QuickDeconvolution: fast and scalable deconvolution of linked-reads sequencing data

Roland Faure

QuickDeconvolution: fast and scalable deconvolution of linked-reads sequencing data

Roland Faure

Internship report
March - August 2021

Supervisor: Dominique Lavenier

Abstract

Linked reads technologies, such as the 10X chromium system, use microfluidics to tag multiple short reads coming from the same long (50-200 kbp) fragment with a small sequence, called *barcode*. They are cheap and easy to prepare. The fact that reads with the same barcode come from the same fragment of the genome is extremely rich in information and can be used in a myriad of software. However, the same barcode may be used several times for several different fragments, complicating the analyses. Here we present *QuickDeconvolution* (QD), a new software capable of *deconvoluting* a set of reads sharing a barcode, *i.e.* telling separating reads coming from the different fragments. QD only takes as input the sequencing data, without the need for a reference genome. Compared to existing software, we show on made-up examples that *QuickDeconvolution* is more precise and faster than existing software, especially with many threads. More importantly, it is more scalable and therefore capable of deconvolving datasets that were inaccessible to previous software. We demonstrate here the first example in the literature of a successfully deconvolved animal genome, a *Drosophila melanogaster* dataset of 33 Gbp.

Contents

1 Introduction

1.1 Working place

My internship took place at the IRISA in Rennes, one of the largest French research laboratory, a joint-venture resulting from the collaboration of several institutions, among which the university of Rennes 1 and the Institut national de recherche en informatique et en automatique (INRIA). The research interests of the lab span a wide range of computer science aspects including bioinformatics.

Three teams focus on bioinformatics within this lab. One of them, Genouest, provides computing facilities. Dyliss' publications revolve around the understanding of metabolic networks and of protein function. Genscale, the team in which I did my internship, is specialized in the treatment of big sequence datasets. The teams are physically mixed in the building so that all the bioinformaticians of the IRISA know each other very well.

1.2 Sequencing technologies

Since the discovery of the role of DNA in transmitting genetic information [1], it has been understood that obtaining the genomes of organisms is key to understanding their biology. Thus, great efforts have been made to extract genomic information from a diversity of organisms, including humans [2].

The first modern sequencers, with which scientists were able to retrieve the precise sequence of a (small) strand of DNA, appeared around 1970 and were named after Sanger, the scientist who created the technique [3]. Each fragment of sequenced DNA that comes out of a sequencer is called a read. Sanger sequencing provides accurate reads of more than 500 base-pairs long and often up to 1000-1200 bp. Though not fully extinct, first generation sequencing has become rare, outpaced by second- and third-generation sequencing whenever large datasets are preferable.

In the late 90s appeared high-throughput (also called second-generation) sequencing methods, starting with 454 pyrosequencing [4] and followed by Illumina sequencing, which are highly scalable and able to produce much more data than Sanger sequencing [5]. These methods are characterized by their ability to produce many short reads (less than 300 bp) with a very low error rate (typically less than 1%). These are still nowadays widely used and appreciated for their high precision and low cost per base.

Third-generation sequencing appeared in the last few years through two main companies, Oxford Nanopore Technologies [6] and Pacific Biosciences [7]. In contrary to second-generation approaches that rely (like Sanger sequencing) on an amplification step, third-generation sequencers can read the bases on a single strand of DNA without prior amplification. For this reason, these techniques allow the production of reads of much greater length (more than 10 kbp and up to two millions bp in extreme cases [8]). However, an important limitation of these technologies is a high error rate of several percent with non-uniform errors pattern in some cases [9].

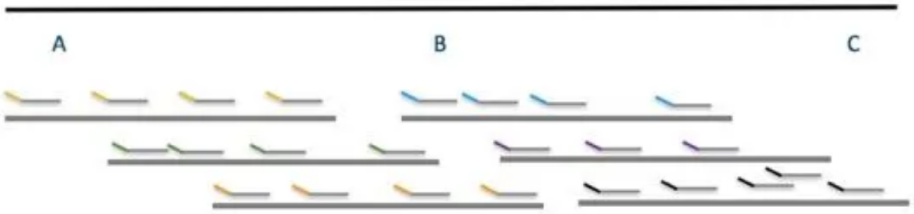


Figure 1: Illustration of the linked-reads sequencing principle. The genome is represented as the black line at the top. Fragments of the genome, represented as thick grey lines, are isolated in separate droplets and partially sequenced with short reads. A barcode, represented as a color, is attached to all reads originating from one fragment. The sequencing data is the barcoded short reads.

1.3 Linked-reads

Linked-reads technologies were developed as a compromise between short and accurate reads and long, imprecise reads. 10X sequencing is its oldest and most widespread form but today a variety of new techniques are emerging, such as LoopSeq, TELL-Seq and BGI's long fragment reads. To produce linked-reads, long DNA fragments are isolated in oil droplets and sequenced with short reads. The short reads cover typically 10-20% of the length of the fragment. At the end of each read is attached a 'barcode' which is a small DNA sequence. All reads originating from the same fragment will share the same barcode. Figure 1, from the 10X website, illustrates this principle.

Using the terminology defined in previous papers [10, 11], the set of reads sharing the same barcode will be referred to as a *read cloud*. The long-range information contained in barcodes (two reads sharing the same barcode are probably 'not far away' on the DNA strand) can be exploited by appropriate software, while being much cheaper and easier to prepare than long-reads sequencing [12]. Typically, linked-reads can be used to phase haplotypes [13] or to propose better *de novo* genome assembly [14]. For instance, a reference for the pepper genome was provided in 2018 using linked reads [15].

A new computational challenge arising from these technologies is that the total number of barcode is limited. The 10X sequencing technology, for example, only provides a few million barcodes. Since the total number of fragments routinely exceeds this number, barcodes have to be used several times for several different fragments. This complicates the exploitation of the data. The barcode deconvolution problem can be defined as separating the reads coming from the different fragments present in each barcode. The ultimate goal is to have "enhanced barcodes", where each barcode identify uniquely reads coming from a single DNA fragment.

1.4 State of the art

Sometimes the content of the sequencing experiment is approximately known beforehand and reference genomes are available, for example when sequencing a model organism such as the human or the drosophila. In this case, the sim-

plest technique of deconvolution is the one implemented by EMA [16]. EMA maps all reads from each barcode of a 10X experiment to a reference genome. If the sequenced genome is similar enough to the reference genome, reads will map. Since reads originating from the same fragment are close to one another on the sequenced genome there is good probability that they map also close to each other on the reference genome. That is, if the general region of the fragment is mostly conserved between the reference and the experiment. If the sample is a metagenome, a variation of this algorithm is to provide several reference genomes, one for each species represented. The major limitation of this approach is the need for a reference genome. References of good quality are not always available, and often the species contained in the sample are not known beforehand. Moreover, the result will be biased by the reference genome: using two different references may yield two different solutions out of fully conserved regions.

The first reference-free barcode deconvolution software has been published in [10], under the name of *Minerva*. It uses the fact that the sample is sequenced with a certain *coverage i.e.* that all portions of the genome are sequenced several times, generally more than 20 times. Many fragments with different barcodes will thus originate from the same region, which Minerva can exploit. The principle behind Minerva is the same as the one behind our software, *Quick-Deconvolution* (QD) and will be discussed in length later. The paper laid solid theoretical background to this method and showed its application on two sets of mock metagenomes. However, the method remained excessively slow to be used on large or even medium-sized datasets and is qualified by its authors as a 'proof-of-concept' algorithm.

Very recently another reference-free software by the authors of Minerva was proposed under the name of *Ariadne* [11]. Based on a totally different concept, Ariadne begins by doing a full De-Bruijn-Graph assembly of the reads using the assembler SPAdes [17], ignoring the barcodes. It then proceeds barcode per barcode. The key idea is that two reads coming from the same fragment should not be far from each other on the assembly graph. Ariadne thus considers that if two reads sharing the same barcode are close on the assembly graph then they originate from the same fragment, and on the contrary that they originate from different fragments if they are far away on the assembly graph.

2 Results

All software were run on the GenOuest cluster, a computing facility maintained by the GenOuest team here in Rennes. GenOuest provides their resources to all researcher present at the IRISA. The jobs were run on a node with 3,1 Tb RAM available and up to 160 threads. The processor frequencies were of 3.2 GHz.

2.1 Accuracy

The most important thing to measure is how well QuickDeconvolution actually deconvolves linked-reads. To test QD on that aspect we used three datasets.

2.1.1 Datasets

For a first approach we wanted a fully artificial dataset with simulated reads, where we knew perfectly the solution of the deconvolution. Since Minerva is known not to run on big datasets, we chose as a basis the genome of *Escherichia coli*. To introduce a little complexity and because linked reads have often been used to phase haplotypes, we created a "fake diploid" *E. coli* by duplicating the genome and introducing 3% difference in random chunks, obtaining overall a 1% difference between the two chromosomes. Then, we simulated a simple, ideal, linked-reads sequencing experiment resembling 10X. Fragments of 70-130kbp were drawn uniformly along the genome. 15% of the length of the fragment was covered by paired-end 150bp reads with 1% error. A barcode was then randomly assigned to all these reads. Enough fragments were drawn to obtain a final coverage of 50 (*i.e* each base of the genome is covered by 50 reads on average, so $50/0.15 = 333$ fragments). The total number of barcodes available was computed to be 4 times less than the total number of fragments. That resulted in a 0.6 Gb dataset.

The second dataset was once again simulated, to be sure exactly of the solution. However, it was simulated using LRSim [18], a linked-read simulator built to reproduce biases and errors of linked reads sequencing. The simulator was run on chromosome 1 of genome of *Homo sapiens*. The sequenced data was 7 Gb.

For a third example we introduced a much bigger dataset than what was found in the papers of Minerva and Ariadne. The reads came from the 10X sequencing of a *Drosophila melanogaster* specimen, totalling 33 Gb of sequencing data. No attempts to deconvolve reads from an animal genome were reported previously in the litterature. To measure how well QD deconvolved the reads we needed to know the solution of the deconvolution. To do that we used the same approach as EMA: we mapped all the reads to a reference genome using Bowtie2. 74% of the reads mapped uniquely on the reference genome. Then reads that had the same barcode and mapped less than 100000bp away on the same chromosome were considered as coming from the same fragments. Fragments containing less than 5 reads were considered dubious, potentially resulting from error in mapping or differences between the reference and the sequenced animal. Overall, 60% of the reads remained and were identified to their fragment of origin with good confidence. Only the deconvolution of these reads was evaluated.

2.1.2 Metrics

In this section, a 'fragment' will refer to a set of reads coming from a fragment according to the solution and 'cloud' will refer to a set of reads proposed by a software as coming from one fragment.

To evaluate the deconvolution, we have decided to present two metrics, which is classical when trying to evaluate a clustering. Indeed, good deconvolution is a compromise between two extremes. On one hand, separating too much and classifying each read in its own cloud is perfect in the sense that all clouds contain only reads coming from one fragment yet is useless. On the

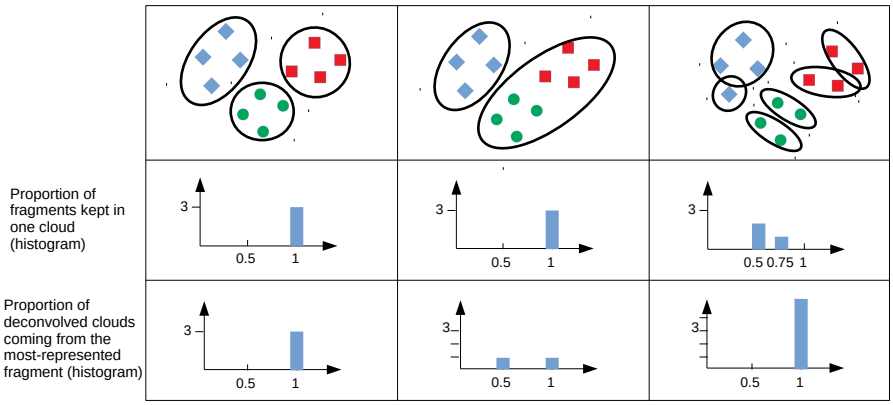


Figure 2: Illustration of the two metrics used to evaluate deconvolution. First row of the table, the set of shapes represent a set of reads with the same barcode, while their shape and color indicate that they come from 3 distinct fragments. The black ellipses illustrate three different possible deconvolution. Second and third row illustrate the histogram of the metrics chosen to evaluate the deconvolution. The first column is a perfect deconvolution. The second column represent a 'careful' deconvolution, where two fragments are still convoluted. The third column represent an 'over-deconvolution', where too many clouds are inferred.

other hand, not separating enough and classifying all reads in an unique cloud is perfect in the sense that all reads of a fragment will still be together in the same cloud but in this case the software did not perform any deconvolution.

The first metric we'll use is applicable to each fragment. We'll measure if all those reads are kept together as they should or if the deconvolution tend to separate fragments in clouds too small. Let N be the number of reads in the fragment. It is measured as follow:

- The cloud containing the most reads of the fragment is found.
- Let's say it contains n reads from the fragment. The measure is n/N

If all reads contained in the fragment are kept together in one cloud, this measure will be 1. In worst case scenario - no pair of reads of the fragment were kept together - it will be $1/N$. This value is measured for all fragments and a violin plot is drawn.

The second metric is applicable to each cloud. It measures if the deconvolution was efficient, *i.e.* if the reads coming from the different fragments have indeed been separated into different clouds. Let N be the number of reads in the cloud. It is measured as follow:

- The fragment best represented in the cloud is found.
- Let's say n reads of the cloud come from this fragment. The measure is n/N

If the cloud comes only from one fragment, the measure will be 1. However, if the cloud still contains two full fragments of equal size, the measure will be 0.5. This value is measured for all clouds with size $\geq$5 (as to not bias the plot with many very small clusters) and a violin plot is drawn.

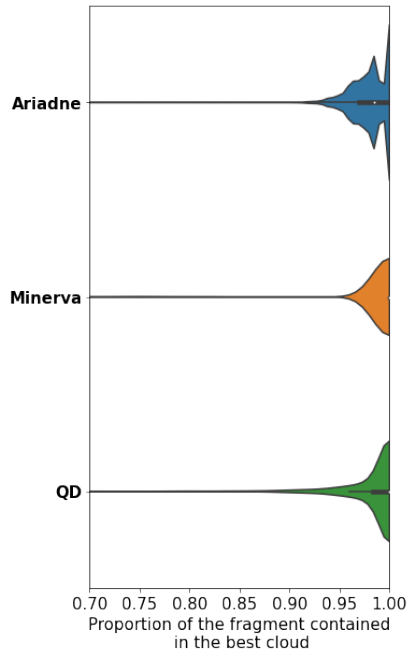
This measure can be applied also before deconvolution to the original clouds (1 cloud = 1 barcode). For example, if on average there are two fragments per barcode, we expect a bulge in the violin plot around 0.5. In the *E. coli* dataset, where there is on average 4 fragments per barcode, we observe a bulge around 0.25. The goal of the deconvolution will be move this distribution toward 1. A schematic view of these metrics is proposed figure 2.

2.1.3 Results

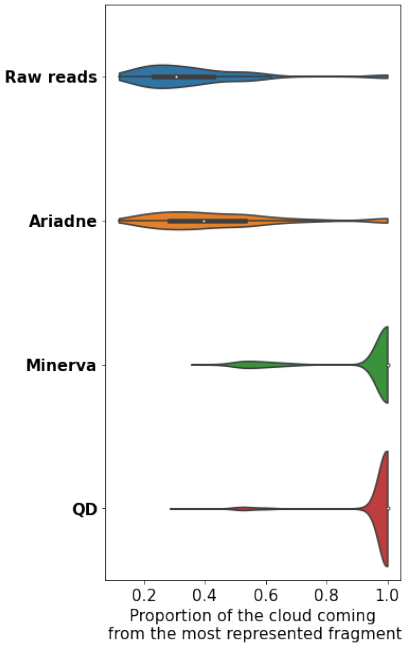
We run QD, Minerva and Ariadne on the three datasets. Minerva and Ariadne were ran with the parameters proposed on their GitHub. QuickDeconvolution was also run using default parameters, *i.e.* k=20 and indexing 1 over 8 k-mer on average.

The *Drosophila* and human dataset were too big for Minerva, which was killed by the cluster after 15 days of running. We were also unable to run Ariadne on these two datasets because it generated huge intermediary files ($\geq$12T) which saturated the space available.

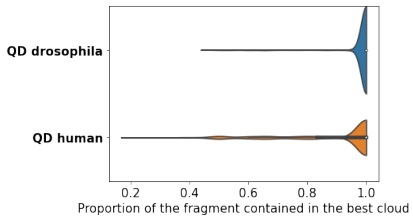
On the *E. coli* dataset, Minerva returned a deconvolution for only 1.4% of the reads, Ariadne for 69% of the reads and QuickDeconvolution more than 99.9% of the reads. QD proposed a deconvolution for 81% of the reads of



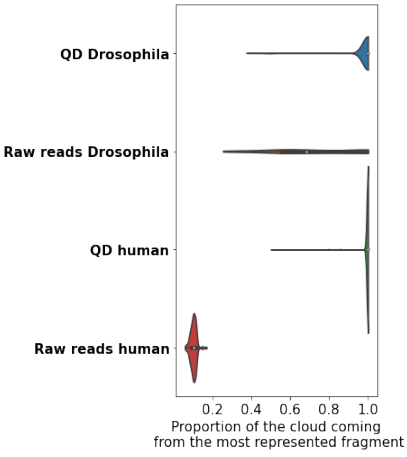
(a) Proportion of the reads coming from single fragments kept in one cloud after deconvolution. Dataset: simulated *E. coli*



(b) Proportion of deconvolved clouds coming from the most-represented fragment. Dataset: simulated *E. coli*



(c) Proportion of the reads coming from single fragments kept in one cloud after deconvolution. Dataset: *D. melanogaster* and *Homo sapiens chr. 1*



(d) Proportion of deconvolved clouds coming from the most-represented fragment. Dataset: *D. melanogaster* and *Homo sapiens chr. 1*

Figure 3: Evaluations of the quality of the deconvolution on two different datasets and for different software.

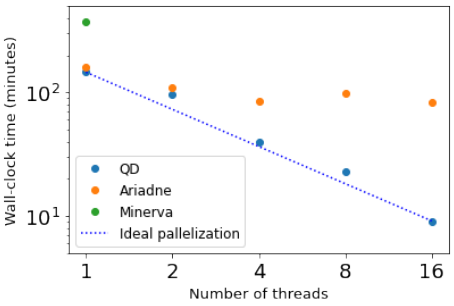


Figure 4: Run-time (in minutes) of the different deconvolution algorithms on the *E. coli* dataset. Axes have logarithmic scales. The dashed blue line represent the expected speed of QD if the parallelization was ideal.

Threads	1	2	4	8	16
QD	5,2	5,3	5,4	7,1	10,1
Ariadne	7,7	8,0	8,3	9,8	12,7
Minerva	10,7				

Figure 5: RAM usage (in gigabytes) of the different deconvolution software while deconvolving the *E. coli* dataset

the *Drosophila* dataset and more than 99.9% for the human dataset. Only the deconvolved reads have been taken into account to measure the quality of the clustering.

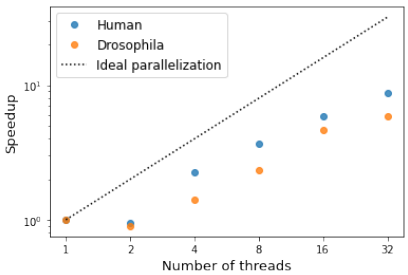
On figure 3a we see that a non-negligible number of fragments of *E. coli* have lost between 1 and 5% elements while being clustered by Ariadne and QD. For QD, they represent generally 2 or 3 reads that have been clustered separately from the rest of the cloud. Only on the human case (3c) did we see QD split some fragments in two clouds of roughly equal size, corresponding to the two ends of the fragment. The resulting clouds remain valid though, in the sense that all reads within each cloud are actually close to each other on the genome.

In term of deconvolution, QD proves superior to the other tools. Figures 3b and 3d shows that after deconvolution, the overwhelming majority of the clouds were composed for more than 90% of the same fragment, and in all datasets more than 75% of the clouds contained a unique fragment. Minerva shows similar performance on *E. coli*. Ariadne, however, hardly improved the deconvolution of the raw reads. We think it might be due to the fact that Ariadne is still in development and that the available, pre-published version is not fully operational yet.

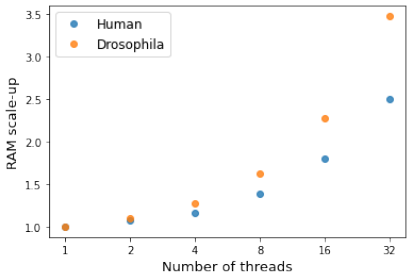
2.2 Performance

Now that it is established that *Quickdeconvolution* returns an accurate result, the question is its speed of execution. Speed was put forward by the authors of Minerva as the main limitation of their algorithm. Hence it was one of the main focus when developing QuickDeconvolution. We measured the run-time of the different algorithm using the command *time* of the Linux systems. We compared the algorithms on the *E. coli* dataset. The results can be appreciated figure 4. The program has been run several times with 1, 2, 4, 8 and 16 threads for the software able to parallelize their computations, *i.e.* Ariadne and QuickDeconvolution.

With one thread running, QD is faster by a few minutes than Ariadne. Both



(a) Speed-up of QD on the human and Drosophila datasets. Speed-up is defined as reference run-time over run-time. Reference for the speedup was taken when running on 1 threads, log scale.



(b) Scale-up of the RAM usage of QD on the human and Drosophila datasets. Scale-up is defined as reference RAM usage over RAM usage. Reference was taken for QD using 1 thread, non-log scale.

Figure 6: Behaviour of multithreaded QD on the human and Drosophila datasets

software run roughly twice as fast as Minerva. When increasing the number of threads the run-time of QD decreases almost ideally, *i.e* the program is n times faster when running on n threads, at least up to 16 threads. Ariadne scales much less well, since it is only twice as fast with 4 threads and does not seem to accelerate at all beyond. We end up with an order of magnitude of difference in run-time when running Ariadne and QD with 16 threads.

We conducted further investigation on the effect of parallelization on the much bigger human and *Drosophila* datasets. Figure 6a is a plot of the speed-up of QuickDeconvolution, *i.e.* the acceleration compared with a reference time, taken as the run-time of QD with 2 threads, which were approximately 7 hours for the human and 30 hours for the Drosophila. We observed a **cost of parallelization**: the difference of run-time between 1 and 2 threads is not significant, then the run-time decreases sharply when further adding threads. We explain this cost by the fact that memory access is globally slower because the dictionary is spread among several sub-dictionaries when the program is parallelized. For QD the parallelization is less interesting between 16 and 32 threads than before. This is an expected behaviour for all parallelized programs: as threads begin to compete for memory access, parallelization becomes less interesting.

RAM usage was significant: for the *Drosophila* dataset, the RAM usage ranged from 459 to 1053 G, while it ranged from 88 to 158 G for the *Homo sapiens* dataset. RAM usage tends to increase with the number of threads, even though all threads use a common memory space and that theoretically no extra information is stored. That is because it takes more memory to store 16 small dictionaries than 1 big dictionary. The *scale-up* of memory space used by QuickDeconvolution is plotted figure 6b, showing the increase of RAM used with multiple threads compared to the reference 1-threaded QD algorithm.

3 Methods and algorithm

3.1 Principle

The basic principle behind the QuickDeconvolution algorithm is the same as behind Minerva [10]. It relies on the fact that all regions of the genome are cloned during sequencing and will be sequenced several times: many fragments will thus come from the same region and share part of their sequence. If two fragments share the same sequence on part of their length they are called *overlapping*. Minerva and QuickDeconvolution uses the fact that several reads from overlapping fragments will likely overlap. In other words, several reads originating from one fragment will likely overlap several reads of an overlapping fragment with a different barcode. Fragments within a barcode can then be distinguished by the set of barcodes they overlap.

More precisely, each barcode is processed separately. For each barcode, let us call it the *anchor*, a bipartite graph is built, with all the reads from the anchor on one side, and all barcodes of the experiment on the other side. For each read in the anchor, the set of all overlapping reads (with an overlap $\geq k$ bp) in the sequencing data is found. Links are added in the graph between the read and the different barcodes of the overlapping reads. Once all the reads of the anchor are processed, the graph is complete. The bipartite graph is then converted in a graph containing only the reads of the anchor: two reads are linked by a link of strength *n* if they overlap with *n* common barcodes. Since reads from a same fragment tend to overlap with the same barcodes, it is expected that this graph can then be clustered, each cluster containing the reads coming from one fragment. This algorithm is illustrated figure 7.

3.1.1 Mathematical justification

The mathematical justification of the model has been very well described in [10] for metagenomic samples. We will propose thereafter a justification in the case of a multi-chromosomic genome.

Let us consider that fragments of length L are drawn with a uniform probability across the genome. Let p the proportion of each fragment covered by reads, typically 10-20% in the case of 10X data. Enough fragments are drawn to cover the genome with coverage c. Considering that two reads should overlap by half their length to be identified as overlapping, two reads distant by $l < L$ on the strand overlap on average with

$$\frac{L-l}{L} * c * p$$

common barcodes. Using values typical of 10X experiments, $L = 50000$, $c = 50$, $p = 0.15$, two reads distant of 25 kbp would overlap on average with 3.7 common barcodes.

Since barcodes are reused, let us say n times each in our model, one must also consider the possibility that two reads overlap with common barcodes even if they are far on the genome. Let N_{tot} be the number of barcodes available. Two reads far on the genome overlap on average $c * \frac{n-1}{N_{tot}}$ common barcodes.

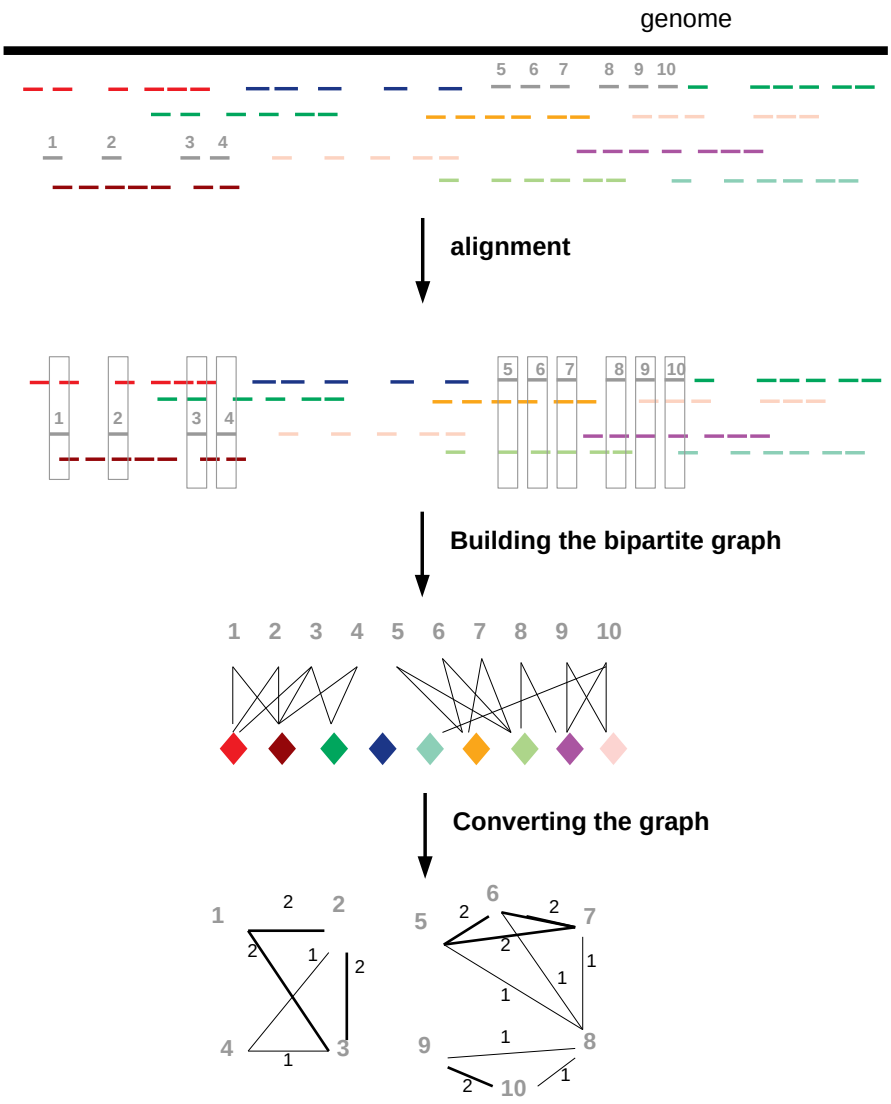


Figure 7: Illustration of the principle behind QuickDeconvolution and Minerva. Top: from the genome, reads are sequenced and barcoded. Barcodes are represented as colors. We will try to deconvolve the 10 reads of the grey cloud, that are actually coming from two different fragments (1234 and 5678910). Each grey read is aligned against all the other reads of the dataset. A bipartite graph linking the reads to all the barcodes they overlap is built. It is then converted to a graph between all the reads from the barcode. The strength of a link, indicated, is the number of common barcodes the two reads overlap. Minerva then outputs as result all the connected components of the graph, while QD clusters the graph.

With the conservative value $N_{tot} = 10^6$, $c = 50$ and $n = 11$ (corresponding to the sequencing a 1.65 Gb genome), this value equals 5.10^{-4}. In other words, two reads far away on the genome very rarely overlap with common barcodes by chance.

While the estimated strengths of the links in this model tend to vouch for very reliable links, it is important to keep in mind that in our model an overlap between two reads meant that the two reads came from the same region. In real genomes this is far from being the case, as repeated regions are extremely frequent and will cause many artefactual links between reads. Two reads may very well overlap with a common barcode because one of them contains a repeated sequence that points to an artefactual overlap with another read. The graph should thus be handled cautiously.

3.2 Alignment

In each barcode, all reads are processed iteratively. For each read, the set of all overlapping reads of the dataset must be found. Since there are million of fragments in the dataset, finding this set is one of the key difficulty of the program. The problem is well-known in the genome assembly community, where many overlapping reads need to be assembled into longer fragments [19]. The strategy implemented by QuickDeconvolution is a well-known strategy, based on *k-mers*.

k-mers are sub-sequences of length k present in the reads. During a preliminary indexing phase, a dictionary is built, reporting for each k-mer of the dataset the list of all its occurrences within the reads. For the deconvolution of the datasets of this report, a value of 20 has been used for k.

In a second phase, when processing a read, let us call it read α, the algorithm looks at each of its k-mer in the dictionary. If read α shares a k-mer with another read, the two reads are aligned to confirm that they overlap on their whole length. Since a read that overlap on $k + s$ bases with read α shares $s + 1$ k-mer with read α, no overlap will be missed.

To speed up the process of indexing and aligning, QuickDeconvolution only indexes a user-defined fraction of k-mers, that we call *sparse k-mers*. For example, if the user chooses to use only 1 out of 4 k-mer on average, only k-mers starting by 'A' will be indexed. The dictionary will be much smaller in size compared to a complete indexation, and only the sparse k-mers (starting with 'A') need to be looked at when trying to find all reads overlapping read α. A supplementary condition is added: if there are no 'A' on w bases in a row, k-mers starting by 'C' will be indexed, *on this stretch only*. This ensures that if two reads overlap by w bases or more they will share at least one sparse k-mer.

To maximize the speed of execution, QuickDeconvolution does not check the overlaps extensively: two reads sharing 3 sparse k-mers are automatically considered as overlapping.

3.3 Graph building

For each barcode, a graph linking all the reads from the cloud is built, as described above: first a bipartite graph between all the reads from the cloud and

the barcodes of all the reads they overlap is drawn, then it is converted in a graph containing only the reads of the cloud. Two reads are linked if they overlap respectively with at two reads from the dataset that have an identical barcode. As proven above on a simple model, two reads coming from the same fragment will be linked with much greater probability than two reads coming from two different fragments.

While testing on the simulated human dataset, we saw that repeated regions could create false positive links in the graph. Indeed, a read containing a repeated region will share k-mers with all the reads containing this repeated region, including all those that are actually far away on the genome. The repeated k-mers are present in much more reads than average k-mers, creating hubs of connections and many false positive links in the graph. Un-indexing k-mers present more than twice the average number of times in the dataset improves greatly the quality of the graph and of the results of QuickDeconvolution. [10] justifies this operation with a parallel to removing stop words in natural-language-processing (NLP).

3.4 Graph clustering

The read graph thus need to be clustered into an unknown number of enhanced read clouds. QuickDeconvolution uses the *Chinese whispers* algorithm, a clustering method introduced in NLP research [20]. It works as follow: at the beginning of the algorithm, each read is contained into its own cluster of size 1. The reads are then processed in a random order for a small number of iterations. Each read inherits the cluster that is seen most often among all the neighbors, pondered by the strengths of the links (in case of multiple equal possibilities, one is chosen randomly). This algorithm is known to converge quickly towards few stable clusters, especially if the diameter of the graph is small (*i.e.* any two vertices are separated by few edges), which is generally the case in our read graphs. In worst-case scenarios the clustering can oscillate, but this is marginal in practice for QuickDeconvolution.

This clustering method has the great advantage of being parameter-free and agnostic regarding the final number of clusters.

This graph clustering method is a novelty compared to what had been done in Minerva. Minerva deleted links weaker than a certain threshold on the read graph and then considered the different connected components of the graph as distinct cloud. David Danko *et al.* showed that the method could work to separate fragments coming from different species in metagenomic samples [10]. However, when trying to separate fragments coming from one genome, we have found on several simulations that it was hard to totally separate clusters because of the redundancies and the small repeated elements present across a genome. It even became harder and harder as the depth of sequencing or the quality of the reads diminished. Hence we opted for a slightly costlier but more flexible approach that allowed for residual false positive links. Knowing that the clustering step will compensate some errors, we could implement the shortcuts to make the graph building step faster.

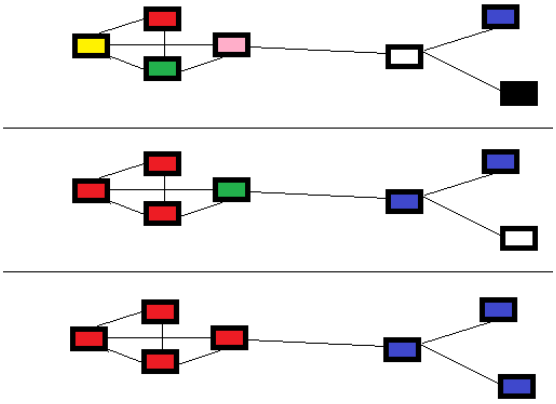


Figure 8: Figure illustrating the principle of the chinese whispers algorithm from Wikipedia. Classes are represented by colors. At the beginning of the algorithm each node (read) has its own class (top). The middle drawing corresponds to the situation after a full iteration through all nodes. At the bottom, after a supplementary iteration. The algorithm has converged: no node will change color with further iterations.

Finally, all reads of a cloud that are not linked to the graph are not clustered separately but are marked with a special '0' tag, to indicate to the user that the clustering has been inefficient there. This can happen when a read is too noisy to be overlapped with something else or if the read finds itself in a very repeated region and all its k-mers are un-indexed.

3.5 Parallelization

One of the objectives of the internship was to build a parallelizable algorithm which could exploit the multiple cores present in computers and in particular on computing clusters. The goal of the parallelization is to dispatch the work between the different threads. A perfectly parallelized program is able to run t times faster when t threads are available.

The algorithm runs in four distinct phase: loading the data from the input file, creating the dictionary, deconvolving the read clouds, and writing data to an output file. The first and last phase being negligible in time compared to the other two, they are not parallelized at all and handled by a single thread.

The third phase, where each graph is built and clustered, is trivially parallelizable. All threads can handle separate clouds, build their graphs, cluster them and store the result. The threads only compete for access to the dictionary, which is not copied t times to keep reasonable RAM usage.

Building the dictionary is the hardest phase to parallelize. Indeed, threads cannot just dispatch the reads between them: if the same k-mer is found on two threads at the same time, the threads cannot update the dictionary simultaneously (if two threads write in the same entry at the same time, one entry will

probably be overwritten). If the k-mer was unseen before, this could even make the program crash. The trick is to dispatch the k-mers between the threads: for example, thread 1 deals with k-mers ending with A or C while thread 2 deals with k-mers ending with G and T. This means that each thread needs to look at all reads, but does not index all k-mers. To avoid recomputing in each thread the set of sparse k-mers, this computation is done beforehand for each read, parallely. We end up with two sub-phases: first the threads dispatch the reads between them and compute all the sparse k-mers; once this is done, the threads dispatch the k-mers between them and go through all the reads to index them.

4 Discussion

QuickDeconvolution is a new tool outperforming all the state-of-the-art software in term of reference-free barcode deconvolution.

On the dataset where comparison was possible, QuickDeconvolution proposed a deconvolution for almost all reads, unlike Minerva and Ariadne. On the real *Drosophila* dataset, QD proposed a deconvolution for 81% of the reads, signalling to the user when the fragment of origin could not be determined with certainty. We chose not to rescue the un-clustered reads, because the risk of mis-clustering would have been too high: QuickDeconvolution is implemented to favor accuracy over completeness.

In terms of precision, QuickDeconvolution was very satisfactory, once again outperforming other tools. It effectively separated the fragments present within one barcode, without fracturing the barcodes in many small clouds on all datasets we tested with the solutions. Compared to Minerva, the introduction of a clustering algorithm allows QD to sacrifice a little precision over speed in the graph-building phase, while improving final deconvolution.

In term of speed, QuickDeconvolution outperforms its competitors. At least up to 16 threads, the parallelization is very good, making it many times faster than the other solutions as soon as the number of thread exceeds 2. At the same time, QuickDeconvolution does not consume more RAM than Minerva or Ariadne. It does not produce any intermediary files, that are limiting for Ariadne. QD thus overcome the main shortcoming of Minerva and Ariadne, the difficulty to handle big datasets. Deconvolution of an animal sequencing experiment is demonstrated here for the first time. Consequently, QuickDeconvolution may help in the future improve all linked-reads applications involving long genomes, for example variant calling [21].

The main limit of QuickDeconvolution today is its consequent RAM usage. Generally, RAM usage depends on the total number of k-mers indexed, *i.e.* the size of the dataset, the quality of the reads and the length(s) of the sequenced genome(s). On the *Drosophila* genome dataset we tested, the RAM usage was over 1000G with 16 threads. The size of the dataset was 33 Gb. We could easily imagine bigger datasets: for example, a 50-fold coverage of a diploid human would generate approximately 300 Gb of data. To give an order of magnitude, the maximum amount of RAM accessible on the Genouest cluster in Rennes is

3000 G, setting a limit to the effective size of the datasets QuickDeconvolution can deconvolve. If the dataset is too big for QuickDeconvolution, the only strategy left for now is to deconvolve using a reference genome. The next logical development for QD would be to develop a more compact index to overcome this limit.

Acknowledgments

We acknowledge the GenOuest bioinformatics core facility (https://www.genouest.org) for providing the computing infrastructure.

Many thanks to Pierre Morisse for providing the human dataset and helping with downstream analyses.

More generally, I want to thank all the Symbiose teams, who provided a very welcoming work environment.

References

[1] Avery, O. T., MacLeod, C. M. & McCarty, M. Studies on the chemical nature of the substance inducing transformation of pneumococcal types: induction of transformation by a desoxyribonucleic acid fraction isolated from Pneumococcus type III. *Journal of Experimental Medecine* **79**, 137–158 (1944).

[2] International Human Genome Sequencing Consortium. Initial sequencing and analysis of the human genome. *Nature* **409**, 860–921 (2001).

[3] Sanger, F., Nicklen, S. & Coulson, A. R. DNA sequencing with chain-terminating inhibitors. *Proceedings of the National Academy of Sciences* **74**, 5463–5467 (1977).

[4] Margulies, M. *et al.* Genome sequencing in microfabricated high-density picolitre reactors. *Nature* **437**, 376–380 (2005).

[5] Mardis, E. R. Next-generation DNA sequencing methods. *Annual Review of Genomics and Human Genetics* **9**, 387–402 (2008).

[6] Deamer, D., Akeson, M. & Branton, D. Three decades of nanopore sequencing. *Nature Biotechnology* **34**, 518–524 (2016).

[7] Korlach, J. *et al.* Real-time DNA sequencing from single polymerase molecules. *Methods in Enzymology* **472**, 431–55 (2010).

[8] Payne, A., Holmes, N., Rakyan, V. & Loose, M. Bulkvis: a graphical viewer for oxford nanopore bulk fast5 files. *Bioinformatics* **35**, 2193–2198 (2019).

[9] Delahaye, C. Nanopore minion long read sequencer:an overview of its error landscape. In *SeqBim* (2020). URL `https://seqbim2020.sciencesconf.org/data/pages/SeqBIM_2020_Clara_Delahaye_1.pdf`.

[10] Danko, D., Meleshko, D., Bezdan, D., Mason, C. & Hajirasouliha, I. Minerva: An alignment- and reference-free approach to deconvolve linked-reads for metagenomics. *Genome Research* **29**, gr.235499.118 (2018).

[11] Mak, L. *et al.* Ariadne: Barcoded linked-read deconvolution using de bruijn graphs. *BioRxiv* (2021).

[12] Wang, O. *et al.* Efficient and unique co-barcoding of second-generation sequencing reads from long dna molecules enabling cost effective and accurate sequencing, haplotyping, and de novo assembly. *Genome Research* **29**, gr.245126.118 (2019).

[13] Zheng, G. *et al.* Haplotyping germline and cancer genomes with high-throughput linked-read sequencing. *Nature Biotechnology* **34** (2016).

[14] Chen, Z. *et al.* Ultra-low input single tube linked-read library method enables short-read ngs systems to generate highly accurate and economical long-range sequencing information for de novo genome assembly and haplotype phasing. *BioRxiv* (2019).

[15] Hulse-Kemp, A. *et al.* Reference quality assembly of the 3.5-gb genome of capsicum annuum from a single linked-read library. *Horticulture Research* **5**, 4 (2018).

[16] Shajii, A., Numanagić, I. & Berger, B. Latent variable model for aligning barcoded short-reads improves downstream analyses. *Research in computational molecular biology : ... Annual International Conference, RECOMB ... : proceedings. RECOMB (Conference : 2005-)* **10812**, 280–282 (2018).

[17] Prjibelski, A., Antipov, D., Meleshko, D., Lapidus, A. & Korobeynikov, A. Using spades de novo assembler. *Current Protocols in Bioinformatics* **70** (2020).

[18] Luo, R., Sedlazeck, F., Darby, C., Kelly, S. & Schatz, M. Lrsim: a linked reads simulator generating insights for better genome partitioning (2017).

[19] Li, Z. *et al.* Comparison of the two major classes of assembly algorithms: Overlap-layout-consensus and de-bruijn-graph. *Briefings in functional genomics* **11**, 25–37 (2011).

[20] Biemann, C. Chinese whispers: An efficient graph clustering algorithm and its application to natural language processing problems. *Proceedings of TextGraphs* 73–80 (2006).

[21] Morisse, P., Legeai, F. & Lemaitre, C. Leviathan: efficient discovery of large structural variants by leveraging long-range information from linked-reads data (2021).