome and the analysis was based on read correction then mapping on the reference [7], [8]. However, for all species that do not have such reference, tools are still lacking for the *de novo* description of the content of a transcriptome sequenced in long, spurious reads.

This paper addresses the issue of assigning transcriptomic long reads to the gene from which they originate. The problem is not new, it dates back before the advent of NGS, with Sanger sequencing and the necessity to cluster ESTs. Tools such as STACK [9] or d2_cluster [10], were designed to perform such task. Early approaches like TCLUST [11] performed EST clustering by employing a simple transitive-closure clustering algorithm. However these methods were tailored to work with lower scalabilty challenges due to the scarcity of data, and a far less important error rate than with current long reads.

In [12] was proposed a clustering procedure for the characterization of genome-wide repetitive sequences in 454 genomics data sets. This tool was based on the concept of *community detection*, which is a natural way of depicting our problem. Communities are densely connected groups of nodes, although there exists no rigorous shared definition. Usually nodes belong to a community because they have something (property related to their nature, their role) in common. A common requirement when defining communities is that the graph has the small world property, i.e. the number of edges in the graph is in the order of the number of nodes $\mathcal{N}$, and the shortest path between two nodes has a length in the order of $log(\mathcal{N})$. Due to the ambiguity of the community definition, a plethora of methods have been proposed for their detection. Moreover this problem has appeared in many disciplines, taking many slightly different forms according to the application domain. The first approach that brought an important contribution to communities detection is a divisive algorithm based on *modularity* [13]. A popular and more recent work is the Clique Percolation Method [14].

In this paper we describe an algorithm to perform *de novo* clustering on a graph of similarity of sequences with spurious edges. In addition we propose an implementation of this algorithm in a tool dubbed CARNAC (**C**lustering coefficient-based **A**cquisition of **RNA C**ommunities) that outputs clusters of transcriptomic long reads by genes. This way, in the case of eukaryotic transcripts, it is expected that two alternative isoforms coming from the same gene will be clustered altogether. CARNAC needs no pre-processing of the data (such as read correction) and performs its clustering task while being very precise and thus reliable. It has also the advantage of being parameter-free. We compared our tool to state of

the art methods and tested our approach on real Nanopore reads from the mouse transcriptome. Results on such a model species were validated by comparing them to mapping on reference genome results. In the discussion we then outlined the perspectives of this work.

Availability: CARNAC was developped for Linux. It is written in C++ and is disponible at github.com/kamimrcht/CARNAC.

II. Methods

A. Problem description

Our problem can be formalized as a community detection problem, where a community is the population of reads coming from a same gene and in fact we will use indifferently the words cluster and community in the rest of this paper. The input is a graph of similarity between nodes that represent the reads. Two nodes are connected by an edge if the two corresponding reads present enough similarity. We expect reads from a same gene to be connected with one another as they are likely to share exons. In our application case, as a consequence of gene expression, all RNA molecules are not present in the same quantity in the cell. This leads to a large heterogeneity in terms of read abundance, and we do not expect an homogeneous distribution of cluster sizes. Particularly, we expect the presence of many small communities. We target a technique that can be used on large graphs and that offers a trade-off between high precision and speed.

As for community detection, the paper considered as fundamental to face this problem was proposed by Newman [13], using the *modularity* concept. *Modularity* measures the difference between the fraction of edges within a same cluster and the fraction of edges that would be observed by chance given the degree of each node. Other methods were proposed to improve this technique, in particular methods relaxing the definition of communities as objects that can overlap, thus proposing a cover rather than a partition of the graph. The most popular method to find overlapping communities is the Clique Percolation Method (CPM) [14]. Such approach has already been validated for the detection of communities in biological context [15].

We propose a clustering technique that aims at finding cuts in graphs of read similarities in order to make a partition that separates subgraphs gathering similar reads, while taking account of the quasi-clique nature of these subgraphs. In order to evaluate the *cliqueness* of read's neighborhood, our method computes different measures that are described in section II-C.

B. Input graph generation

Recently, several mapping methods based on k-mers were developed to efficiently compute similarity or overlaps between long reads ([16], [17], [18]), which avoid a quadratic number of pairwise comparisons between reads. However these tools only compute pairwise overlaps between reads and do not perform clustering. We obtain a read similarity graph by applying a mapping method on a multi-set of reads versus itself. Any read in the output of this method is a node of the graph. Anytime the method highlights a similarity between two reads, an edge is drawn between the pair. In this work we chose the tool Minimap [18] for its efficiency and its very high level of precision, which is critical for our application.

In the ideal case, the mapping method's output would contain all links between similar reads and lead to no false connection. It follows that each connected component would be a clique as each read should pair with any other. However, the methods of the state of the art use heuristics and some false negative and false positive links are produced. The results also suffer from the very high error rates in the data (as confirmed in Table I). This leads to a few false connections between subgraphs of similar reads as well as missing links between pairs of reads. Thus one should rather expect pseudo-cliques than cliques in the graph.

C. Clustering step

1) Algorithm overview: Our method makes no assumption on the number of clusters, nor on the size distribution, and it needs no parameters. Communities are defined locally, and built iteratively following an ordering depending on the nodes degrees and clustering coefficients (CC). Starting by seed nodes that root with their neighborhoods the different communities, the boundaries are successively refined to build a partition of communities by assigning nodes of their intersections to a single community. Then different partition results are compared for different cutoff values fixing an upper bound for the CC and we the one that induces the minimal cut of the graph is kept.

The procedure to detect communities can be outlined the following way (see also Figure 1). Let $\mathcal{G} = (\mathcal{N}, \mathcal{E})$ be an undirected graph of reads. Let n_i be a node from $\mathcal{N}$ and $N_i \subset \mathcal{N}$ its direct neighborhood. Let $deg(n_i)$ be the number of edges connecting n_i to its direct neighbors, i.e. $deg(n_i) = |N_i|$.

1) Compute and store clustering coefficient CC_i and degree $deg(n_i)$ for each node n_i (described in II-C2)
2) Set the *cutoff* value to one of the CC_i values. Compute communities:
   a) If a given node n_i has its CC_i greater than or equal to *cutoff*, build the community $s_i = \{n_i\} \cup N_i$.
   b) All pairs of communities issued from two nodes n_i and n_j with non empty intersection are compared. A decision is taken to either merge them or remove nodes from the intersection from one of them, so that their intersection becomes empty. The criterion for this decision is based on the generalized clustering coefficient CC_{ij} of the two neighborhoods N_i and N_j (defined in II-C3). Shortly:
      - If the CC_{ij} is greater than or equal to *cutoff*, merge the communities
      - Else, assign the nodes of the intersection to one community or the other depending on the option that minimizes the cut in the subgraph.
3) Steps 1 and 2 are repeated on all stored values of CC. Each time the global cut is updated.

4) Select the value of CC that minimizes the global cut value in the graph. Output the corresponding clusters.

Details are given in Algorithm 1. The rationale behind the search of a minimal cut in the graph is that the mapping procedure to detect overlaps produces not too many false positives. Thus most of the edges in the initial graph have to be kept during the clustering.

2) Selection of seed nodes and communities initiation: For a node $n_i \in N$ with degree $deg(n_i) > 1$ and a neighborhood $N_i \subset \mathcal{N}$, the *local clustering coefficient* is:

$$CC_i = \frac{2 \ |\{(n_j, n_k) \in \mathcal{E} : n_j, n_k \in N_i\}|}{deg(n_i) \times (deg(n_i) - 1)} \quad (1)$$

Nodes of degree 0 and 1 have a CC of 1.

This local coefficient represents the *cliqueness* of the $N_i \cup n_i$ set of nodes. It divides the actual number of edges in the set by the maximal possible number of edges, which is reached for a clique. The closer to 1, the more the set of nodes forms a complete subgraph. By contrast, if the coefficient is close to 0, the nodes of the set are weakly connected.

We sort the nodes first by degree and second by CC. At a given cutoff, the procedure starts by computing the local CC for each node. The nodes which CC is above or equal to the cutoff are seeds that originate communities. One starts by including themselves and their direct neighbors in a community. At this point it is possible that two communities overlap.

3) Refinement of community boundaries: In a second time, the intersecting communities are looked up pairwise in order to assign nodes of the intersection to only one community and realize a partition. This step relies on two procedures: split (see Algorithm 2) and merge communities. In order to decide if two intersecting communities must be merged in a bigger one or if the intersection must be split, we again use the *cliqueness* notion. In order to compute the *cliqueness* of a union of two nodes n_i and n_j, we introduce an *aggregated clustering coefficient*:

$$CC_{ij} = \frac{2 \ |\{(n_k, n_l) \in \mathcal{E} : n_k, n_l \in N_i \cup N_j\}|}{|N_i \cup N_j| \times (|N_i \cup N_j| - 1)} \quad (2)$$

This generalized clustering coefficient is used when two nodes have intersecting neighborhoods. It helps deciding if the neighborhoods of n_i and n_j must remain separated quasi-cliques or if the *cliqueness* gains to scale to the union of the two neighborhoods. If the CC_{ij} is greater or equal than the current cutoff, then the two communities are merged. In the other case, the nodes of the intersection are reported to only one of the two communities by realizing the minimal cut between the two possible cuts. If a node is not in a set anymore we remove any edge that connects it to the element of the set, which gives a cut value when computed on all nodes of the intersection. In the case both cuts are equal, we choose the cut minimizing the $\Delta CC_{N_1,N_{1'}}$, defined for two sets N_1, $N_{1'}$, $N_{1'} \subseteq N_1$:

$$\Delta CC_{N_1,N_{1'}} = CC_{N_1} - CC_{N_{1'}} \quad (3)$$

With CC calculated such as in Equation 2, where N_1 (resp. $N_{1'}$) is the union of nodes. Let be $N_1 =$ and N_2 two clusters, $I = N_1 \cap N_2$. By computing $\Delta CC_{N_1,N_1 \setminus I}$ for a cluster N_1, we are enabled to compare the gain or loss of *cliqueness* when nodes from I are removed. The $\Delta CC_{N_1,N_1 \setminus I}$ can be negative when the removal of elements from I increased the *cliqueness*. Thus we perform the cut associated to $min(\Delta CC_{N_1,N_1 \setminus I}, \Delta CC_{N_2,N_2 \setminus I})$ in order to maximize the local *cliqueness*.

The global result depends on the order in which pairs of clusters are compared. This order is carefully designed. First the community associated to the node of maximal degree (and secondly maximal CC) is compared to the community intersecting it that is associated with the node of second maximal degree (and secondly maximal CC), the intersection is resolved and the first community is updated. Then it is compared to the third community that intersected it if it exists, and so on until all intersections are solved.

D. Implementation choices

Clustering algorithms often present a quadratic step. In our case the algorithm can be cubic in the worst case because we explore the whole space of clustering coefficients found in the graph, while performing pairwise comparisons of clusters. For a given connected component, the number of clustering coefficients can be equal to the number of nodes. In practice, key features have proved to be very efficient to reduce the complexity.

1) Approximated minimal cut: In large connected components, many clustering coefficients values are very close. Introducing a rounding factor in Equation (1) results in a neat decrease of the number of different values observed, and thus restrains drastically the number of iterations necessary for the main loop (line 7 of Algorithm 1). As a consequence the optimization only computes an upper bound of the minimal cut, an acceptable compromise since it has no impact on the precision level. In the section Results, we show that this has in fact an overall positive impact on result quality.

2) Graph pre-processing: The most costly phase relies on the treatment of the largest connected components. A straightforward pre-processing splits the connected components of the graph in smaller ones. This should preserve most of the edges. We chose to disconnect the articulation points of the graph to remove nodes that can be targeted as potential bridges between two real clusters. These are nodes whose removal increases number of connected components components in the graph. For our case study, such nodes can be spotted as problematic for we do not expect that a single read is the only link between many others. Their detection can be done with a DFS in time complexity of $\mathcal{O}(\mathcal{N} + \mathcal{E})$ for the whole graph. Simulations shown that the best results occur when a first removal is performed on the whole graph and a second time within each connected component.

III. Results

All experiments were run on Linux distribution with 24 Intel Xeon 2.50GHz processors, using 40 threads and with 200 GB

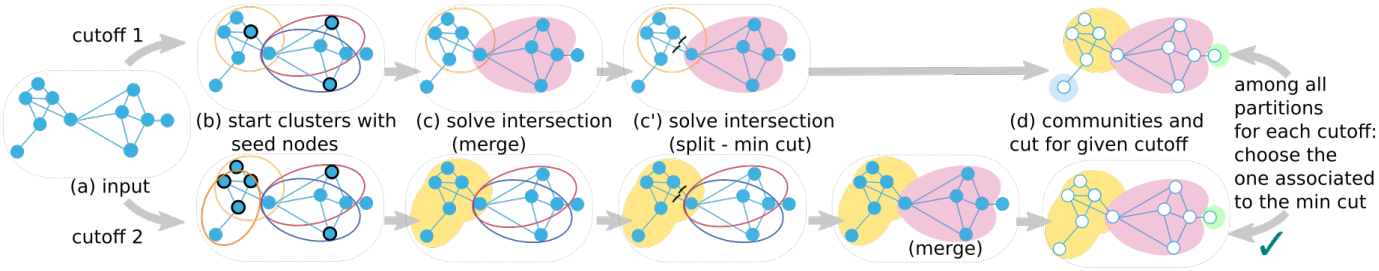

Fig. 1. Summary of the algorithm. (a) All CC and degrees are computed (lines 2-6 from Algorithm 1). Each CC value is a cutoff. For a given cutoff, (b) seed nodes (black stroke) initiate clusters with their neighborhood (lines 9-17). (c, c') Intersection between clusters are solved (lines 18-26) either by (c) merging them or by (c') splitting (Algorithm 2). (d) For each cutoff a partition is output. When compare them and keep the one associated with the minimal cut, that correspond to the final communities (lines 27-30).

RAM.

A. Data

1) RNA MinION sequencing: cDNA were prepared from 4 aliquots (250ng each) of mouse commercial total RNA (brain, Clontech, Cat# 636601 and 636603), according to the Oxford Nanopore Technologies (UK) protocol 1D cDNA by ligation (SQK-LSK108). The data generated by MinION software (MinKNOWN, Metrichor) were stored and organized using a Hierarchical Data Format. FASTA reads were extracted from MinION HDF files using poretools [19].

2) Reference community clusters from read mapping: Reference clusters were made to allow comparison and evaluation of the results of community detection methods on RNA long reads transcriptome instances. Nanopore reads were aligned to the masked mouse genome assembly (version GRCm38 + the ribosomal region extracted from NT database, BK000964 and GU372691) using BLAT [20]. For each read, the best matches based on BLAT score (with an identity percent greater than 90%) were selected. Then, those matches were realigned on the unmasked version of the genome (extended of 1Kb on both extremities) using Est2genome [21]. Next, Nanopore reads were clustered according to their genomic positions, two reads were added in a given cluster if they share at least 10nt in their exonic regions. Reads that mapped onto the mitochondrial and ribosomal sequences were excluded before the clustering procedure was performed. Moreover, one region on chromosome 1 which correspond to an unprocessed pseudogene was excluded of the clustering step as it harbors a high number of Nanopore reads (>10k).

B. Method evaluation

1) Raw results and impact of heuristic choices: We first assess the impact of CARNAC heuristics on the quality of results and on performances. We chose a subset of 10K reads (10,183 mouse reads within 207 "true" clusters determined by mapping). In order to generate the input graph, Minimap version 0.2 was launched with parameters tuned to improve recall (`-Sw2 -L100 -t10`). To assess the results, we used recall and precision measures defined according to the expected clusters defined by mapping. We compute true positives (tp) nodes for any cluster (equivalent to the reads expected

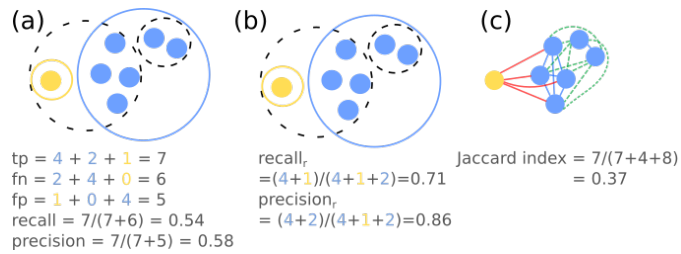

Fig. 2. Summary of the measures used to assess the results. Expected clusters are circles of different colors and result clusters are dashed circles. Reads are color-filled dots. In (a) we compute recall and precision (Equation 4) based on true positives, false positives and false negatives reads by comparing expected and result clusters. In (b), we only take into account the most representative clusters to compute recall$_r$ and precision$_r$ (Equation 5). In (c) we show the Jaccard index (Equation 6) in this example. True positives pairs (a_{11}) are represented in blue, false negative pairs (a_{10}) are dashed in green and false positives pairs (a_{01}) are in red.

when looking at the mapping and present in the results), false positives (fp) nodes (those output in the results that are wrong according to the mapping), and false negative nodes (fn) (those that were not reported in the results but that exist according to the mapping). The final TP, FN and FP values are sums of tp, fn and fp for all the clusters. We computed a precision P, a recall R and the F-measure:

$$P = \frac{TP}{TP + FP} \quad R = \frac{TP}{TP + FN} \quad F_m = 2.\frac{P \times R}{P + R} \qquad (4)$$

The F-measure combines the precision and the recall to summarize them. Being the result of the weighted harmonic mean between the two measures, it gives them an equal importance.

We computed results using the raw algorithm (CARNAC raw), and using both heuristics: approximated minimal cut and graph pre-processing (CARNAC optimized). Results are presented Table I. The optimized version presents an important decrease of the run time, and in the meantime has a very positive impact of the precision. The recall is sightly impacted. In the rest of the document, we will refer to CARNAC as its optimized version.

2) Comparison to state of the art methods: We show results of state of the art algorithms and expose a comparison to our tool's results. We selected the same input graph than the previous section, as we wanted to focus on the quality of

Algorithm 1: Main algorithm for the clustering based on minimal cut to find pseudo-cliques. The **Compute_Cut** step is detailed in Figure 2. The Merge step, which is trivial, is not detailed.

1 *Min_Cut_CC*
Data: Graph $\mathcal{G} = (\mathcal{N}, \mathcal{E})$
Result: A cutoff CC_{min}, and a partition P of $\mathcal{N}$
$P = \{\mathcal{C}(n_1), \mathcal{C}(n_2), ...\mathcal{C}(n_k)\}$ such that
all $\mathcal{C}(n_i)$ have a clustering coefficient $\geq CC_{min}$,
and result in a minimal cut of $\mathcal{G}$ with value Cut_{min}
2 **foreach** *node n of $\mathcal{N}$* **do**
3 Compute its degree $deg(n)$;
4 Compute its clustering coefficient $CC(n)$ (equation 1);
5 $LN = sort(\mathcal{N})$; *% Sorted list in decreasing $deg(n)$, then of value of $CC(n)$*
6 $CC_{min} = 0$; $Cut_{min} = \infty$
7 **% Loop over possible cutoff values**
8 **foreach** *cutoff in CC* **do**
9 **% Compute possibly overlapping clusters**
10 **foreach** *node n of $\mathcal{N}$* **do**
11 $Lclust(n) = \emptyset$; *% Sorted list of clusters containing n*
12 **foreach** *n in LN such that $CC(n) \geq$ cutoff* **do**
13 $\mathcal{C}(n) = \{n\} \cup neighbours(n)$; *% Cluster= star centered on n*
14 **foreach** *x in $C(n)$* **do**
15 *% incremental sort, in decreasing value of $CC(n)$*
16 $Lclust(x)$= sort($Lclust(x) \cup \{n\}$);
17 LN_{cl}=*sort(N) % in decreasing $deg(n)$, then $CC(Lclust(n)[1])$*
18 **% Make a partition from overlapping clusters**
19 **foreach** *node n in LN_{cl}* **do**
20 x= $Lclust(n)[1]$;
21 **foreach** *y in $Lclust(n)[2 : length(Lclust(n))]$* **do**
22 Compute the clustering coefficient CC_{xy} (equation 2);
23 **if** $CC_{xy} \geq$ *cutoff* **then**
24 x=**Merge**(x, y); % $\mathcal{C}(x) = \mathcal{C}(x) \cup \mathcal{C}(y)$
25 **else**
26 x=**Cut**(x, y, $\mathcal{E}$) (algo[2]); *% n is discarded from one of the clusters, x refers now to the cluster containing n*
27 Cut= number of inter-cluster edges in $\mathcal{E}$;
28 **% Update partition with the minimal cut value**
29 **if** $Cut < Cut_{min}$ **then**
30 $Cut_{min} = Cut$; CC_{min} =cutoff; $P = \{\mathcal{C}(n)\}$

Algorithm 2: Node removal from one of two sets whose intersection is non empty. This procedure chooses the set to shrink by keeping the minimal cut between the two. In case of two equals cuts, this procedure attributes the nodes of the intersection to the set that has the greatest gain or the lowest loss of connectivity when keeping the nodes of the intersection.

1 *Cut*
Data: Graph $\mathcal{G} = (\mathcal{N}, \mathcal{E})$.
$\mathcal{C}_i$ and $\mathcal{C}_j$ two clusters of nodes $\subset \mathcal{N}$ with a non null intersection $\mathcal{I}_k = \mathcal{C}_i \cap \mathcal{C}_j$.
Cl is the set of clusters.
2 $cut_{\mathcal{C}_i} = |e_{lm} : n_l \in \mathcal{C}_i \setminus \mathcal{I}_k, n_m \in I_k, e_{lm} \in \mathcal{E}|$;
3 $cut_{\mathcal{C}_j} = |e_{lm} : n_l \in \mathcal{C}_j \setminus \mathcal{I}_k, n_m \in I_k, e_{lm} \in \mathcal{E}|$;
4 **if** $cut_{\mathcal{C}_i} > cut_{\mathcal{C}_j}$ **then**
5 Remove nodes of $\mathcal{I}_k$ from C_j;
6 **if** *$\mathcal{C}_j$ is not connected anymore* **then**
7 Split($\mathcal{C}_j$) using steps 11, 12, and 13 of main algorithm[1];
8 Append new clusters in Cl;
9 **if** $cut_{\mathcal{C}_i} < cut_{\mathcal{C}_j}$ **then**
10 Remove nodes of $\mathcal{I}_k$ from $\mathcal{C}_i$;
11 **if** *$\mathcal{C}_i$ is not connected anymore* **then**
12 Split($\mathcal{C}_i$) using steps 11, 12, and 13 of main algorithm[1];
13 Append new clusters in Cl;
14 **if** $cut_{\mathcal{C}_i} == cut_{\mathcal{C}_j}$ **then**
15 Compute $\Delta CC_{\mathcal{C}_i, \mathcal{C}_i \setminus \mathcal{I}}$ and $\Delta CC_{\mathcal{C}_j, \mathcal{C}_j \setminus \mathcal{I}}$ (equation 3);
16 **if** $\Delta CC_{\mathcal{C}_i, \mathcal{C}_i \setminus \mathcal{I}} \leq \Delta CC_{\mathcal{C}_j, \mathcal{C}_j \setminus \mathcal{I}}$ **then**
17 Do steps 5 to 8.
18 **else**
19 Do steps 10 to 13.

	Recall (%)	Prec. (%)	F-measure (%)	Time (h:mm:ss)	Mem (GB)
Carnac raw	**69.73**	53.03	60.24	5:01:58	31.3
Carnac optimized	65.0	**92.80**	**76.45**	**0:0:25**	**0.907**

TABLE I

Comparison of quality and performances between the raw algorithm implementation and the optimized implementation. The benchmark was realized on a 10K reads data set from the mouse brain transcriptome. Both versions were launched with 40 threads. Time is given in wall-clock. Formulas for recall, precision and F-measure are given in Equation 4 and examples are given in Figure 2.

results. These recall and precision measures are completed by recall$_r$ and precision$_r$ measures based on representative graphs [22]. These measures were already used to assess the relevance of biological sequences clustering [23]. Let be L the number of result clusters $\{\mathcal{C}_1, \ldots \mathcal{C}_i\}_{1 \leq i \leq L}$ and K the number of expected clusters $\{\mathcal{K}_1, \ldots \mathcal{K}_j\}_{1 \leq j \leq K}$. Let R_{ij} be the number of nodes from $\mathcal{C}_i$ that are in expected cluster $\mathcal{K}_j$.

We compute a recall$_r$ and a precision$_r$ such as:

$$R_r = \frac{\sum_{j=1}^{K} max_i(R_{ij})}{\sum_{i=1}^{L}\sum_{j=1}^{K} R_{ij}} \quad P_r = \frac{\sum_{i=1}^{L} max_j(R_{ij})}{\sum_{i=1}^{L}\sum_{j=1}^{K} R_{ij}} \tag{5}$$

Measures R_r and P_r are not tailored to express the correctness of the partition but they indicate the confidence that can be placed in the method, according to its ability to retrieve information and to be precise.

In order to compare the different algorithms, we relied on yet another measure. Let $\mathcal{X}_0$ be the partition given by the clusters obtained by mapping, that we consider as "true" clusters, and $\mathcal{X}$ the partition obtained using a given clustering method. Then a_{11} is the number of pairs of nodes that are placed in a same community in each partition. a_{00} indicates the number of pairs for which nodes are placed in different communities in both partition. a_{10} (resp. a_{01}) is the number of pairs of nodes placed in the same community in the case of $\mathcal{X}_0$ (resp. $\mathcal{X}$) but in different communities in the partition $\mathcal{X}$ (resp. $\mathcal{X}_0$). Based on those, metrics show the adequation between the two partitions described, such as the Jaccard index:

$$J(\mathcal{X}_0, \mathcal{X}) = \frac{a_{11}}{a_{11} + a_{10} + a_{01}} \tag{6}$$

The Jaccard index is between 0 and 1, the closer to 1, the more the partitions agree on their common pairs. All measures are summarized in Figure 2.

We compared CARNAC to an implementation of the classical modularity-based clustering by Newman (from http://igraph.org/) and to an implementation of the Clique Percolation method (CPM, https://sites.google.com/site/cliqueperccomp/). We also included a simple transitive-closure algorithm that consists in partitioning the graph according to its connected component in order to have a reference to a straightforward approach. Results are presented in Table II. Our method has the best trade off between precision and recall on the overall clusters as shown by the F-measure. Results show it also has the highest precision. Its recall is not the highest and suffers from over-clustering, as demonstrated by the ratio between the size of set of result clusters and the size of the set of expected clusters. However, CARNAC recall is better or at least comparable to the other approaches. It also has the highest Jaccard index among all tested approaches. The *modularity*-based approach fails to find good clusters for this graph, with both low recall and precision in comparison to other approaches. The approach consisting in grouping reads by connected components has higher recall than the others but suffers from a lower precision. It can be seen looking at the representative graph measures: for each gene, this approach retrieves almost all reads in a cluster but tends to include many false positives. CPM was tested with values k from 3 to 50 (over this value no community at all was found). Results are presented only for k=5 and k=50 and summarize the behaviour of this approach on our input graph. For low values of k, the CPM outputs more clusters than for high values. However its precision is globally not very high, although the most representative clusters are very precise. This means that some of the result clusters are very accurate while others are not. For higher values of k, the results are strongly enhanced but represent only a small fraction of the input. On the contrary, our approach is more conservative and shows comparatively good results in any case.

C. Results on real size data set

For this experiment, all reads that could be mapped on the reference were taken into account (501.787). Due to repeats (paralogy, transposable elements...), some reads were mapped at multiple loci on the reference. When a given read maps on several loci, such loci are gathered in a single expected cluster (12.596 expected clusters). This means that for instance reads from copies of paralog genes that have not diverged to much or reads containing a copy of a transposable elements are expected to be in the same cluster. After running Minimap (same options than in previous sections), 60.742 reads were lost (12%). The obtained input graph $\mathcal{G}$ is defined by $(\mathcal{N} = 441.045, \mathcal{E} = 15.109.290)$, with 38.643 connected components (the biggest connected component has 94.868 nodes).

Results are presented in Table III. There were computed using 43GB RAM and in 18 minutes using 40 threads. We placed 46% of reads in their expected cluster, with a precision of 82% which means that a vast majority of reads were positioned in the right cluster. The combined recall and precision values show that our approach performs over-clustering. The representative graph-based recall and precision are consistent with this results. We then selected all result clusters which size was greater or equal to 10 elements, that could be more meaningful for users. The recall is almost the same than previously, and the precision increased from 82% to 88%. Then in order to present a visual example of the output, we used a genome browser to display read grouped by our approach.

The output of CARNAC is a text file with one line per cluster, each cluster containing the read indexes. We selected a cluster of 112 reads from our output, the reads contained in the cluster are presented mapped on the gene they come from in Figure 3. In this case there were no false positive read in the cluster. It is possible that not all read represent full isoforms, a part of them might be truncated. However, even looking only at the full-length reads according to the annotation, the figure illustrates that our approach enables to collect a diversity of RNA isoforms from the gene Pip5k1c in the same cluster. The 6 black reads on the left represent intronic sequences and share no overlap with the rest of the reads, therefore they can never appear in the same cluster using our approach.

IV. Discussion

A. Achievements

Our application, whose implementation is called CARNAC, differs from previous sequence clustering problems in bioin-

	Recall (%)	Precision (%)	F-measure (%)	Recall$_r$ (%)	Precision$_r$ (%)	F-measure$_r$ (%)	*size ratio*	Jaccard index
Connected comp.	**75.74**	66.91	74.33	75.74	5.614	13.62	1.7	0.726
Modularity	42.26	21.37	39.29	60.70	71.16	65.51	1.9	0.0972
CPM5	66.77	7.075	12.79	**79.00**	69.35	73.86	**0.92**	0.353
CPM50	43.56	92.83	59.30	49.21	89.92	63.60	0.27	0.0757
CARNAC	65.0	**92.80**	**76.45**	76.70	**98.41**	**86.62**	3.8	**0.791**

TABLE II

Comparison with state of the art methods. Size ratio divides the number of result clusters by the expected number of clusters. CPM5 (resp. CPM50) designs the CPM algorithm using $k = 5$ (resp. $k = 50$).

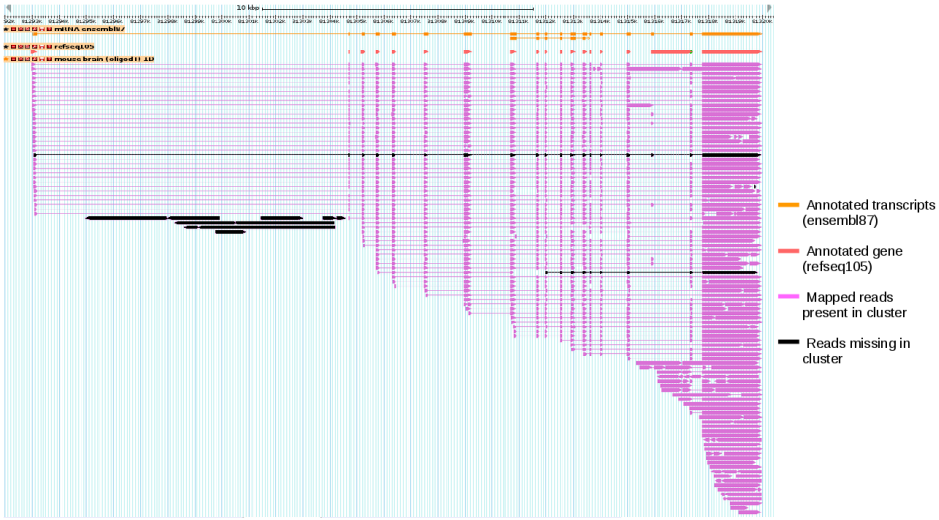

Fig. 3. 120 reads from the experiment could be mapped on the gene Pip5k1c (chr 10:81,293,181-81,319,812). Within those 120, 112 have been placed in the same cluster of total size 112 (no false positive read in this cluster). Reads are spliced-mapped, bold parts are the mapped sequences from the reads and thin parts represents the gaps between the different mapped parts of the reads. The bold parts of annotations are known exons and the thin parts are introns.

CARNAC		
Recall	all clusters	46.42
	clusters of size $\geq$ 10	46.41
Precision	all clusters	81.80
	clusters of size $\geq$ 10	88.40
F-measure	all clusters	59.23
	clusters of size $\geq$ 10	60.86
Recall$_r$	all clusters	57.08
Precision$_r$		96.58
F-measure		71.75

TABLE III

Results on full mouse brain transcriptome mapped reads, using the CARNAC optimized version with 40 threads. Recalls and precisions are shown in % (Equations 4 and 5).

formatics because of the spurious edges in the initial graph. These are due to the high error rate in long reads. We propose a method that has the characteristics of being precise and of being able to retrieve communities of heterogeneous sizes in those reads. *De novo* methods are commonly not as precise and sensitive as reference-based methods. Although we show we can scale up to a full transcriptome experiment and successfully retrieve a part of the clustering that is obtained by mapping the reads on a reference. We demonstrated that we could effectively cluster reads by genes with such data, with a very high precision. We applied a size selection procedure, by keeping only the larger clusters (of size $\geq$ 10), which had almost no impact on the recall but increased the precision 7%. We advocate for this practice as it seems to be a good filter for our clustering and seems sound since users are likely to look first at clusters that contain the most information.

We demonstrate that our approach can compete the common state of the art algorithms to detect communities. On a rather small instance already, state of the art approaches show at least a lack of precision in comparison to our approach. We show that a *modularity*-based algorithm is not well-tailored for such a problem, probably because of the heterogeneous size distribution of the clusters, and because of overlapping effects due to the repeats. Of state of the art approaches tested, only the CPM qualifies for retrieving communities in our input graphs. However, by concentrating its results in a small subsbet of communities, it lacks of recall at the global scope and not all its output clusters can be trusted. On the other hand our approach shows a good consistency. Finally Jaccard index shows it realized the best partition of all approaches.

B. Limitations and future work

The clustering of sequences from transcriptomes is made difficult by the repeats shared by the reads. This first attempt to cluster RNA reads by gene is not designed to precisely assign reads from paralog genes to their original locus. It rather gathers all reads from a gene family together, provided the different copies have not diverged too much. For the moment,

it remains undisclosed which level of divergence is needed to form two separates clusters in our application. Another related future work would be to describe how repeats like transposable elements that can be found in exons or retained introns are treated by the clustering. This clustering also shows limitations. In principle reads from a same gene should be clustered together. However if it occurs that some reads share no common exon at all, the clustering will fail to retrieve a common gene for them.

While the precision is satisfying, a perspective could be to enhance the recall of our approach. The problem is twofold. First, there exist no thirdparty tool we can use to obtain the input graph that is tailored specifically for the overlap detection between long RNA reads. Second, the algorithm we propose leads to over-clustering. This is a track we could follow in order to increase the recall and thus deliver more comprehensive description of a data set.

Finally, long reads technologies are allegedly expected to lower their error rates in the coming years. However providing methods able to deal with the current error rate remains an issue. Moreover, very recent and promising technologies for the study of transcriptomes like RNA-direct [24] by Oxford Nanopore show higher error rates than the 1D reads presented in this study, according to unpublished works. We plan to test our approach on such reads.

V. Conclusion

De novo clustering of long spurious reads is a hard task due to their current error rate and to the lack of tools to detect overlaps between RNA reads, not to mention the difficulties due to repeats. We provide an algorithm and an adequate implementation to retrieve clusters of reads by gene in such data sets. This approach is parameter-free and computing can be done in parallel. It was tested on a real transcriptome and gives results of high precision, using low time and acceptable memory resources. By gathering reads by genes or gene families we wanted to provide a prior elucidation step of transcriptome data sets, that can be followed by read correction or more precise read comparison within clusters at a highly reduced scale.

Acknowledgment

The study has been partially supported by ANR ASTER, contract ANR-16-CE23-0001, which provided the data. Authors would like to thank all members of ASTER for their advices.

References

[1] M. G. Grabherr, B. J. Haas, M. Yassour, J. Z. Levin, D. A. Thompson, I. Amit, X. Adiconis, L. Fan, R. Raychowdhury, Q. Zeng *et al.*, "Trinity: reconstructing a full-length transcriptome without a genome from rna-seq data," *Nature biotechnology*, vol. 29, no. 7, p. 644, 2011.

[2] T. Steijger, J. F. Abril, P. G. Engström, F. Kokocinski, T. J. Hubbard, R. Guigó, J. Harrow, P. Bertone, R. Consortium *et al.*, "Assessment of transcript reconstruction methods for rna-seq," *Nature methods*, vol. 10, no. 12, pp. 1177–1184, 2013.

[3] M. L. Gonzalez-Garay, "Introduction to isoform sequencing using pacific biosciences technology (iso-seq)," in *Transcriptomics and Gene Regulation*. Springer, 2016, pp. 141–160.

[4] J. L. Weirather, M. de Cesare, Y. Wang, P. Piazza, V. Sebastiano, X.-J. Wang, D. Buck, and K. F. Au, "Comprehensive comparison of pacific biosciences and oxford nanopore technologies and their applications to transcriptome analysis," *F1000Research*, vol. 6, 2017.

[5] J. D. Merker, A. M. Wenger, T. Sneddon, M. Grove, Z. Zappala, L. Fresard, D. Waggott, S. Utiramerur, Y. Hou, K. S. Smith *et al.*, "Long-read genome sequencing identifies causal structural variation in a mendelian disease," *Genetics in Medicine*, 2017.

[6] A. Byrne, A. E. Beaudin, H. E. Olsen, M. Jain, C. Cole, T. Palmer, R. M. DuBois, E. C. Forsberg, M. Akeson, and C. Vollmers, "Nanopore long-read rnaseq reveals widespread transcriptional variation among the surface receptors of individual b cells," *bioRxiv*, p. 126847, 2017.

[7] B. Wang, E. Tseng, M. Regulski, T. A. Clark, T. Hon, Y. Jiao, Z. Lu, A. Olson, J. C. Stein, and D. Ware, "Unveiling the complexity of the maize transcriptome by single-molecule long-read sequencing," *Nature communications*, vol. 7, 2016.

[8] S. Thomas, J. G. Underwood, E. Tseng, A. K. Holloway *et al.*, "Long-read sequencing of chicken transcripts and identification of new transcript isoforms," *PloS one*, vol. 9, no. 4, p. e94650, 2014.

[9] A. Christoffels, A. v. Gelder, G. Greyling, R. Miller, T. Hide, and W. Hide, "STACK: sequence tag alignment and consensus knowledge-base," *Nucleic Acids Research*, vol. 29, no. 1, pp. 234–238, 2001.

[10] J. Burke, D. Davison, and W. Hide, "d2_cluster: a validated method for clustering est and full-length cdna sequences," *Genome research*, vol. 9, no. 11, pp. 1135–1142, 1999.

[11] B. Dost, C. Wu, A. Su, and V. Bafna, "Tclust: A fast method for clustering genome-scale expression data," *IEEE/ACM Transactions on Computational Biology and Bioinformatics (TCBB)*, vol. 8, no. 3, pp. 808–818, 2011.

[12] P. Novák, P. Neumann, and J. Macas, "Graph-based clustering and characterization of repetitive sequences in next-generation sequencing data," *BMC bioinformatics*, vol. 11, no. 1, p. 378, 2010.

[13] M. E. Newman, "Fast algorithm for detecting community structure in networks," *Physical review E*, vol. 69, no. 6, p. 066133, 2004.

[14] G. Palla, A.-L. Barabási, and T. Vicsek, "Quantifying social group evolution," *arXiv preprint arXiv:0704.0744*, 2007.

[15] B. Adamcsek, G. Palla, I. J. Farkas, I. Derényi, and T. Vicsek, "Cfinder: locating cliques and overlapping modules in biological networks," *Bioinformatics*, vol. 22, no. 8, pp. 1021–1023, 2006.

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

[17] 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*, vol. 33, no. 6, pp. 623–630, 2015.

[18] H. Li, "Minimap and miniasm: fast mapping and de novo assembly for noisy long sequences," *Bioinformatics*, vol. 32, no. 14, pp. 2103–2110, 2016.

[19] N. J. Loman and A. R. Quinlan, "Poretools: a toolkit for analyzing nanopore sequence data," *Bioinformatics*, vol. 30, no. 23, pp. 3399–3401, 2014.

[20] W. J. Kent, "Blatthe blast-like alignment tool," *Genome research*, vol. 12, no. 4, pp. 656–664, 2002.

[21] R. Mott, "Est_genome: a program to align spliced dna sequences to unspliced genomic dna," *Bioinformatics*, vol. 13, no. 4, pp. 477–478, 1997.

[22] J. K. Senior, "Partitions and their representative graphs," *American Journal of Mathematics*, vol. 73, no. 3, pp. 663–689, 1951.

[23] Y. Wang, H. C. Leung, S. Yiu, and F. Y. Chin, "Metacluster 5.0: a two-round binning approach for metagenomic data for low-abundance species in a noisy sample," *Bioinformatics*, vol. 28, no. 18, pp. i356–i362, 2012. [Online]. Available: +http://dx.doi.org/10.1093/bioinformatics/bts397

[24] D. R. Garalde, E. A. Snell, D. Jachimowicz, A. J. Heron, M. Bruce, J. Lloyd, A. Warland, N. Pantic, T. Admassu, J. Ciccone, S. Serra, J. Keenan, S. Martin, L. McNeill, J. Wallace, L. Jayasinghe, C. Wright, J. Blasco, B. Sipos, S. Young, S. Juul, J. Clarke, and D. J. Turner, "Highly parallel direct rna sequencing on an array of nanopores," *bioRxiv*, 2016. [Online]. Available: http://www.biorxiv.org/content/early/2016/08/12/068809