De novo clustering of extensive long-read transcriptome datasets with isONclust3

Alexander J. Petri[1,*], Kristoffer Sahlin[1,*]

Department of Mathematics, Science for Life Laboratory, Stockholm University, Stockholm, Sweden
[*]alexander.petri@math.su.se, ksahlin@math.su.se

Abstract. Long-read sequencing techniques can sequence transcripts from end to end, greatly improving our ability to study the transcription process and enabling more detailed analysis of diseases such as cancer. While several well-established tools exist for long-read transcriptome analysis, most are reference-based and, therefore, limited by the reference genome. This prevents analysis of organisms without high-quality reference genomes and samples or genes with high variability (e.g., cancer samples or some gene families) from being analyzed to their full potential. In such settings, analysis using a reference-free method is favorable. The computational problem of clustering long reads by region of common origin is well-established for reference-free transcriptome analysis pipelines. Such clustering enables large datasets to be split up roughly by gene family and, therefore, an independent analysis of each cluster. There exist tools for this. However, none of those tools can efficiently process the large amount of reads that are now generated by long-read sequencing technologies.

We present isONclust3, an improved algorithm over isONclust and isONclust2, to cluster massive long-read transcriptome datasets at the gene family level. Like isONclust, IsONclust3 represents each cluster with a set of minimizers. However, unlike other approaches, isONclust3 dynamically updates the cluster representation during clustering by adding high-confidence minimizers from new reads assigned to the cluster. We show that isONclust3 yields results with higher or comparable quality to state-of-the-art algorithms but is 10-100 times faster on large datasets. Also, using a 256Gb computing node, isONclust3 was the only tool that could cluster 37 million PacBio reads, which is a typical throughput of the recent PacBio Revio sequencing machine.

1 Introduction

Long-read transcript sequencing using PacBio and Oxford Nanopore Technologies (ONT) has been key in investigating complex isoform landscapes across various organisms. There exist several genome and annotation-based tools capable of analyzing transcriptomic long-read sequencing data [24,28,11,39], but those tools are limited to well-sequenced and annotated organisms, and can, at best, be used with references from closely related organisms. Additionally, such tools do not work well for more specialized analysis of, e.g., gene families not well represented in current reference genomes [36] or cancer samples with high diversity to the reference. This has created a need for de novo transcriptome analysis. The generally established workflow of de novo transcriptome analysis is to cluster the reads into transcripts, genes, or gene families, then perform read error correction (if needed), and finally reconstruct transcripts present in the individual clusters [2,34,35,26]. The most resource-intensive step of the three is clustering, as the two downstream steps can work with sub-instances of the data (the clusters). However, the data yields for long-read sequencing techniques have been greatly enhanced over the last couple of years [20]. The order of magnitude increase in data poses scaling challenges to existing state-of-the-art algorithms. While several different algorithms exist for the de novo clustering of generic protein [15,25,37,38] and nucleotide sequences [15,4,10,6], transcripts have distinguished characteristics such as isoforms stemming from the same gene but having different splice patterns that set this problem apart from the other sequence clustering problems. Additionally, long-read transcriptome sequencing data pose challenges such as 3' and 5' transcript and sequencing variability, RNA degradation, and variable error rates between and within the reads.

1.1 Previous work on clustering of long-read transcriptome data

Several *de novo*-based clustering methods have been proposed (ToFU [7], IsoCon [36], CARNAC-LR [18], isONclust [34], RATTLE [2], GeLuster [16]). However, early tools such as ToFU and IsoCon [36] were initial tools designed only for early PacBio datasets containing in the order of thousands to tens of thousands of reads. CARNAC-LR, RATTLE, GeLuster, isONclust were introduced and shown to handle hundreds of thousands up to a couple of million of reads. The CARNAC-LR algorithm creates a similarity graph using an all-vs-all read mapper such as minimap2 [14] and derives an upper bound for a number of clusters in the graph. In the next step boundaries for initial clusters are refined to fulfill a partitioning condition to yield the final clustering. In CARNAC-LR, all-vs-all mapping is a bottleneck and the support for CARNAC-LR has been discontinued [17]. We have previously introduced isONclust [34], which is based on minimizers. IsONclust is a greedy algorithm that sorts the reads in decreasing order by a combination of their quality and length. In the second step, reads are iterated over in the sorted order and the first read forms a cluster and subsequently all reads are compared to existing clusters, using minimizers [29]. If a read has high similarity with an existing cluster, the read is added to the cluster, if not, the read forms a new cluster. Each cluster is represented by the minimizers of the read that initialized the cluster. IsONclust has widely been used over recent years both for transcriptome [12,41,42,40] and amplicon sequencing analysis [33,27]. However, it is not able to cluster reads transcribed in different orientations, and the algorithm is, by today's standards, slow and memory intense. A more integral issue with the algorithm is that isONclust uses only the minimizers from the initial read forming a new cluster as the cluster's representation. Therefore, the cluster representation is static and this prevents reads from isoforms with larger difference in exon structures to be placed in the same cluster. ONT, in collaboration with us, reimplemented the isONclust algorithm in the C++ programming language to improve its runtime and added some optional features naming it isONclust2 [9]. IsONclust2 guarantees the same output as isONclust, but has in independent evaluation demonstrated about 3.5-4 times speedup over isONclust [31]. IsONclust2 was, however, written with a complex interface that requires it to be run within a snakemake wokflow, hindering usability of the tool. In addition, the tool has been discontinued and is no longer supported [9]. The RATTLE algorithm works similarly to isONclust in a greedy fashion, but with the differences that reads are sorted only by length in decreasing order in the first step. In the clustering step, RATTLE uses all k-mers to compare reads to existing clusters. Both isONclust and RATTLE have scalability issues, as we demonstrate. GeLuster is a recently published algorithm. In a first step, it selects a subset of the reads with a minimum length requirement (called pseudo-references), sharing low identity with each other by iterating through the reads in sorted order. In a second step, it aligns the remaining reads to the pseudo-references to form pre-clusters. The reads that have not been aligned properly to any

of the pseudo-references will be passed as the input to the next iteration where the length requirement for being a pseudo-reference is reduced. This algorithm is performed iteratively with 3 iterations set as default. It has shown to improve the clustering while requiring less computational resources than isONclust, and RATTLE [16]. However, none of these algorithms are able to scale to datasets with tens to hundreds of millions of reads, as is common for the latest PacBio Revio or ONT PromethION devices. For example, isONclust is a core component in de novo transcriptomics pipelines used by companies such as biobam [1]. Biobam was unable to process such large datasets with any of the algorithms (personal communication with Biobam), which we also demonstrate in this study using a PacBio Revio dataset. This prompted further development of algorithms to cluster currently generated long-read transcriptome sequencing datasets into gene families.

1.2 Our contribution

We developed isONclust3, an algorithm designed to tackle the scalability and accuracy shortcomings of isONclust. Similarly to isONclust, isONclust3 is a greedy algorithm and uses a minimizer-based approach to estimate similarity and to represent clusters. However, isONclust3 addresses the accuracy, time, and memory limitations with isONclust and other algorithms mentioned above. Unlike isONclust and other approaches, isONclust3 updates the cluster representations during clustering by adding *high-confidence minimizers* from new reads assigned to the cluster. High-confidence minimizers are inferred from the quality values of the base calls in the minimizer. Such cluster updating is needed to accommodate new information, from e.g., additional exons, SNP variation from alleles, intron retentions, or other transcriptomic variability belonging to the gene families. The dynamic updating with high-confidence minimizers enables isONclust3 to cluster more reads from the same gene family, while keeping the number of minimizers stored for each cluster informative in comparison to adding all minimizers of a read (as for isONclust) when clustering reads from error prone sequencing technologies. Also, genes can express transcripts in both directions on the genome. While isONclust only implements non-canonical minimizers, isONclust3 implements both non-canonical and canonical minimizers as seeds, to tailor for the scenario to keep these in separate and the same cluster, respectively. Finally, isONclust3 is implemented in Rust compared to, e.g., isONclust which is implemented in Python. We tested isONclust3 on ONT and PacBio data from different organisms as well as on simulated data. IsONclust3 shows slightly higher or comparable clustering quality to other tools. Most importantly, we demonstrate that isONclust3 is in many cases 1-2 orders of magnitude faster than other tools with comparable or smaller memory footprint and is the only tool that can cluster data from the PacBio Revio sequencing machine using a 256Gb computing node.

2 Methods

2.1 Preliminaries

Canonical minimizers Let $r \in R$ denote a string consisting of letters in $\Sigma = \{A, C, G, T\}$, that represents a sequencing *read*. We use zero-indexed notation for strings and let $r[i, i+k)$ denote a substring (*k-mer*) on r, covering positions from i to $i+k-1$. Let a hash function h induce a *total ordering* over the k-mers. Given two integers k and w with $1 \leq k \leq |r|$ and $1 \leq w \leq |r| - k + 1$, the (k, w)-minimizer [29] m in an interval $r[i, i+w)$, referred to as a *window*, is the smallest k-mer starting in the window. Repeating k-mers within a window are resolved by choosing the leftmost k-mer. We refer to a (k, w)-minimizer as a minimizer when k and w are not important for context.

A canonical minimizer is a minimizer formed from taking the smallest k-mer among both the forward and the reverse complement k-mers in a substring $s \geq k$ of r, where $s = r[i, i+w+k-1]$. We will refer to minimizers and canonical minimizers as seeds, as our approach work with any k-mer construct used for similarity search such as syncmers [3], or when using all k-mers.

High-confidence seeds Base call qualities are measured for each base by the Phred quality score [5] (Q score) that can be transformed to a probability q_i that the base is correctly called through $p_i = 1 - 10^{-q_i/10}$, where $p_i \in [0, 1]$ of the base located at position i in r. We compute the confidence of a seed spanning positions

$r[i, i+k)$ as $P_i = \prod_{j=i}^{i+k} p_j$. The formula computes the probability (as estimated by quality values) of the k-mer being correctly base-called in all of its nucleotides. If Q_i is greater than a threshold T_1, we denote the seed as a *high-confidence seed* (HCS). We set T_1 depending on the sequencing technology (analysis in section 3.3).

Algorithm 1: isONclust3

```
input  : Reads R, k, w, thresholds T1, T2
output: Read clusters C
Function SortReads(R, k, w, T1)
    for r ∈ R do
        seeds = generateSeeds(r.sequence, k, w)
        hcs = findHCS(seeds, T1)
    R_sort ← R.sort(hcs)
    return R_sort
Function Clustering(R_sort, T1, T2)
    Initialize hash table C (clusterID → readIDs).
    Initialize hash table S (seed → clusterIDs).
    c_id = 0
    for r ∈ R_sort do
        seeds = generateSeeds(r.sequence, k, w)
        if C is empty then
            C.insert(c_id, r.acc)
            hcs = findHCS(seeds, T1)
            S.addNewHCS(hcs)
            c_id += 1
        else
            Initialize hash table H (clusterID → seed count)
            for s ∈ seeds do
                H ← SeedCount(s, S, T2)
            for c ∈ clusters do
                bestCluster ← compareReadToCluster(r, c.representative, T2)
            A, c' = max(H.counts, H.clusterID )
            hcs = findHCS(seeds, T1)
            if A/|r.seeds| > T2 then
                C[c'].add(r.acc)
                S.addNewHCS(hcs, c')
            else
                C.insert(c_id, r.acc)
                S.addNewHCS(hcs, c_id)
                c_id += 1
    return C
R_sort = SortReads(R, k, w, T1)
C = Clustering(R_sort, T1, T2)
```

2.2 Algorithm

Sorting For each read we generate canonical minimizers and their associated qualities Q. We then sort the reads in decreasing order with respect to the number of HCS. This is in contrast to isONclust, which sorts according to the expected number of error-free k-mers in a read r and is not directly correlated to the amount of informative seeds used in the clustering. IsONclust3's read ordering is therefore directly linked to

the amount of useful information a read carries for the clustering. The sorted reads are stored in a file on disk.

Clustering In the second step, the sorted reads are streamed over. The first read initializes a cluster and the HCS of the read are stored as a representation of the cluster. Consecutive reads are then compared against existing clusters in the following manner. If a read does not share a sufficient fraction (threshold T_2) of (all) seeds from the read with the HCS of any existing cluster, it forms a new cluster. Only the HCS of the read kept as the cluster representation. Otherwise, if a read shares enough seeds with the HCS of at least one cluster, it is added to the cluster with the most shared seeds, and any HCS of the read not already existing in the cluster are added to the HCS of the cluster. For the clustering, isONclust3 stores a hash table that associates seeds as keys with arrays of cluster identifiers. For a visualization of the clustering approach, see Suppl. Fig. 1. The pseudocode of the algorithm is presented in Algorithm 1. The on-the-fly updated cluster representations accommodate HCS stemming from additional exons, SNV variations, or other transcriptomic variability. This is in contrast to isONclust, which only uses the minimizers of the initial read as the cluster representation.

2.3 Time complexity

Our algorithm is a greedy heuristic and therefore it is challenging to derive an informative worst-case runtime. We attempt to give a worst case analysis by parameterizing our analysis. Treating the function for hashing k-mers h as a constant and letting l be the length of the longest read, then producing minimizers has time complexity $O(hln)$ for n reads while filtering of minimizers to yield HCS takes at most $O(kln)$. The sorting step is $O(n \log n)$ for n reads (we use timsort). As for the clustering, the worst case scenario is that all reads form new clusters and the fraction of shared seeds with the HCS of existing clusters is high but lower than T_2. If we let $T_2 > 1$ (forcing no clustering) with a set of identical reads where all seeds ($< l$) are shared, the clustering has time complexity $O(n^2 l)$, since each read needs at most l seed look-ups and updates at most n cluster identifier counters for each seed. The worst-case runtime is therefore $O(hln+kln+n \log n+n^2 l))$. This is dominated by the worst time clustering complexity of $O(n^2 l)$. However, in practice there are substantially less clusters than reads, and seeds typically only match a small subset of clusters.

2.4 Input, output, and implementation details

The input of our algorithm consists of long transcriptomic reads that have been generated by either PacBio or ONT sequencing and were pre-processed via a barcode removal tool, such as Pychopper (for ONT data) and Lima (for PacBio data) removing artificial concatamers and non-full-length reads. Similarly to isONclust, isONclust3 outputs a tab separated file, which indicates the mapping of each read to a cluster, as well as separate *fastq* files containing the reads per cluster.

In our benchmarks we set $k = 15$ and $w = 51$ for PacBio and $k = 13$ and $w = 21$ for ONT, which is similar to isONclust ($k = 15, w = 50$ for PacBio and $k = 13, w = 20$ for ONT). We had to increase the value of w by 1 due to limitations (no even values of w allowed) with the minimizer implementation we use. The isONclust3 algorithm is written in the Rust programming language and is available via `https://github.com/aljpetri/isONclust3`. We use rust-bio [13] to parse the reads and the minimizer-iter library [19] for the generation of minimizers and canonical minimizers. We used rustc-hash, as the hash function for minimizers. IsONclust3 is not parallellized like other algorithms such as isONclust. A similar parallelization strategy to isONlclust could be implemented, although the clustering is not guaranteed to be the same when using different amounts of cores.

3 Results

3.1 Tools and datasets

We benchmarked isONclust3 against the existing state-of-the-art whole transcriptome clustering methods, namely isONclust, GeLuster, and RATTLE, on biological, synthetic, and simulated datasets (Table 1). We were not able to run isONclust2. We benchmarked the algorithms on four biological datasets consisting

of two PacBio Human datasets (ALZ, HG002) and ONT datasets from Drosophila (Droso) and human (ONT_human). We also investigated the algorithms' results on a SIRV dataset with real ONT sequencing reads generated from a pool of known transcript products, and for a simulated dataset used in the isONclust study [34]. See section S1 of the supplementary materials for data availability.

	Reads			Classes (Estimated ground truth)		
Dataset	Error rate (%)	#reads	Unaligned (%)	NS	S	% reads in NS classes
SIRV	6.9	1,409,565	1.1	7	0	98.8
Droso	7.0	3,764,978	13.0	5,488	2,153	86.9
ONT_human	4.5 *	1,473,632	27.3	13,831	15,300	71.6
SIM	1.9	500,000	4.0	14,792	2,152	99.6
ALZ	< 1.0	4,277,293	0.1	17,947	12,573	99.5
HG002	< 1.0	37,200,651	0.0	27,055	45,198	99.9

Table 1. Overview over the datasets used for the evaluation: NS: non-singleton clusters (clusters containing more than one read), S: singleton clusters (clusters containing a single read), *: for the ONT_human dataset we estimated the average error rate using the Phred score $Q = 13.5$ ([22]). We then calculated the respective error rate according to the Phred formula $P = 10^{\frac{-Q}{10}}$.

3.2 Evaluation metrics

For the clustering quality assessment of the data we use a clustering based on alignments to the reference genome via minimap2 [14] as a proxy, similarly to previous studies [34,16]. The *class* (ground truth estimation) of a read is the cluster the read was assigned to by the alignment. We denote reads as *unclassified* if the read could not be aligned. While the usage of our proxy of the true clustering may contain misclassifications through misalignments or sequencing artifacts in reads, e.g., due to reverse transcription errors, it is generally agreed upon as benchmarking approach for such methods and was used in previous studies [34,16]. We deem it a reasonable way to assess the relative performance of clustering tools. Table 1 shows an overview of the alignment characteristics and classes of the datasets. We refer to clusters and classes as non-singleton (NS) if they contain more than one read and singleton (S) it they consist of only one read. The class classification serves as the truth when we compute the clustering accuracy of the algorithms.

As in [34,16], we employed the completeness c, homogeneity h and the V-measure [30], as well as the Adjusted Rand (ARI) statistics to assess the clustering quality of the algorithms. A clustering is homogeneous if all of its clusters contain only data points from a single class. A clustering is complete if, for all classes, all the data points from a class are assigned to a single cluster. Intuitively, homogeneity penalizes wrongly clustering together reads from different classes, or over-clustering, while completeness penalizes reads that belong to the same class ending up in different clusters, or under-clustering. Specifically, let X be an array of n ordered reads where the elements in the array are the cluster identities (c_{id}) as appointed by the clustering algorithm. We define Y in a similar manner for the ground truth class identities of the reads. The homogeneity measure is then calculated as $h = 1 - H(Y|X)/H(Y)$. Here $H(*)$ and $H(*|*)$ refer to the entropy and conditional entropy functions as described in [30]. The completeness measure is given by $c = 1 - H(X|Y)/H(X)$. The V-measure [30] is computed as the harmonic mean of h and c, and therefore combines the two measures in a similar way as precision and recall are combined into the F-measure for a classification problem. We also compute the adjusted rand index measure (ARI) [8] to avoid bias with respect to a single clustering measure. Rand index (RI) as a measure of the percentage of correct pairings of elements. That is, a pairing (a, b) of elements a and b is correct if it agrees between the clustering and the classes, regardless if a and b belong to the same class or not, through the formula:

$$RI = \frac{TP + TN}{TP + FP + TN + FN}$$

Where TP, TN, FP, and FN stands for True Positives, True Negatives, False Positives, and False Negatives, respectively. Depending on the nature of correct classes (e.g., the extremes of all singleton classes or all elements in one class), the expected similarity of a random clustering differs. The ARI is the RI but corrected for chance by subtracting the expected similarity of all pair-wise comparisons between clusterings specified by a random model. While the RI ranges $\in [0, 1]$, the ARI can take values $\in [-1, 1]$, where random labelings get an ARI close to 0. The ARI is calculated as follows:

$$ARI(\pi_a, \pi_b) = \frac{\sum_{i=1}^{k_a}\sum_{j=1}^{k_b}\binom{n_{ij}}{2} - t_3}{\frac{1}{2}(t_1+t_2) - t_3}, \text{ with } t_1 = \sum_{i=1}^{k_a}\binom{n_{ia}}{2},\ t_2 = \sum_{j=1}^{k_b}\binom{n_{bj}}{2} \text{ and } t_3 = \frac{2t_1t_2}{n(n-1)}.$$

Here π_a and π_b are two clusterings on dataset with k_a and k_b clusters, having n samples; n_ij indicates the number of common objects in cluster c_i in clustering π_a and in cluster c_j in clustering π_b, n_{ia} denotes the number of objects in cluster c_i and n_{bj} represents the number of objects in cluster c_j in clustering π_b.

3.3 Deciding values for thresholds T_1 and T_2

We first performed an experiment to assess suitable values of T_1 and T_2 for PacBio and ONT by analyzing one ONT (Droso) and one PacBio (ALZ) dataset with respect to the clustering measures. Following the results from the threshold analysis (Fig. 1 top row panels) we set T_1 to 0.95 and 0.98 for ONT and PacBio data, respectively. Following the T_2 threshold experiment (Fig. 1, bottom row panels), we set T_2 to 0.5 for both ONT and PacBio data.

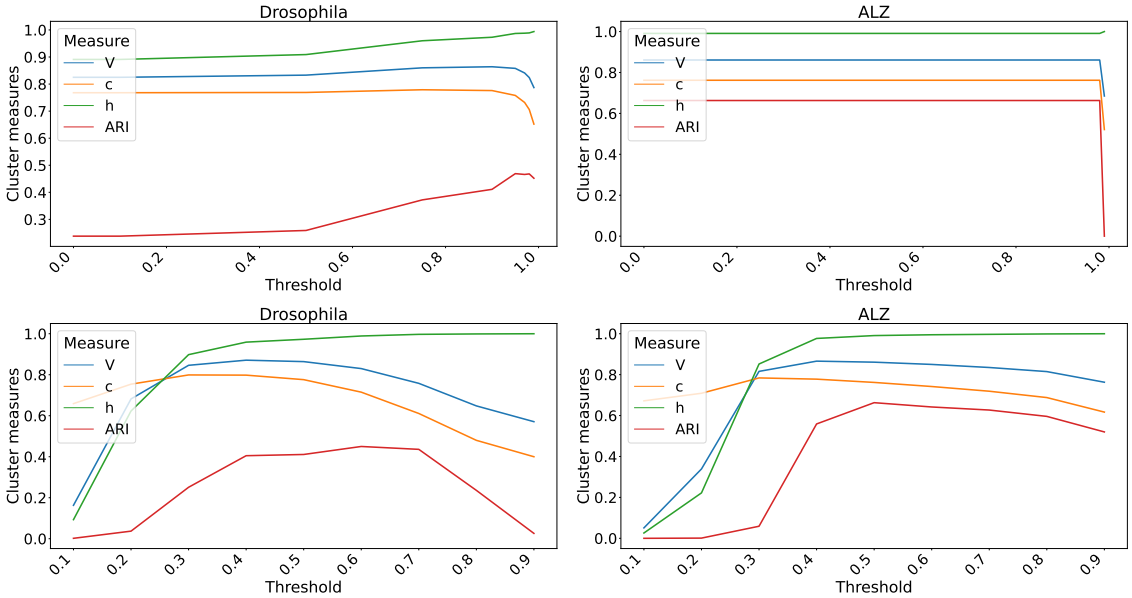

Fig. 1. Threshold analysis for T_1 (top row) and T_2 (bottom row) on the ONT Droso (left panels) and a Pacbio ALZ (right panels) datasets. The x-axis indicates the value of T_1 or T_2, while the y-axis shows the resulting V, c, h and ARI values. For T_1, we searched in $[0, 0, 0.1, 0.5, 0.75, 0.9, 0.95, 0.97, 0.98, 0.99]$. For T_2 we searched in $[0, 1, 0.2, \ldots, 0.9]$.

3.4 Benchmarking details

We ran isONclust3 with one core and all other algorithms with one and eight cores. For some algorithms such as isONclust, both clustering quality and memory usage will change depending on the number of cores. We ran isONclust3 both with and without canonical minimizers, denoted isONclust3 and isONclust3_nc) respectively, to illustrate the impact on datasets where a gene expresses transcripts from both directions.

We assessed the runtime performance of all tools measuring the wall-clock time using */usr/bin/time -v*. To minimize the impact of file access times on the overall run times we let all algorithms write their outputs onto fast access scratch memory space, a space available individually for each compute node. The slurm command *finishedjobinfo -j* was used to determine the RAM usage of the algorithms. We used a CentOS 7, with two 10-core Intel Xeon V4 CPUs each. The analyses were run on the default nodes having 128GB of memory and high-RAM nodes, each having 256GB of RAM available, when needed.

3.5 Clustering quality evaluation

The clustering quality results for the six datasets with all tools using one core are shown in Figure 2. Overall, isONclust3 fares comparable or favourable to the other tools. While the V, c and h measures show similar results for isONclust3 and GeLuster, the ARI for isONclust3 is consistently higher for all the datasets. For our largest dataset, HG002, which was produced with the new PacBio Revio system, none of the algorithms except isONclust3 was able to be run on our server due to too high RAM usage ($>$ 256Gb). The isONclust3 and GeLuster algorithms show consistently better results than isONclust and RATTLE.

IsONclust3_nc yields results close to the canonical version except for the SIRV and SIM datasets, which contain genes expressed in both directions. Notably, isONclust3_nc shows better results than isONclust (which uses non-canonical seeds). Together with the fact that we use similar parameter values of k and w in isONclust3, it indicates the improvements in isONclust3 over isONclust comes from the algorithmic differences (usage of HCS and dynamic updating) rather than through only implementing canonical seeds (see, e.g., SIRV and Droso). For the ONT_human dataset, isONclust only was able to cluster 12 percent of the reads with a perfect homogeneity but a low completeness and an ARI-measure of 0 for both one and eight cores. We did not observe significant differences in clustering quality when the other tools were run with 8 cores instead of one core (Suppl. Fig. 2).

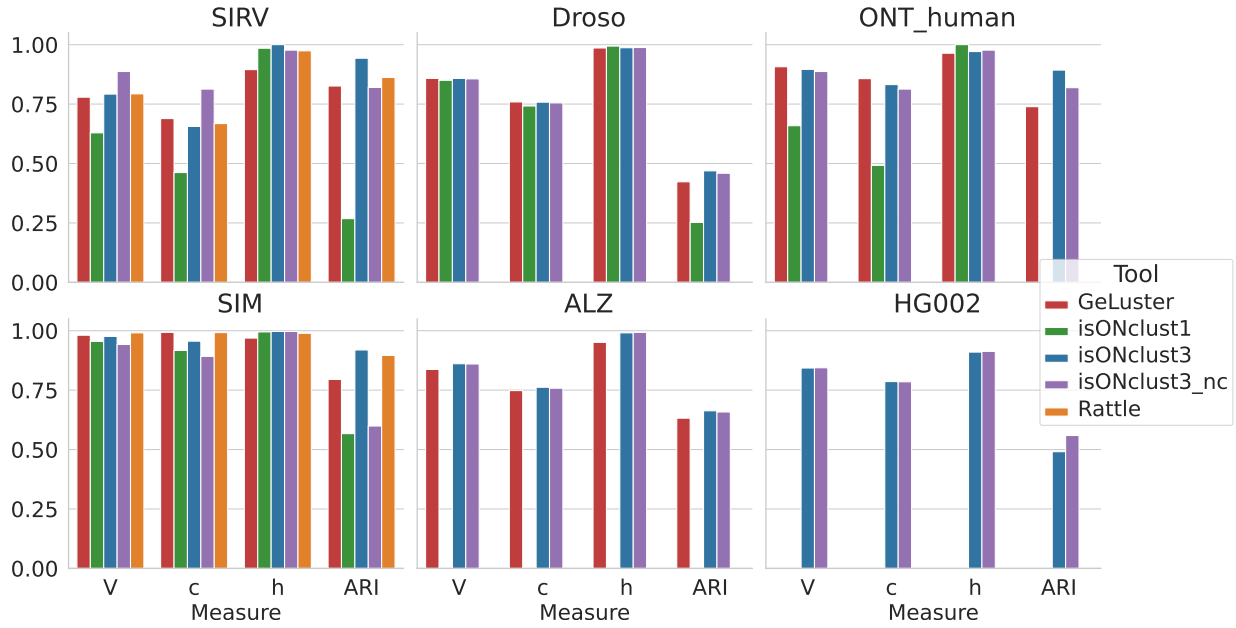

Fig. 2. Cluster quality measures. All algorithms were run with one core. For the ALZ dataset isONclust and RATTLE did not finish during the given run time limit (120 hours). For the HG002 dataset no algorithm except isONclust3 was able to be run due to their RAM usages exceeding the cluster RAM usage limit of 256GB.

3.6 Runtime analysis and memory usage assessment

Figure 3 shows runtimes and peak memory usages of all algorithms when using one core. Missing datapoints indicate that the algorithm either ran out of memory (256Gb) or took more than than the given time limit of five days (120h). IsONclust3 consistently has the lowest runtime and is 10 to 100 times faster than isONclust, RATTLE, and GeLuster on individual datasets. We also ran the other algorithms with eight cores and included isONclust3 with one core as a comparison (Suppl. Fig. 2). When the other algorithms are run with eight cores, isONclust3 is still 5 to 20 times faster for larger, challenging datasets such as ALZ. Notably, isONclust3 is the only algorithm that can cluster the HG002 dataset on our system due to the high memory usages of other algorithms, regardless of the number of cores used. For the ALZ dataset the RATTLE algorithm exceeded the time limit. IsONclust was able to complete clustering of ALZ in the time limit only when run with eight cores. The GeLuster algorithm showed a high runtime for the ALZ dataset that we did not observe for the isONclust3 algorithm. In terms of memory usage, isONclust3 used slightly more RAM for the analysis of the SIM dataset with respect to GeLuster, for all other datasets, the RAM usage was lower.

4 Discussion

We introduced the isONclust3 algorithm for clustering massive transcriptomic long-read sequencing datasets. IsONclust3 uses minimizer seeds to compare reads to clusters and dynamically updates representations of clusters with high-confidence minimizer seeds instead of choosing seeds from one read as the representative of a cluster (isONclust). Additionally, the isONclust3 algorithm was implemented in the high-performance programming language Rust also facilitating simple installation and usage of the tool (unlike the Python implementation of isONclust and the prototype isONclust2). We showed that isONclust3 performs favourable to existing tools in terms of cluster measures on a diverse set of datasets.

IsONclust3 uses canonical minimizers as default seeds. However, to enable a better head-to-head comparison to isONclust, we also ran isONclust3 with noncanonical minimizers, showing the algorithmic improvements, beyond canonical seeds, over isONclust. IsONclust3 also outperforms all other tools in memory usage and runtime on most datasets, and is the only tool that can cluster PacBio Revio sequencing datasets with ten times more reads to older sequencing machines. IsONclust3 replaces isONclust, and is applicable to any use case that isONclust has been utilized for.

4.1 Future work

While isONclust3 generally outperforms existing algorithms with respect to runtime and RAM usage, it is not yet parallelized. If parallelized in the same manner as isONclust, it could improve the runtime of isONclust3 further with a trade-off in RAM usage (depending on how many cores are used). The peak RAM usage in isONclust3 occurs in the sorting step when all reads and their HCS exist in RAM. It is possible to use, e.g., external merge sort to reduce the peak memory. We may also consider using bit-encoding of two bits per nucleotide to represent the reads, and possibly discretizing the Phred quality values further to fit in less bits. As for the methodology aspects, strobemers [32] have been used successfully in de novo transcriptome analysis tools to classify transcripts [23] based on a set of overlapping strobemers. It is possible that isONclust3 could implement such an approach to compare and cluster reads on transcript-level for applications that require it, or to further increase processing speed through uniqueness of the seeds.

5 Conclusion

We presented isONclust3, an algorithm for clustering long-read transcriptomic data by gene families. Using simulated, synthetic, and biological data we demonstrated that isONclust3 produces accurate clusterings and has comparable or higher V, c, h and ARI than other methods. IsONclust3 is also substantially faster and less memory consuming than other tools on most datasets and is the only algorithm that can cluster a PacBio Revio dataset with 37M reads. IsONclust3 replaces isONclust, and is suitable to cluster recent transcriptome long-read sequencing datasets with tens of millions of reads.

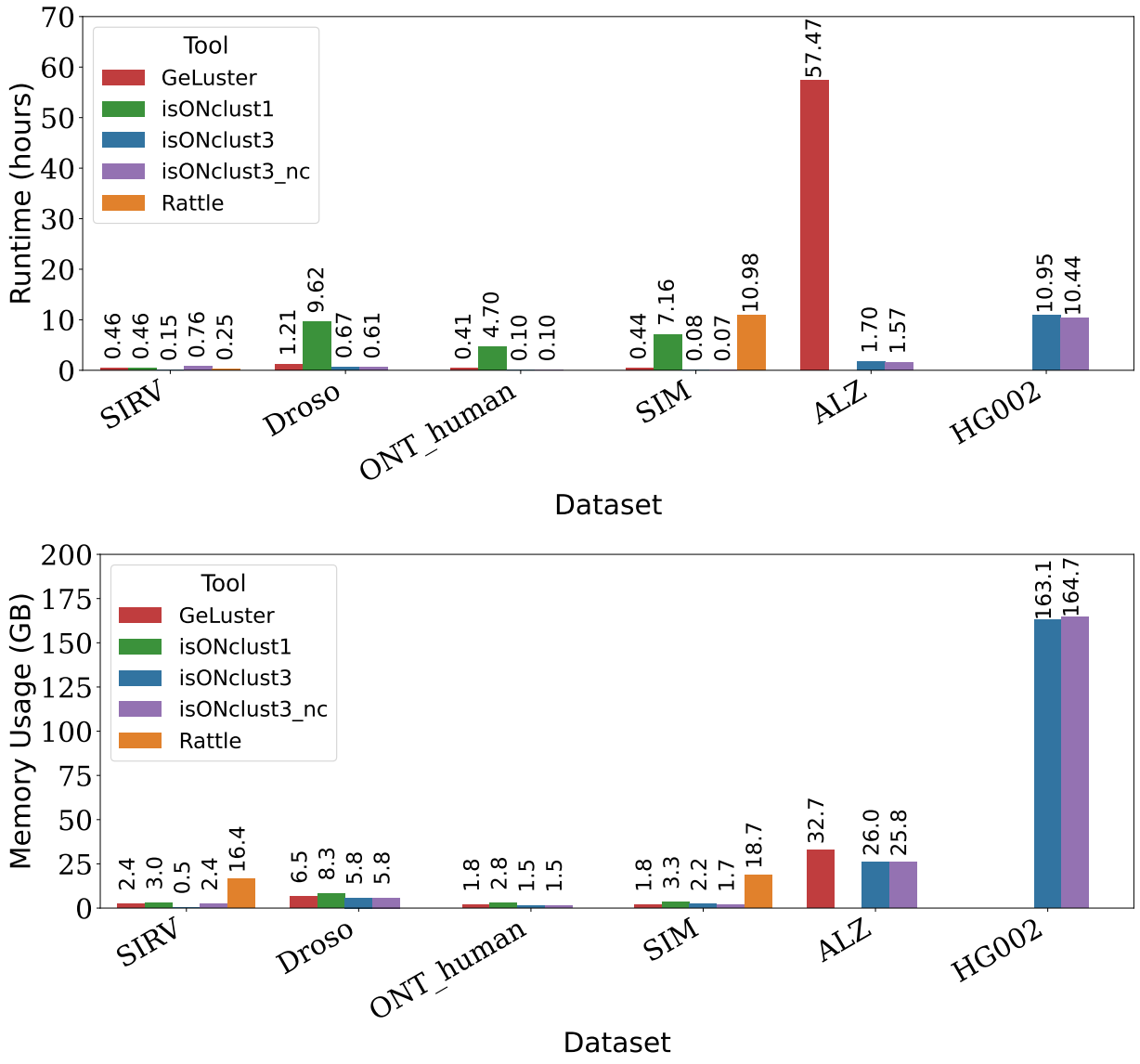

Fig. 3. Runtime and memory usage. Upper panel: runtime of each algorithm for each dataset in hours. Lower panel: memory usage in GB. x-axis: the different datasets and tools, y-axis: Runtime in hours/ RAM usage in GB. IsONclust3 was the only algorithm able to run the HG002 dataset on our machine.

Availability IsONclust3 is available at `https://github.com/aljpetri/isONclust3`. The analysis pipelines were written using snakemake [21]. The pipeline and intermediate results used for plotting are available via `https://github.com/aljpetri/isONclust_analysis`. The sources of the datasets can be found in the supplementary materials to this paper.

Acknowledgements Parts of the evaluation were enabled by resources in project snic2022-5-592 provided by the Swedish National Infrastructure for Computing (SNIC) at UPPMAX, partially funded by the Swedish Research Council through grant agreement no. 2018-05973.

Funding Kristoffer Sahlin was supported by the Swedish Research Council (SRC, Vetenskapsrådet) under Grant No. 2021-04000.

References

1. biobam. https://www.biobam.com/transcriptomics/, 2022.
2. Ivan de la Rubia, Akanksha Srivastava, Wenjing Xue, Joel A Indi, Silvia Carbonell-Sala, Julien Lagarde, M Mar Albà, and Eduardo Eyras. Rattle: reference-free reconstruction and quantification of transcriptomes from nanopore sequencing. *Genome Biology*, 23(1):153, 2022.
3. Robert Edgar. Syncmers are more sensitive than minimizers for selecting conserved k-mers in biological sequences. *PeerJ*, 9:e10805, 2021.
4. Robert C Edgar. Search and clustering orders of magnitude faster than blast. *Bioinformatics*, 26(19):2460–2461, 2010.
5. Brent Ewing, LaDeana Hillier, Michael C. Wendl, and Phil Green. Base-calling of automated sequencer traces usingphred. i. accuracy assessment. *Genome Research*, 8(3):175–185, 1998.
6. Mohammadreza Ghodsi, Bo Liu, and Mihai Pop. Dnaclust: accurate and efficient clustering of phylogenetic marker genes. *BMC bioinformatics*, 12:1–11, 2011.
7. Sean P Gordon, Elizabeth Tseng, Asaf Salamov, Jiwei Zhang, Xiandong Meng, Zhiying Zhao, Dongwan Kang, Jason Underwood, Igor V Grigoriev, Melania Figueroa, et al. Widespread polycistronic transcripts in fungi revealed by single-molecule mrna sequencing. *PloS one*, 10(7):e0132628, 2015.
8. Lawrence Hubert and Phipps Arabie. Comparing partitions. *Journal of classification*, 2:193–218, 1985.
9. isONclust2. https://github.com/nanoporetech/isonclust2, 2020.
10. Benjamin T James, Brian B Luczak, and Hani Z Girgis. Meshclust: an intelligent tool for clustering dna sequences. *Nucleic acids research*, 46(14):e83–e83, 2018.
11. Sam Kovaka, Aleksey V Zimin, Geo M Pertea, Roham Razaghi, Steven L Salzberg, and Mihaela Pertea. Transcriptome assembly from long-read rna-seq alignments with stringtie2. *Genome biology*, 20:1–13, 2019.
12. Minoru Kumazawa, Hiroyo Nishide, Ryo Nagao, Natsuko Inoue-Kashino, Jian-Ren Shen, Takeshi Nakano, Ikuo Uchiyama, Yasuhiro Kashino, and Kentaro Ifuku. Molecular phylogeny of fucoxanthin-chlorophyll a/c proteins from chaetoceros gracilis and lhcq/lhcf diversity. *Physiologia Plantarum*, 174(1):e13598, 2022.
13. Johannes Köster. Rust-Bio: a fast and safe bioinformatics library. *Bioinformatics*, 32(3):444–446, 10 2015.
14. Heng Li. Minimap2: pairwise alignment for nucleotide sequences. *Bioinformatics*, 34(18):3094–3100, 2018.
15. Weizhong Li and Adam Godzik. Cd-hit: a fast program for clustering and comparing large sets of protein or nucleotide sequences. *Bioinformatics*, 22(13):1658–1659, 2006.
16. Junchi Ma, Xiaoyu Zhao, Enfeng Qi, Renmin Han, Ting Yu, and Guojun Li. Highly efficient clustering of long-read transcriptomic data with geluster. *Bioinformatics*, 40(2):btae059, 2024.
17. Camille Marchet. https://github.com/kamimrcht/carnac-lr/issues/7#issuecomment-1087230658.
18. Camille Marchet, Lolita Lecompte, Corinne Da Silva, Corinne Cruaud, Jean-Marc Aury, Jacques Nicolas, and Pierre Peterlongo. Carnac-lr: Clustering coefficient-based acquisition of rna communities in long reads. In *JOBIM 2018-Journées Ouvertes Biologie, Informatique et Mathématiques*, pages 1–3, 2018.
19. Igor Martayan. Minimizer-iter: Iterate over minimizers of a dna sequence., year = 2024, url = https://github.com/rust-seq/minimizer-iter, urldate = 2024-10-14.
20. Vivien Marx. Method of the year: long-read sequencing. *Nature Methods*, 20(1):6–11, 2023.
21. Felix Mölder, Kim Philipp Jablonski, Brice Letcher, Michael B Hall, Christopher H Tomkins-Tinch, Vanessa Sochat, Jan Forster, Soohyun Lee, Sven O Twardziok, Alexander Kanitz, et al. Sustainable data analysis with snakemake. *F1000Research*, 10, 2021.
22. Ying Ni, Xudong Liu, Zemenu Mengistie Simeneh, Mengsu Yang, and Runsheng Li. Benchmarking of nanopore r10. 4 and r9. 4.1 flow cells in single-cell whole-genome amplification and whole-genome shotgun sequencing. *Computational and Structural Biotechnology Journal*, 21:2352–2364, 2023.

23. Ka Ming Nip, Saber Hafezqorani, Kristina K. Gagalova, Readman Chiu, Chen Yang, RenéL. Warren, and Inanc Birol. Reference-free assembly of long-read transcriptome sequencing data with rna-bloom2. *Nature Communications*, 14(1):2940, 2023.
24. Baraa Orabi, Ning Xie, Brian McConeghy, Xuesen Dong, Cedric Chauve, and Faraz Hach. Freddie: annotation-independent detection and discovery of transcriptomic alternative splicing isoforms using long-read sequencing. *Nucleic Acids Research*, 51(2):e11–e11, 2023.
25. Alberto Paccanaro, James A Casbon, and Mansoor AS Saqi. Spectral clustering of protein sequences. *Nucleic acids research*, 34(5):1571–1580, 2006.
26. Alexander J Petri and Kristoffer Sahlin. Isonform: reference-free transcriptome reconstruction from oxford nanopore data. *Bioinformatics*, 39(Supplement_1):i222–i231, 2023.
27. Aaron Pomerantz, Kristoffer Sahlin, Nina Vasiljevic, Adeline Seah, Marisa Lim, Emily Humble, Susan Kennedy, Henrik Krehenwinkel, Sven Winter, Rob Ogden, and Stefan Prost. Rapid in situ identification of biological specimens via dna amplicon sequencing using miniaturized laboratory equipment. *Nature Protocols*, 17(6):1415–1443, 2022.
28. Andrey D Prjibelski, Alla Mikheenko, Anoushka Joglekar, Alexander Smetanin, Julien Jarroux, Alla L Lapidus, and Hagen U Tilgner. Accurate isoform discovery with isoquant using long reads. *Nature Biotechnology*, 41(7):915–918, 2023.
29. Michael Roberts, Wayne Hayes, Brian R Hunt, Stephen M Mount, and James A Yorke. Reducing storage requirements for biological sequence comparison. *Bioinformatics*, 20(18):3363–3369, 2004.
30. Andrew Rosenberg and Julia Hirschberg. V-measure: A conditional entropy-based external cluster evaluation measure. In *Proceedings of the 2007 joint conference on empirical methods in natural language processing and computational natural language learning (EMNLP-CoNLL)*, pages 410–420, 2007.
31. Mélanie Sagniez, Anshul Budhraja, Bastien Paré, Shawn M. Simpson, Clément Vinet-Ouellette, Marieke Rozendaal, and Martin A. Smith. Assembly arena: Benchmarking rna isoform reconstruction algorithms for nanopore sequencing. *bioRxiv*, 2024.
32. Kristoffer Sahlin. Effective sequence similarity detection with strobemers. *Genome Research*, 31(11):2080–2094, 2021.
33. Kristoffer Sahlin, Marisa CW Lim, and Stefan Prost. NGSpeciesID: DNA barcode and amplicon consensus generation from long-read sequencing data. *Ecology and evolution*, 11(3):1392–1398, 2021.
34. Kristoffer Sahlin and Paul Medvedev. De novo clustering of long-read transcriptome data using a greedy, quality value-based algorithm. *Journal of Computational Biology*, 27(4):472–484, 2020. PMID: 32181688.
35. Kristoffer Sahlin and Paul Medvedev. Error correction enables use of oxford nanopore technology for reference-free transcriptome analysis. *Nature communications*, 12(1):2, 2021.
36. Kristoffer Sahlin, Marta Tomaszkiewicz, Kateryna D. Makova, and Paul Medvedev. Deciphering highly similar multigene family transcripts from iso-seq data with isocon. *Nature Communications*, 9(1):4601, 2018.
37. Martin Steinegger and Johannes Söding. Mmseqs2 enables sensitive protein sequence searching for the analysis of massive data sets. *Nature biotechnology*, 35(11):1026–1028, 2017.
38. Martin Steinegger and Johannes Söding. Clustering huge protein sequence sets in linear time. *Nature communications*, 9(1):2542, 2018.
39. Laura H Tung, Mingfu Shao, and Carl Kingsford. Quantifying the benefit offered by transcript assembly with scallop-lr on single-molecule long reads. *Genome biology*, 20:1–18, 2019.
40. Per Unneberg, Mårten Larsson, Anna Olsson, Ola Wallerman, Anna Petri, Ignas Bunikis, Olga Vinnere Pettersson, Chiara Papetti, Astthor Gislason, Henrik Glenner, Joan E. Cartes, Leocadio Blanco-Bercial, Elena Eriksen, Bettina Meyer, and Andreas Wallberg. Ecological genomics in the northern krill uncovers loci for local adaptation across ocean basins. *Nature Communications*, 15(1):6297, 2024.
41. Mrinalini Walter and Nalini Puniamoorthy. Discovering novel reproductive genes in a non-model fly using de novo gridion transcriptomics. *Frontiers in Genetics*, 13:1003771, 2022.
42. Karl Johan Westrin, Warren W. Kretzschmar, and Olof Emanuelsson. Clustrast: a short read de novo transcript isoform assembler guided by clustered contigs. *BMC Bioinformatics*, 25(1):54, 2024.