Weighted de novo clustering of third-generation transcriptomic datasets

Luca Denti[1,*,†], Yoshihiro Shibuya[2,*,†]

[1]*Department of Applied Informatics, Faculty of Mathematics, Physics and Informatics, Comenius University in Bratislava, Slovakia*
[2]*Pasteur Institute, 25-28 Rue du Dr Roux, 75015 Paris*

Abstract

The ability of third-generation sequencing technologies to sequence end-to-end transcripts is creating new opportunities to enhance our understanding of the transcriptomic landscape in eukariotic organisms. In this context, a common task is the *de novo* clustering of long transcriptomic reads, that is, split a long read sample into smaller samples (one per gene family) that can then be more easily analyzed. Solving this computational problem in a reference- and annotation-free fashion is of the utmost importance when the organism under investigation is not well studied and complete gene annotations are not available. To this end, we present `SolidClust`, an approach for the de novo clustering of long reads. `SolidClust` is heavily inspired by its predecessor, `isOnClust3`: both algorithms are greedy algorithms based on the notion of *high-confidence* minimizers. In this work, we extend upon this basis by introducing the notion of *solid high-confidence* minimizers. As demonstrated by our experimental evaluation of real datasets, `SolidClust` is able to achieve comparable or higher clustering accuracy w.r.t. `isOnClust3` while drastically reducing the memory requirements. `SolidClust` is freely available at https://github.com/ldenti/solidclust.

Keywords

Long reads, Transcriptomic, De novo clustering

1. Introduction

Third-generation sequencing technologies, such as Oxford Nanopore and Pacific Biosciences, are nowadays able to produce extremely long reads that can be used to improve the accuracy of many bioinformatics tasks and advance our understanding of several biological processes. Such technologies proved to be crucial in unveiling the full complexity of the *transcription* process, that is the biological mechanism by which a gene, i.e., a DNA locus, is transcribed in a transcript, i.e., a mRNA molecule, which is then translated into a protein. However, due to (alternative) splicing, a single gene can exhibit multiple splice variants (transcripts) and then code for multiple proteins, increasing the complexity of the transcriptomic landscape in eukaryotes. Analyzing such a mechanism is of the utmost importance, since it has been associated with cancer [1, 2], genetic diseases [3], and various biological processes, such as aging [4] and tissue differentiation [5].

Many approaches have been designed to analyze transcription and splicing using short-reads sequencing technologies [6, 7, 8, 9, 10, 11, 12], whereas long-reads can fully cover the transcripts [13], creating new opportunities to thoroughly analyze the transcriptomic landscape and enhance our understanding of it. Many reference- and annotation-based approaches have been proposed to analyze long-reads RNA samples [14, 15, 16, 17]. However, these approaches rely on the availability of accurate reference genomes and complete gene annotations and can be hardly applied to under-studied organisms. To overcome this limitation, reference- and annotation-free approaches have been proposed. These approaches do not rely on any prior knowledge and can thus work directly on input reads. In this context, a well-established computational problem is the *de novo* clustering of reads by gene locus (or gene family). In other words, the goal of this problem is to split a read sample into multiple smaller samples, each roughly referring to the same set of genes. By doing so, the analysis of a large sample

ITAT'25: Information Technologies – Applications and Theory, September 26–30, 2025, Telgárt, Slovakia
*Corresponding authors.
†These authors contributed equally.
denti1@uniba.sk (L. Denti); yoshihiro.shibuya@pasteur.fr (Y. Shibuya)

(e.g., transcript reconstruction) can be reduced to multiple independent analyses of smaller samples. Recent advancements in third-generation sequencing technologies have enhanced the quality of the reads, simplifying this clustering problem, but have also increased the sequencing throughput and the size of the generated read sample. These technologies are now capable of generating tens of millions of accurate reads in a single experiment [18]. This poses a scalability challenge. Among the several *de novo* approaches proposed in the literature [19, 20, 21, 22], only the latest `isOnClust3` [23] is capable of analyzing large long-read samples produced by third-generation sequencing technologies.

Heavily inspired by `isOnClust3`, we introduce here `SolidClust`, a new approach for the *de novo* clustering of third-generation transcriptomic datasets. Similarly to `isOnClust3`, `SolidClust` uses high-confidence minimizers to sort and cluster reads in a greedy fashion, adding a read to an already created cluster if they share a sufficient fraction of minimizers. Although `SolidClust` follows the same general greedy idea proposed in [23], it enhances its predecessor in several ways, here intuitively described. For more details on `SolidClust`, we refer the reader to Section 2. On the algorithmic side, `SolidClust` tags each minimizer as *solid* minimizer depending on how many times it has been seen globally among all clusters and how many times it has been included in the cluster currently being compared to. Only solid minimizers are considered when comparing a read to a cluster to decide if the read shares enough minimizer with the cluster. Indeed, if a minimizer has been seen several times among all created clusters and too few times in the cluster currently under investigation, it is considered an erroneus (non solid) minimizer and filtered out. This additional check allows `SolidClust` to cope with two situations that can arise when a minimizer is the product of sequencing errors: (i) when a minimizer is too frequent among most of the clusters and (ii) when a minimizer occurs many times in a single cluster but fewer times in the current cluster under investigation. The use of solid minimizers allows `SolidClust` to be more robust to minimizers that occur in the read by chance due to sequencing errors. A second algorithmic difference between `isOnClust3` and `SolidClust` is that the latter do not consider repetitions when extracting minimizers from a read. In such a way, `SolidClust` clustering is less impacted by minimizers that are too frequent, such as those produced by long stretches of the same nucleotide, which can be observed in the datasets produced by third generation sequencing technologies due to biological and technical factors. Finally, our new efficient `C` implementation reduces the memory requirement by more than 66% while maintaining very high clustering accuracy and clustering speed, as demonstrated by our experimental evaluation on real datasets, described in Section 3.

2. Methods

Our work is a modification of `isOnClust3` [23], but with extended functionality and better time/memory tradeoffs. For these reasons, and to keep the paper self-contained, we first report the main ideas behind `isOnClust3` in Section 2.2. We then introduce `SolidClust` in Section 2.3, highlighting the novelties of our approach compared to its predecessor.

2.1. Preliminaries

2.1.1. Minimizers

Let χ be a set of strings over alphabet $\Sigma = \{A, C, G, T\}$. We call k-mers the substrings of length k obtainable by sliding a window over strings in χ.

Minimizers are (left-most) minimal k-mers over windows of w consecutive k-mers for a given order $\mathcal{O}_k$ [24, 25]. Since the minimum k-mer can only change when a new minimum is found, or the old one goes out of the window, minimizers are an effective sampling scheme with a wide range of applications in bioinformatics [26, 27, 28]. In practice, a fast, non-cryptographic random hash function is used to define the order $\mathcal{O}_k$ (random minimizer schemes [29]).

In the following, we will use *canonical* minimizers by treating each k-mer and its reverse complement as the same object by taking the (lexicographic) minimum between the two.

2.1.2. High confidence seeds (HCS)

By reusing isOnClust3 terminology we define *High Confidence Seeds* (HCS) as minimizers with probability higher than a given threshold. A minimizer's probability can be easily computed as the product of all its bases probabilities, encoded as Phred quality scores (Q scores) in FASTQ files following the relation:

$$Q = -10\log_{10}(P) \tag{1}$$

Quality values are further encoded in FASTQ files by representing them as ASCII letters of code $Q + 33$. In the following we will use the terms seeds and minimizers interchangeably so that HCS can also be read as high confidence minimizers.

2.2. isOnClust3

isOnClust3 is a greedy algorithm designed for the de novo clustering of third-generation RNA sequencing datasets, such as those generated by Oxford Nanopore or PacBio platforms. These datasets are characterized by long read lengths and comparatively high error rates, presenting unique challenges for accurate sequence clustering. The algorithm employs a greedy strategy, which does not guarantee globally optimal results, but provides substantial gains in computational efficiency and scalability, making it suitable for large-scale transcriptome analyses. The overall workflow can be divided into three principal stages: read sketching, clustering, and an optional cluster merging phase intended to refine the final clusters.

2.2.1. Read sketching

In the initial stage, each read is transformed into a compact representation to facilitate efficient comparison. This is achieved by extracting a set of canonical minimizers, which are representative k-mers selected from overlapping windows across the read. To ensure strand consistency, the canonical form is defined as the lexicographically smaller of the forward sequence and its reverse complement. These minimizer sets are subsequently filtered according to base-level quality scores, typically represented as Phred scores [30]. The quality of a k-mer is computed as the product of the base-wise quality probabilities, enabling the removal of low-confidence k-mers that are more likely to arise from sequencing errors. Only k-mers whose quality exceeds a user-defined threshold q are retained. This filtering step reduces noise while preserving the most informative features of each read. The filtered reads are then sorted in descending order of sketch size, such that longer and higher-quality reads are prioritized during clustering. This ordering increases the likelihood that the earliest clusters will be seeded by representative, high-confidence reads. The processed reads are written to a temporary FASTQ file to allow sequential disk-based streaming in the subsequent stage.

2.2.2. Clustering

In the clustering stage, the sorted reads are processed sequentially to form clusters in a greedy fashion. The first read encountered becomes the seed for the initial cluster, with its High Confidence Seeds (HCS) – the quality-filtered minimizers – serving as the defining elements of that cluster. Each subsequent read is compared against all clusters generated thus far. The comparison is asymmetric: the read is represented by all of its minimizers, whereas each cluster is represented solely by its HCS. For each cluster, the fraction of shared seeds between the read and the cluster's HCS is calculated. If this fraction meets or exceeds a user-defined similarity threshold τ (T_2 in the original paper [23]), the read is assigned to that cluster, and its HCS are incorporated into the cluster's seed set. If no existing cluster meets the similarity criterion, a new cluster is initiated using the read's HCS. Because the clustering is performed in a single pass without re-evaluating earlier assignments, the method is computationally efficient, but sensitive to errors in the early stages of processing.

2.2.3. Cluster merging

The optional cluster merging stage is designed to address over-clustering that may arise from the greedy nature of the algorithm or from noise in the sequencing data. Clusters are processed in order of increasing size, ensuring that smaller clusters are evaluated first for potential merging into larger, more established clusters. For each candidate pair, the fraction of shared HCS is computed; if this proportion exceeds a merging threshold, the clusters are combined, and their reads are pooled into a single, consolidated cluster. This procedure is repeated iteratively until no further merges are possible. The merging step serves as a corrective measure, mitigating the impact of early misclassifications and producing a final clustering that more accurately reflects the true transcriptomic structure of the dataset.

2.3. SolidClust

We call our implemetation `SolidClust` since, like its predecessor, it follows the overall steps outlined in Section 2.2, but on the other hand, it significantly deviates in many important ways. The main difference is that our algorithm works on sketched sequences only, thus eliminating the need to load and sort the whole input file. Reads are sketched as sets of HCS, which are then sorted by (decreasing) size and stored on disk. Not having access to the original reads means we only compare the HCS of sketches to clusters. This is in contrast to the original implementation where all seeds of a read (including repetitions) are compared to the HCS contained in clusters. From our preliminary investigation of the data, considering repetitions produces an unexpected sorting of the reads, with reads containing long stretches of `A`s (or `T`s) being placed very high in the ordering (although most minimizers were simple stretches of a single nucleotide).

The other main difference is the way we compute similarities between reads and clusters. The original `isOnClust3` computes a simple Jaccard similarity between reads and clusters (albeit allowing multiple copies of the same minimizers in the reads), whereas `SolidClust` only allows certain seeds to contribute to the similarity. Our filtering strategy is based on the past history of the clustering process. At the time of adding read r from χ, let M and C be the sets of minimizers seen so far, and the set of clusters built. For each pair of (μ, c) with $\mu \in M$ and $c \in C$, we count how many times minimizer μ has been added to c. For ease of visualisation, one can imagine a matrix M of counters, with minimizers as rows, and clusters as columns.

A minimizer μ_r in r is *solid* w.r.t. a given cluster c_x iff its count is a tangible fraction of all the counts associated to μ_r (i.e. a row in M). In practice, for each minimizer we check if the ratio $M_{\mu_r,c_x}/\Sigma_i M_{\mu_r,c_i}$ is above a user-defined threshold σ. If yes, we consider μ_r to be a valid minimizer for similarity computation. See Algorithm 1 for an overview. Clustering is then performed as before, with sketched reads merged to the best cluster iff the number of shared (solid) minimizers is above threshold τ. Cluster merging is left as a future development and it is not used in any of the comparisons presented in Section 3.

3. Results

To evaluate `SolidClust`, we considered 3 real PacBio datasets with varying coverages. The datasets used in our experiments are described in Table 1. To put our results in perspective, we compared `SolidClust` with `isOnClust3`. We decided to not include `RATTLE` [21] and `GeLuster` [22] in our evaluation since from [23], `isOnClust3` resulted the most accurate and the only approach capable of analyzing recent long-read RNA-Seq datasets. The objective of our experimental evaluation was twofold: (i) to evaluate the impact of the new σ parameter and to establish the best value for the `SolidClust` parameters (τ and σ) that allow maximizing its clustering accuracy, and (ii) to compare the accuracy and efficiency of `SolidClust` with those of `isOnClust3`. Our experimental evaluation can be reproduced using the Snakemake workflow available at https://github.com/ldenti/solidclust.

```
Data: A list $L$ of sketches sorted by decreasing size
Result: Clustering $C$ of $L$
$C \leftarrow L[0]$;
$M \leftarrow$ matrix of counters;
for minimizer $\mu$ in $L[0]$ do
    $M[\mu] \leftarrow 0$;
    $M[\mu][0] \leftarrow 1$;
end
$i \leftarrow 1$;
while $i < |L|$ do
    $r \leftarrow L[i]$;
    $hits \leftarrow$ vector of 0 of size $|C|$;
    for minimizer $\mu_r$ in $r$ do
        if $\mu_r$ already seen before then
            $Sum \leftarrow \Sigma_i M[\mu_r][c_i]$;
            foreach $c_x$ in row $M[\mu_r]$ do
                if $M[\mu_r][c_x] \div Sum > \sigma$ then
                    $hits[c_x] \leftarrow hits[c_x] + 1$;
                end
            end
        end
    end
    $b \leftarrow \underset{x}{\mathrm{argmax}}(hits[x])$;
    $ratio \leftarrow M[\mu_r][c_b] / |r|$;
    if $ratio > \tau$ then
        Add $r$ to $C[b]$;
        if $\mu_r$ in $M$ then
            if $b$ in $M[\mu_r]$ then
                $M[\mu_r][b] \leftarrow M[\mu_r][b] + 1$;
            else
                $M[\mu_r] \leftarrow b$;
                $M[\mu_r][b] \leftarrow 1$;
            end
        else
            $M[\mu_r] \leftarrow b$;
            $M[\mu_r][b] \leftarrow 1$;
        end
    else
        $C.push(L[i])$;
        $b \leftarrow |C| - 1$;
        for minimizer $\mu_r$ in $r$ do
            $M[\mu_r] \leftarrow M[\mu_r] \cup b$;
            $M[\mu_r][b] \leftarrow 1$;
        end
    end
    $i \leftarrow i + 1$;
end
```

Algorithm 1: `SolidClust` algorithm.

Dataset	No. reads	Unclassified (%)	NS	S	Reads in NS (%)
PB	354 678	6 128 (1.73)	8 583	2 508	99.3
ALZ	4 277 293	6 278 (0.15)	17947	12573	99.7
HG002	37 200 651	30 738 (0.08)	19 725	30 481	99.9

Table 1
Datasets used in the experimental evaluation. NS: non-singleton clusters, S: singleton clusters. The ALZ and HG002 are the same datasets used in [23]. PB dataset has been downloaded from the Sequence Read Archive (SRR33026141)

3.1. Evaluation criteria

We evaluated the accuracy of the clustering in the same way as proposed in [23]. For completeness, we intuitively describe here the methodology used to create the ground truth and the evaluation metrics.

The ground truth (i.e., the true clusters) is computed using a reference-guided clustering. We considered the T2T reference genome [31] and we used read alignments computed with minimap2 [26] as a proxy to assign a class (i.e., the ground truth cluster) to each read. Although such a proxy can result in misclassifications due to misalignments and mapping artifacts, it is the benchmarking approach used in many previous studies [23]. A read is denoted as "unclassified" if it could not be aligned. A true cluster is defined as "non-singleton" (NS) if it contains more than one read, "singleton" otherwise. Table 1 reports the results of this ground truth creation for the datasets considered in our evaluation.

As in [23], we used the V-measure [32] and the Adjusted Rand Index [33] to evaluate the clustering accuracy. We will give here the intuition behind these metrics, referring the reader to the corresponding papers for their formal definition. The V-measure (V) is computed as the harmonic mean between homogeneity (h) and completeness (c). A clustering is homogeneous if its clusters contain reads coming from the same class whereas it is complete if reads coming from the same class are assigned to the same cluster. All these metrics range from 0 to 1. A homogeneity of 1 indicates that all clusters contain only members of a single class whereas a completeness of 1 indicates that all members of a given class are assigned to the same cluster. In other terms, homogeneity penalizes over-clustering and completeness penalizes under-clustering. The Adjusted Rand Index (ARI), instead, measures the percentage of correct pairings of elements (reads), corrected for chance agreements. The ARI evaluates the similarity between two different clusterings and provides a score in the range $[-1, 1]$, where 1 indicates perfect agreements, 0 indicates random clustering, and negative values indicate a clustering worse than random. These measures are computed using the same python script used in the evaluation described in the isOnClust3 paper.

3.2. Impact of parameters τ and σ

We first focus on the smallest dataset (PB) and we evaluated the impact of the parameters τ and σ on the clustering accuracy of SolidClust. To this end, we ran SolidClust with different combinations of the two parameters, namely $\tau \in \{0.1, 0.25, 0.5\}$ and $\sigma \in \{0, 0.1, 0.25, 0.33, 0.5, 0.9\}$. We decided to focus on these two parameters since σ is the new parameter introduced in SolidClust and τ is highly affected by the choice of σ. Indeed, by considering only *solid* minimizers, we may end up with a lower similarity between a read and a cluster, since we might reduce the number of shared minimizers (numerator in the Jaccard similarity formula) but the total number of minimizers from the read (the denominator) does not change. We did not test the other parameters (minimizer size k, window length w, and quality threshold q) since we believe that they do not have a great impact on the results of SolidClust. As in [23], we set $k = 15$, $w = 51$, and $q = 0.98$ (since the dataset is a PacBio dataset). To allow for a better comparison, we also run the original isOnClust3 varying its similarity threshold parameter τ, which is independent of the sequencing technology and, by default, is set to 0.5. Figure 1 reports the results of this analysis in terms of V-measure and ARI. Table 2 reports the full results, including completeness, homogeneity, and number of clusters reported by each approach.

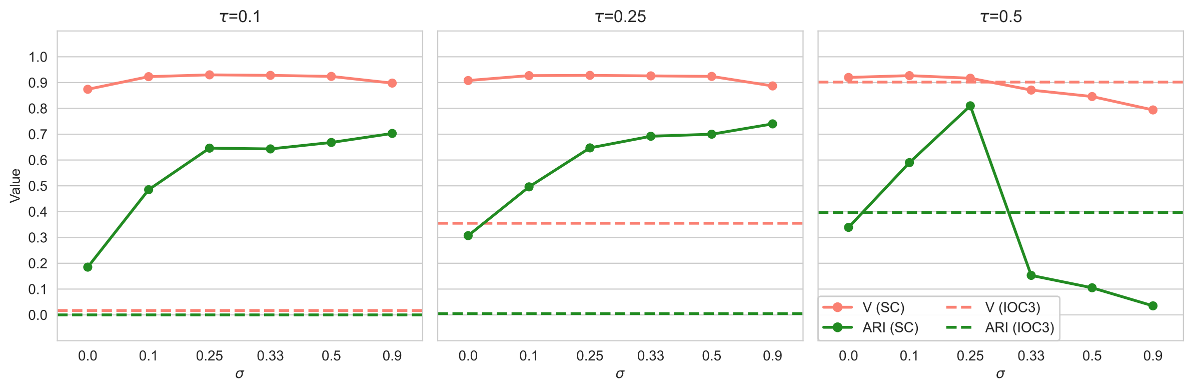

Figure 1: V-measure (V) and Adjusted Rand Index (ARI) results on the PB dataset. Each subplot presents the results obtained using a different value for the τ parameter. The x-axis reports the values for the σ parameter. Results are provided for `isOnClust3` (IOC3 in the legend, single value independent from σ) and `SolidClust` (SC in the legend).

Overall, in terms of V-measure, τ and σ do not greatly affect the accuracy of `SolidClust`. `SolidClust` always achieves a very high V-measure, proving that its clustering is complete and homogeneous. The homogeneity of `SolidClust` clustering starts to drop when $\tau = 0.5$ and $\sigma > 0.25$, lowering the V-measure. This is somehow expected since using a high value for both parameters reduce the likelihood for a read to be included in an already created cluster, as shown by the number of clusters reported by `SolidClust` (Table 2). The same trend can be observed when considering the ARI values (which drops from 0.810 to 0.153 when moving from $\sigma = 0.25$ to $\sigma = 0.33$ in the case of $\tau = 0.5$). However, the choice of the parameter σ seems to have a greater impact on this measure. Indeed, increasing the σ parameter results in a higher ARI. This holds up to the point (combination of τ and σ) where ARI remains stable or drops significantly.

When comparing `SolidClust` to `isOnClust3`, we can clearly see that `isOnClust3` achieves higher accuracy when τ is set to 0.5 (that is the default value). However, even in this case, `SolidClust` is able to achieve higher clustering accuracy for some combinations of the τ and σ parameters. Remarkably, `SolidClust` is able to achieve good clustering accuracy even with low value of the τ parameter (when the parameter σ is correctly set). Overall, when using the default value for the τ parameter in `isOnClust3` and correctly setting the two `SolidClust` parameters (that is not a straightforward task), the two approaches achieve comparable results.

Finally, we notice that the clustering accuracy of `isOnClust3` and `SolidClust` (ran with $\sigma = 0$) greatly differs, especially for smaller values of τ. We recall that when σ is set to 0, `SolidClust` considers all minimizers from the reads and not only solid minimizers. Although we were expecting the two approaches to yield similar results in this setting, we believe that the different way the two approaches handle repetitions (i.e., those minimizers repeated more than once in a read) is the reason behind this discrepancy. This is an additional proof that, although `SolidClust` follows the same greedy methodology of `isOnClust3`, `SolidClust` can be considered a novel method that exhibits its own pecularities (described in Section 2 and here experimentally validated).

3.3. Clustering accuracy and efficiency

We then focus on the bigger datasets and we evaluated the clustering accuracy and efficiency of `SolidClust`. We first considered the ALZ dataset and ran `SolidClust` using $\tau \in \{0.25, 0.5\}$ and $\sigma \in \{0.1, 0.25, 0.5\}$. We decided to limit our analysis to these values since we believe they are the values that could provide the best accuracy results, based on the results presented in the previous section. Table 3 reports the results of this analysis.

The best clustering accuracy is achieved by `SolidClust` when τ is set to 0.25 and σ to 0.1. With this combination, `SolidClust` is able to achieve a V-measure of 0.878 (+0.017 w.r.t. the original `isOnClust3`

Approach	τ	σ	V	c	h	ARI	NS	S	NS+S
`isOnClust3`	0.10	0.00	.017	.761	.009	.000	59	921	980
`SolidClust`	0.10	0.00	.874	.934	.821	.185	7 850	9 447	17 297
`SolidClust`	0.10	0.10	.923	.945	.902	.485	9 199	11 750	20 949
`SolidClust`	0.10	0.25	.930	.936	.925	.646	9 327	17 817	27 144
`SolidClust`	0.10	0.33	.928	.928	.927	.643	9 364	21 081	30 445
`SolidClust`	0.10	0.50	.924	.915	.933	.668	9 465	27 530	36 995
`SolidClust`	0.10	0.90	.898	.845	.957	.703	9 887	62 866	72 753
`isOnClust3`	0.25	0.00	.355	.905	.221	.005	2 514	5 901	8 415
`SolidClust`	0.25	0.00	.908	.932	.885	.307	9 866	13 711	23 577
`SolidClust`	0.25	0.10	.927	.939	.914	.496	9 941	15 164	25 105
`SolidClust`	0.25	0.25	.928	.917	.938	.647	10 004	26 830	36 834
`SolidClust`	0.25	0.33	.926	.908	.945	.692	10 032	31 777	41 809
`SolidClust`	0.25	0.50	.924	.900	.950	.700	10 061	37 120	47 181
`SolidClust`	0.25	0.90	.887	.815	.974	.740	10 202	88 837	99 039
`isOnClust3`	0.50	0.00	.902	.894	.911	.397	9 987	33 902	43 889
`SolidClust`	0.50	0.00	.920	.918	.923	.339	11 165	20 691	31 856
`SolidClust`	0.50	0.10	.927	.880	.980	.590	10 913	45 169	56 082
`SolidClust`	0.50	0.25	.917	.858	.985	.810	10 646	62 014	72 660
`SolidClust`	0.50	0.33	.871	.776	.992	.153	10 497	103 421	113 918
`SolidClust`	0.50	0.50	.846	.735	.995	.105	9 976	139 985	149 961
`SolidClust`	0.50	0.90	.794	.659	.999	.035	7 863	232 450	240 313

Table 2
Results on the `PB` dataset. Ground truth composed of 11 091 clusters: 8 583 non-singleton (NS) and 2 508 singleton (S).

ran with $\tau = 0.5$) and an ARI of 0.701 (+0.038). Surprisingly, all other tested combinations of τ and σ provide worse results. In particular, the number of singleton clusters computed by `SolidClust` is extremely high. This is in contrast to what we saw when considering the smaller `PB` dataset. We therefore believe that the choice of the two parameters should also depend on the expected coverage of the input sample and the expected number of clusters (e.g., how many genes have been sequenced). Remarkably, the best run of `SolidClust` ($\tau = 0.25$, $\sigma = 0.1$) was twice as fast as the original `isOnClust3` while requiring 1/3 of the RAM (11GB instead of 37GB). We note that the efficiency of `SolidClust` seems directly proportional to the number of clusters computed. The more clusters, the longer the running time and the higher the memory requirement. However, it is not fully clear how the number of clusters produced in output is affected by the two tested parameters τ and σ. Investigating this relationship is a compelling future direction that we plan to explore.

We finally analyzed the biggest dataset (`HG002`), comprising 37 milion reads. Following the results on the `ALZ` dataset, we ran `SolidClust` setting τ to 0.25 and σ to 0.1 and 0.25. Unfortunately, we did not manage to run `isOnClust3` on our cluster due to, we suspect, some I/O issue (the process got stuck in uninterruptible sleep state). For this reason, we report the results (accuracy and efficiency) presented in the original paper [23]. Results of this analysis are presented in Table 4. `SolidClust` has been able to achieve the best clustering accuracy, but this time there is no clear winner between the two tested combinations of τ and σ. Indeed, one of the two achieved the highest V-measure whereas the other achieved the highest ARI. In any case, both combinations achieved a very high ARI compared to the original `isOnClust3` (+0.233/+0.257). However, similarly to the `ALZ` dataset, setting σ to 0.25 produced a very high number of singleton clusters and increased the running times of `SolidClust`, making it slower than `isOnClust3`. For this reason, we believe that the best combination of parameters is $\tau = 0.25$ and $\sigma = 0.1$. In this setting, `SolidClust` has been able to achieve very high clustering accuracy while resulting 1.6 times faster and 4.85 times less memory intensive than `isOnClust3`. We note that

Approach	τ	σ	V	c	h	ARI	NS	S	NS+S	Time	RAM
isOnClust3	0.25	0.00	.608	.746	.513	.003	55 082	56 913	111 995	1:24	37
SolidClust	0.25	0.10	.878	.787	.994	.701	95 702	421 110	516 812	0:48	11
	0.25	0.25	.843	.731	.996	.684	46 001	1 169 467	1 215 468	1:21	12
	0.25	0.50	.771	.628	.999	.595	20 317	2 512 281	2 532 598	2:25	15
isOnClust3	0.50	0.00	.861	.762	.991	.663	140 809	449 312	590 121	1:29	37
SolidClust	0.50	0.10	.804	.673	.999	.631	30 129	1 879 539	1 909 668	1:57	14
	0.50	0.25	.733	.579	1.00	.530	9 599	3 272 111	3 281 710	3:07	17
	0.50	0.50	.696	.534	1.00	.029	3 218	4 021 109	4 024 327	4:36	19

Table 3
Results on the ALZ dataset. Ground truth composed of 30 520 clusters: 17 947 non-singleton (NS) and 12 573 singleton (S).

Approach	τ	σ	V	c	h	ARI	NS	S	NS+S	Time	RAM
isOnClust3	0.50	0.00	.860	.760	.990	.580	-	-	-	8:40	165.2
SolidClust	0.25	0.10	.881	.793	.991	.813	123 676	328 020	451 696	5:15	34
	0.25	0.25	.868	.771	.993	.837	96 121	1 649 156	1 745 277	13:20	35

Table 4
Results on the HG002 dataset. Ground truth composed of 50 206 clusters: 19 725 non-singleton (NS) and 30 481 singleton (S).

increasing the value of τ and σ parameters substantially impacts the running times of `SolidClust` but not its memory requirements.

4. Discussion

We presented `SolidClust`, an approach for the de novo clustering of third-generation transcriptomic datasets. Heavily inspired by its predecessor (`isOnClust3`), `SolidClust` implements a greedy algorithm that iteratively merges reads based on the fraction of minimizers shared between a read and the already created clusters. The main novelty of `SolidClust` is the usage of *solid* minimizers: when a read is compared to a cluster, `SolidClust` does not use all minimizers of the read but it filters out all minimizers that potentially come from sequencing errors, thus considered *non solid*. As shown in our experimental evaluation, `SolidClust` is able to achieve very high clustering accuracy and is as fast as its predecessor for comparable number of clusters while requiring less memory.

In this work, we considered 3 PacBio datasets and we analyzed the impact of two of `SolidClust` parameters (τ and σ) on its accuracy and efficiency. Future works will be devoted to evaluate `SolidClust` on ONT datasets which comprise longer and (usually) less accurate reads than the PacBio datasets used in this evaluation. An additional compelling future direction consists in devising an automated way to select the best values for the parameters τ and σ. Although we believe that this choice mainly depends on the sequencing technology (hence the expected error rate), the coverage of the datasets, and the expected number of clusters (i.e., the number of sequenced genes), being able to provide automatic parameter selection requires further investigation and additional experiments.

Acknowledgments

This research work has received funding from the European Union's Horizon programme under the Horizon Europe grant agreement (ASVA-CGR No. 101180581 to L.D.). This work has also been supported by the European Union's Horizon 2020 research and innovation programme under the Marie Skłodowska-Curie grant agreement PANGAIA No. 872539 (Y.S.).

Declaration on Generative AI

During the preparation of this work, the author(s) used Writefull and GPT-4o mini in order to: (i) Paraphrase and reword and (ii) Grammar and spelling check. After using these tool(s)/service(s), the author(s) reviewed and edited the content as needed and take(s) full responsibility for the publication's content.

References

[1] A. Sveen, S. Kilpinen, A. Ruusulehto, R. Lothe, R. Skotheim, Aberrant rna splicing in cancer; expression changes and driver mutations of splicing factor genes, Oncogene 35 (2016) 2413–2427.

[2] S. C. Bonnal, I. López-Oreja, J. Valcárcel, Roles and mechanisms of alternative splicing in cancer—implications for care, Nature reviews Clinical oncology 17 (2020) 457–474.

[3] G. Biamonti, A. Amato, E. Belloni, A. Di Matteo, L. Infantino, D. Pradella, C. Ghigna, Alternative splicing in alzheimer's disease, Aging clinical and experimental research 33 (2021) 747–758.

[4] M. Bhadra, P. Howell, S. Dutta, C. Heintz, W. B. Mair, Alternative splicing in aging and longevity, Human genetics 139 (2020) 357–369.

[5] G. Yeo, D. Holste, G. Kreiman, C. B. Burge, Variation in alternative splicing across human tissues, Genome biology 5 (2004) 1–15.

[6] B. Li, C. N. Dewey, Rsem: accurate transcript quantification from rna-seq data with or without a reference genome, BMC bioinformatics 12 (2011) 1–16.

[7] C. Trapnell, A. Roberts, L. Goff, G. Pertea, D. Kim, D. R. Kelley, H. Pimentel, S. L. Salzberg, J. L. Rinn, L. Pachter, Differential gene and transcript expression analysis of rna-seq experiments with tophat and cufflinks, Nature protocols 7 (2012) 562–578.

[8] S. Shen, J. W. Park, Z.-x. Lu, L. Lin, M. D. Henry, Y. N. Wu, Q. Zhou, Y. Xing, rmats: robust and flexible detection of differential alternative splicing from replicate rna-seq data, Proceedings of the national academy of sciences 111 (2014) E5593–E5601.

[9] M. Pertea, G. M. Pertea, C. M. Antonescu, T.-C. Chang, J. T. Mendell, S. L. Salzberg, Stringtie enables improved reconstruction of a transcriptome from rna-seq reads, Nature biotechnology 33 (2015) 290–295.

[10] L. Denti, R. Rizzi, S. Beretta, G. D. Vedova, M. Previtali, P. Bonizzoni, Asgal: aligning rna-seq data to a splicing graph to detect novel alternative splicing events, BMC bioinformatics 19 (2018) 1–21.

[11] J. A. Sibbesen, J. M. Eizenga, A. M. Novak, J. Sirén, X. Chang, E. Garrison, B. Paten, Haplotype-aware pantranscriptome analyses using spliced pangenome graphs, Nature Methods 20 (2023) 239–247.

[12] S. Ciccolella, D. Cozzi, G. Della Vedova, S. N. Kuria, P. Bonizzoni, L. Denti, Differential quantification of alternative splicing events on spliced pangenome graphs, PLOS Computational Biology 20 (2024) e1012665.

[13] A. Byrne, C. Cole, R. Volden, C. Vollmers, Realizing the potential of full-length transcriptome sequencing, Philosophical Transactions of the Royal Society B 374 (2019) 20190097.

[14] S. Kovaka, A. V. Zimin, G. M. Pertea, R. Razaghi, S. L. Salzberg, M. Pertea, Transcriptome assembly from long-read rna-seq alignments with stringtie2, Genome biology 20 (2019) 1–13.

[15] L. H. Tung, M. Shao, C. Kingsford, Quantifying the benefit offered by transcript assembly with scallop-lr on single-molecule long reads, Genome biology 20 (2019) 1–18.

[16] B. Orabi, N. Xie, B. McConeghy, X. Dong, C. Chauve, F. Hach, Freddie: annotation-independent detection and discovery of transcriptomic alternative splicing isoforms using long-read sequencing, Nucleic Acids Research 51 (2023) e11–e11.

[17] A. D. Prjibelski, A. Mikheenko, A. Joglekar, A. Smetanin, J. Jarroux, A. L. Lapidus, H. U. Tilgner,

Accurate isoform discovery with isoquant using long reads, Nature Biotechnology 41 (2023) 915–918.

[18] C. Monzó, T. Liu, A. Conesa, Transcriptomics in the era of long-read sequencing, Nature Reviews Genetics (2025) 1–21.

[19] C. Marchet, L. Lecompte, C. D. Silva, C. Cruaud, J.-M. Aury, J. Nicolas, P. Peterlongo, De novo clustering of long reads by gene from transcriptomics data, Nucleic Acids Research 47 (2019) e2–e2.

[20] K. Sahlin, P. Medvedev, De novo clustering of long-read transcriptome data using a greedy, quality value-based algorithm, Journal of Computational Biology 27 (2020) 472–484.

[21] I. de la Rubia, A. Srivastava, W. Xue, J. A. Indi, S. Carbonell-Sala, J. Lagarde, M. M. Albà, E. Eyras, Rattle: reference-free reconstruction and quantification of transcriptomes from nanopore sequencing, Genome Biology 23 (2022) 153.

[22] J. Ma, X. Zhao, E. Qi, R. Han, T. Yu, G. Li, Highly efficient clustering of long-read transcriptomic data with geluster, Bioinformatics 40 (2024) btae059.

[23] A. J. Petri, K. Sahlin, De novo clustering of large long-read transcriptome datasets with isonclust3, Bioinformatics 41 (2025) btaf207.

[24] M. Roberts, W. Hayes, B. R. Hunt, S. M. Mount, J. A. Yorke, Reducing storage requirements for biological sequence comparison, Bioinformatics 20 (2004) 3363–3369. URL: https://doi.org/10.1093/bioinformatics/bth408. doi:10.1093/bioinformatics/bth408.

[25] S. Schleimer, D. S. Wilkerson, A. Aiken, Winnowing: local algorithms for document fingerprinting, in: Proceedings of the 2003 ACM SIGMOD International Conference on Management of Data, SIGMOD '03, Association for Computing Machinery, New York, NY, USA, 2003, p. 76–85. URL: https://doi.org/10.1145/872757.872770. doi:10.1145/872757.872770.

[26] H. Li, Minimap2: pairwise alignment for nucleotide sequences, Bioinformatics 34 (2018) 3094–3100.

[27] G. E. Pibiri, Y. Shibuya, A. Limasset, Locality-preserving minimal perfect hashing of k-mers, Bioinformatics 39 (2023) i534–i543. URL: https://doi.org/10.1093/bioinformatics/btad219. doi:10.1093/bioinformatics/btad219.

[28] B. Ekim, B. Berger, R. Chikhi, Minimizer-space de bruijn graphs: Whole-genome assembly of long reads in minutes on a personal computer, Cell Systems 12 (2021) 958–968.e6. URL: https://www.sciencedirect.com/science/article/pii/S240547122100332X. doi:https://doi.org/10.1016/j.cels.2021.08.009.

[29] R. Groot Koerkamp, G. E. Pibiri, The mod-minimizer: A Simple and Efficient Sampling Algorithm for Long k-mers, in: 24th International Workshop on Algorithms in Bioinformatics (WABI 2024), Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2024, pp. 11:1–11:23. URL: https://drops.dagstuhl.de/entities/document/10.4230/LIPIcs.WABI.2024.11. doi:10.4230/LIPIcs.WABI.2024.11.

[30] B. Ewing, L. Hillier, M. C. Wendl, P. Green, Base-Calling of Automated Sequencer Traces Using-Phred. I. Accuracy Assessment, Genome Research 8 (1998) 175–185. URL: http://genome.cshlp.org/content/8/3/175. doi:10.1101/gr.8.3.175.

[31] S. Aganezov, S. M. Yan, D. C. Soto, M. Kirsche, S. Zarate, P. Avdeyev, D. J. Taylor, K. Shafin, A. Shumate, C. Xiao, et al., A complete reference genome improves analysis of human genetic variation, Science 376 (2022) eabl3533.

[32] A. Rosenberg, J. 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), 2007, pp. 410–420.

[33] L. Hubert, P. Arabie, Comparing partitions, Journal of classification 2 (1985) 193–218.