Towards Accurate De Novo Assembly for Genomes with Repeats

Doina Bucur
University of Twente, The Netherlands
d.bucur@utwente.nl

***Abstract*—De novo genome assemblers designed for short k-mer length or using short raw reads are unlikely to recover complex features of the underlying genome, such as repeats hundreds of bases long. We implement a stochastic machine-learning method which obtains accurate assemblies with repeats and self-validates assemblies via consensus. For this, a prior assembler is extended with the ability to (a) assemble *variable-length raw reads*, which may span and unambiguously recover *interspersed repeats* in the genome, and (b) recognize *long, direct terminal repeats* during the assembly, then report an unambiguous *circular assembly*. Consensus is obtained via stochastically independent runs of the assembler on the same read library. We experiment on viral and mitochondrial genomes of up to 41 kbp, with synthetic raw-read libraries, to be able to evaluate the assembly against a reference. We show the prerequisites for obtaining accurate assemblies. For genomes with interspersed repeats, using raw reads of average length comparable to the repeat length likely gives an accurate genome. Genomes with long direct terminal repeats can be assembled accurately also with reads shorter than the repeat length. In both cases, a simple majority forms consensus, since over 70 % of independent runs on this set of genomes yield a correct assembly.**

I. Introduction

Although sequencing methods have improved, *de novo* assembly remains challenging for two reasons: (1) the underlying problem of fragment assembly into a shortest common superstring (SCS) is computationally NP-hard with the size of the read library even without the added complications of double-strandedness, read errors, and non-uniform coverage [1], and (2) for complex genomes with repeats, formulating the problem as finding the SCS using short-read libraries leads to overcompression. Inaccurate assemblies become an issue when attempting to do comparative genomics: although the genomes of many more species are available, the fraction of a genome that is accurate is below 80 %; there is an imbalance between the quantity and the quality of genomes [2].

Limited algorithmically by the short k-mer length, de-Bruijn/Eulerian-path assemblers for reads of any length are unlikely to recover complex features of genomes, because the k-mer graph construction loses the connectivity information given by long reads. Current de-Bruijn assemblers such as SOAPdenovo [3] and Velvet [4] attempt to recover repeats by reintroducing long reads into the picture, and heuristically (e.g., greedily) guiding the path construction through the graph to match the reads. However, greedy heuristics come with no guarantee as to the optimality of the result, and thus de-Bruijn assemblers tend to have low accuracy on complex genomes. While overlap-layout-consensus assemblers such as Celera [5] are designed to use long-read information natively, computing the optimal path through the overlap graph is a computationally hard task, so in practice it is broken down hierarchically, with analysis steps done again by suboptimal greedy heuristics. In studies such as an assembly of a human genome, Celera "lost" 99.1 % of duplicated sequences, and produced a genome 16.2 % shorter than the reference [2].

Summary of contribution. This study shows a prototype assembler for variable-length raw reads, which can recover long interspersed repeats, either direct or inverted (if the raw reads are also sufficiently long), and deals with direct terminal repeats by constructing an unambiguous circular assembly. The tool implements a computationally intensive assembly method based on our prior *genetic algorithm* for the SCS problem [6]. This metaheuristic, inspired by natural selection, stochastically machine-learns a good permutation among the library reads in input, by maintaining a *population* of intermediate, part-assembled scaffolds, and iteratively *selecting* the best scaffolds and improving these partial solutions into even shorter scaffolds. This improvement is done using random *mutation* and *crossover* operators over permutations of reads, so that an already good partial scaffold may become a better one by moving, reverse-complementing, or merging contigs.

While also built around a heuristic, like all other existing assemblers, this metaheuristic allows the exploration and comparison of a large number of assembly possibilities, with choices based on random draws and the overall objective of achieving a short, if possible contiguous, assembly, rather than by making suboptimal greedy choices. It is this stochasticity which gives this metaheuristic a degree of objectivity: (a) it enables accurate assemblies due to exploring different assembly scenarios in a single algorithm, and (b) multiple stochastically independent runs of the algorithm on the same library (easily run in parallel) will serve to confirm the final, correct form of the genome, thus forming consensus.

We evaluate this heuristic on viral and mitochondrial genomes of up to 41 kbp, over synthetic raw-read libraries, and compare the consensus assembly against the original genome. Interspersed repeats are recoverable if the raw reads have average length comparable to the repeat length. Genomes with long direct terminal repeats can be assembled accurately also with reads shorter than the repeat length. In both cases, a simple majority vote gives the consensus genome.

978-1-4673-8988-4/17/$31.00

II. Related work: resolving repeats with current assemblers

We list known facts about the ability of existing industrial-grade assembly algorithms to reconstruct repeated sequences in genomes. For an overview of prior software tools solving the general fragment-assembly problem using machine-learning methods based on genetic algorithms, see [6]. None of these caters for genomes with repeats.

The computationally simplest, *greedy assemblers*, such as SSAKE [7], have runtimes polynomial in the size of the read library. Error-free reads are joined into contigs iteratively, starting with the reads with longest contiguous overlap. SSAKE and other recent greedy assemblers apply the greedy iteration starting with an unassembled read, and attempt to construct a strand in one direction; the assembly terminates when a conflict is found, i.e., multiple reads which could extend the current contig, but which themselves do not overlap with each other. Even in the absence of genomic repeats, there is no guarantee that this heuristic obtains a global optimum. In the presence of repeats, misassemblies are likely, and manifest into a fragmented assembly [8].

Assemblers based on the *overlap-layout-consensus* (OLC) three-step routine aim to compute the global SCS in an overlap graph where raw reads form the nodes, and their pairwise overlaps become weighted edges; it is known that a path of maximum weight in this graph provably gives a minimum-length assembly [1], so doing layout means computing a Hamiltonian path in the overlap graph, which is NP-hard. In practice, a hierarchical approach is taken to scale down the layout step: Celera [5] first assembles reads into unambiguous uniquely assemblable contig (unitigs), essentially by following linear, non-branching subgraphs in the overlap graph. This step can result in the collapse of multiple repeat copies into a single unitig, and further heuristic analyses of the graph to join the unitigs are guided by error-prone coverage information, or by precise mate-pair information.

De-Bruijn/Eulerian-path assemblers build an overlap graph in which the nodes are the unique strings (k-mers) of length k present in the raw reads, and an edge is drawn between k-mers overlapping by $k-1$ bp; an Eulerian path is then computed to traverse all edges, and this computation is polynomial. Unfortunately, the presence of repeats will interfere with this logic: an exponential number of distinct Eulerian paths may exist, as per the many different ways the genome can be shuffled around its repeats [8]; essentially, basing the calculation on the de-Bruijn graph means a loss of the long-range connectivity information given by long reads. Guiding the path search towards the optimal path leads to a hard computational problem, equivalent to an OLC approach. De-Bruijn assemblers are thus suitable for short reads. SOAPdenovo [3] performed comparatively well on a bacterial genome, yielding contigs that were eight times longer than those it constructed for a human genome [9]; it uses the `-R` flag to attempt to resolve repeats by using the original raw reads. Velvet [4] uses supplementary long reads (`-long, -exp_cov`) to greedily confirm the most likely path through the graph (a long path between two nodes is confirmed if a majority of long reads follow it), and fill in any gaps in the string assembly in the process.

Either due to the assembly algorithm, or to the quality of the reads, no assembler is known to perform accurately and deliver correct genomes with repeats. Instead, errors such as repeat overcompression must be located post-assembly and corrected using additional information, such as coverage information, or additional libraries of long reads or mate-pair reads with long insert length. Longer read lengths with high read accuracy can be obtained with the third-generation Illumina TruSeq Synthetic Long-Read Sequencing method, which partly assembles high-quality short reads into higher-quality long consensus sequences, with corrected PacBio reads, or with Oxford Nanopore reads.

Even with accurate TruSeq long reads, both coverage-related and algorithmic difficulties remain. A TruSeq library for a *C. elegans* genome had a median length around 6 kbp, and a SNV ratio (at around 0.6 %) much lower than the standard error rate of Illumina raw reads [10]; the worst coverage of a repeat region was for tandem repeats whose cluster length was above 500 bp. Despite the accuracy of the long reads, performing a de-novo assembly with the OLC assemblers Celera and MIRA produced hundreds of contig misassemblies. The authors call for better assemblers for long reads. The Celera assembler also used TruSeq long reads to assemble a *D. melanogaster* genome, and could not reconstruct 22.2 % of the transposable elements, nor 15.8 % of the genes [11].

Recent comparative studies [9], [12], [13] provide an overview of the performance of many commercial-grade assemblers with both short- and long-read, single- and paired-end libraries. All studies found "chaff" contigs, duplicated contigs, contig misjoins, and compressions of repeat sequences, to the conclusion that a single assembler and set of assembler parameters cannot be trusted; instead, *consensus* assemblies should be calculated by using multiple assembly algorithms and parameter settings.

III. Method: genomes and read libraries

This section and Section IV give the assembly methodology; this section describes the biological data sets we used, while Section IV summarizes the algorithmic method.

We obtained variable-length read libraries synthetically from known genomes with repeat regions annotated in the NCBI database; the original genome then serves as the ground truth against which the new assembly is precisely evaluated.

A. Genomes

Table I lists the genomes assembled here and their most significant repeats, in terms of size of the repeat cluster. The genomes have been selected via (a) NCBI nucleotide search runs for complete genomes, using groups of keywords related to the type of repeat targeted, i.e., *direct*, *inverted*, *tandem*, *interspersed*, and *terminal*, and (b) textbook references to

Table I: Genomes assembled.

Species	Acc. no.	Ref.	Genome size (bp)	Repeat type (size in bp)	Main repeat
Hepatitis C virus (**HCV**)	NC004102	[14]	9646	tandem mono-DIR (51, 17, 20)	DIR
Ciconia mitochondrial DNA (**mtDNA**)	AB026818	[15]	17347	tandem tetra-DIR (240), tandem DIR (310)	
Simian virus 40 (**SV40**)	J02400	[16]	5243	tandem DIR (144)	
Frog adenovirus (**FrAdV**)	NC002501	[17]	26163	DIR (609), ITR (36)	
Human immunodeficiency virus 1 (**HIV-1**)	JQ316128	[18]	9257	DTR (142)	DTR
Adeno-associated virus (**AAV**)	AF043303	[19]	4679	DTR (125), ITR (43) + IIR (63)	
Proteus phage (**PM16**)	KF319020	[20]	41268	DTR (450)	

classic features of viral RNA or double- or single-stranded DNA genomes, such as the potential for circularity using direct terminal repeats. In Table I, the abbreviations for repeat types are: *direct interspersed repeat* (DIR), *direct terminal repeat* (DTR), *inverted interspersed repeat* (IIR), and *inverted terminal repeat* (ITR). Tandem repeats are adjacent repetitive regions of the genome, and short repetitive tandem sequences are denoted as, e.g., mono-DIR (when the repeat contains a single base, e.g., `T` in the case of the mono-DIR of the HCV strain in the table).

Table I mostly lists the size of perfect contiguous repeat sequences, as follows.

a) Direct interspersed repeats: The repeats in **HCV** are contained in the longer imperfect sequence `T..TCT..TCC T..TCTTTCCTTCTTTTTTCCTTTCTTTTCCTTCCTTCTTT`, of which the sizes of the three perfect tandem `T` mono-DIRs (partly elided from the sequence above) are between 50 and 17 bp. Very short such microsatellites are common in Hepatitis C viruses, and may have a functional effect in the evolution of the genomes, due to the high mutability of this type of repeat [21]. Similarly, the microsatellites in the **mtDNA** molecule are part of a `CAAA` and `CAA` repeat sequence of 240 nucleotides, located in the displacement loop of the control region of the circular molecule [15].

The **mtDNA** molecule has another tandem DIR, also in the control region, with a longer, 71-bp repeat (adjacent to itself more than 4 times, with only localized imperfections) [15]. The longest perfect repetitive sequence in the **SV40** molecule is 72 bp, repeated twice in tandem, with a role in enhancing molecule transcription [16].

The adenovirus **FrAdV** has a short perfect ITR of 36 bp, but more importantly two instances of a long perfect DIR of 609 bp near the right end, likely due to this genome having been derived by the recombination of a slightly shorter genome without long repeats [22].

b) Direct terminal repeats: The **HIV-1** genome contains long DTRs, and is thus prone to forming circular molecules; circular HIV-1 DNA is known to be a by-product of HIV-1 genome transcription [23]. This strain has a perfect 142-bp DTR. The ssDNA molecule of the adeno-associated virus **AAV** is annotated on NCBI [19] with an ITR of 145 bp. However, this has imperfections, and a more precise description of these terminal repeats is as imperfect palindromes, including two perfect shorter inverted repeats at the ends of this imperfect ITR (of which one repeat is terminal), both overlapping with a perfect DTR of 125 bp. The table lists these perfect repeats. The DTR-in-ITR terminal structure of the **AAV** was found to enable an intermediate circular form of the genome, similar to circular genomes in other viruses [24].

B. Synthetic read libraries

Given a reference genome, reads are randomly sampled, so that a read library obtained satisfies the following conditions:

- All reads have random variable lengths in a given interval. We test multiple intervals, with average read length around the length of the main repeat type in the genome.
- The reference genome is covered by the read library without gaps.
- The average genome coverage (i.e., the sum of all read lengths divided by the genome size) is at least equal to 10. No other coverage constraint is set: the actual coverage across the genome length will vary widely, including the coverage of the repeat sequences.

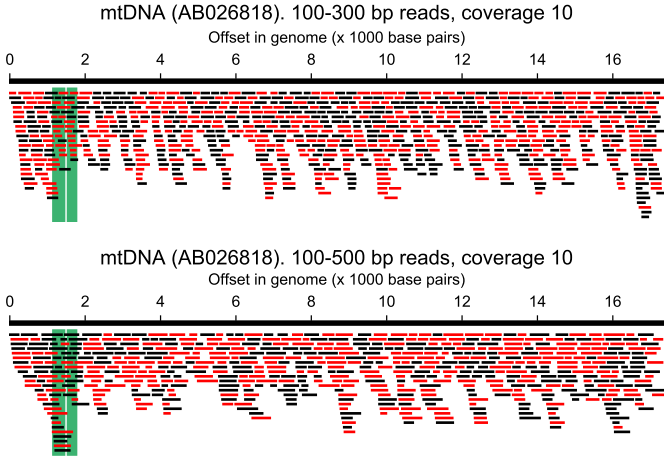

Figure 1: Two read libraries of average coverage 10, but with different read-length intervals for the **mtDNA** genome. The length of the genome is shown as a contiguous black line at the top. The two tandem DIR regions are marked in green. The reads are shown in two colours: black for forward reads, and red for reads which are reverse-complemented.

Figures 1 and 2 show two read libraries with different read-length intervals, for each of two genomes (**mtDNA** and **FrAdV**). The shorter the average read length, the less likely the library is to span the repeats.

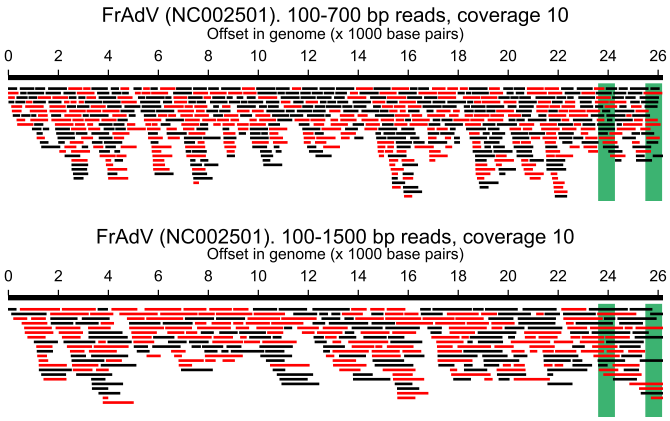

Figure 2: Two read libraries of average coverage 10, but with different read-length intervals for the **FrAdV** genome. The length of the genome is shown as a contiguous black line at the top. The two copies of the DIR are marked in green. The reads are shown in two colours: black for forward reads, and red for reads which are reverse-complemented.

No artificial faults (single nucleotide variations or indels) are added to the read library in this study; this idealized evaluation setting serves this stage of tool development to reveal the strengths and weaknesses of the core assembly method.

Each genome is assembled in trials from 5 read libraries sampled randomly with a given read-length interval from the same genome, to avoid drawing conclusions based on a sampling bias. Furthermore, read libraries are sampled with multiple read-length intervals, to learn the extent to which read lengths improve the assembly process using this method.

IV. Method: Assembly Algorithm

The assembler is a recently redesigned computational method based around stochastic optimization with a gradient descent [6]. It "learns" the order (and sense, or strand orientation) in which the raw reads are assembled into contigs using an optimization method called a *genetic algorithm*: essentially an iterative soft computation in which every subsequent iteration attempts to improve the scaffold obtained thus far via operators inspired by natural evolution, under the guidance of a *fitness function*. We summarize the method in this section, and refer to [6] for the remaining technical details. Compared to the assembler in [6], this version allows for variable-length reads, adds the new Align mutation, and the ability to perform circular assemblies (the latter two described below).

A. Linear assembly

We call by *candidate solution* for the assembly a *segmented permutation* of the variable-length reads in the library. A simple permutation such as $[r_1, r_2, r_3]$, where r_1, r_2, r_3 are raw reads, dictates the order in which these three reads are adjoined into a contig; in other words, the offsets at which these reads start in the contig are monotonically non-decreasing positive integers. A segmented permutation adds markers for contig bounds, i.e., the segmented permutation $[r_1, r_2, r_3 \mid r_4]$ is a scaffold with two contigs assembled from four reads. An example is shown in Figure 3.

Figure 3: A simple example of a candidate solution; this scaffold has three contigs. The reads are shown in two colours: black reads which are assembled into a contig in the forward sense, i.e., the same sense of the read as found in the read library, and red for reads which are reverse-complemented in order to be assembled at that position on the scaffold.

The genetic algorithm maintains a "population" of such candidate solutions for the assembly; the initial population is generated randomly, and each of these first candidate solutions consists of single-read contigs, with these contigs ordered randomly on the scaffold of each assembly. At each iteration of the algorithm, the candidate solutions in the population are evaluated with a fitness function, and the best among these will be selected, modified and combined into the next "generation" of solutions. Here, a practical choice for the fitness function is the sum of:

- the number of contigs on the scaffold of the solution, and
- the string length of the scaffold.

A better candidate solution will have lower fitness. This choice of fitness function suits de-novo assembly (which lacks a reference genome), provided that the raw reads span the interspersed genomic repeats; otherwise, it would promote contig overcompression.

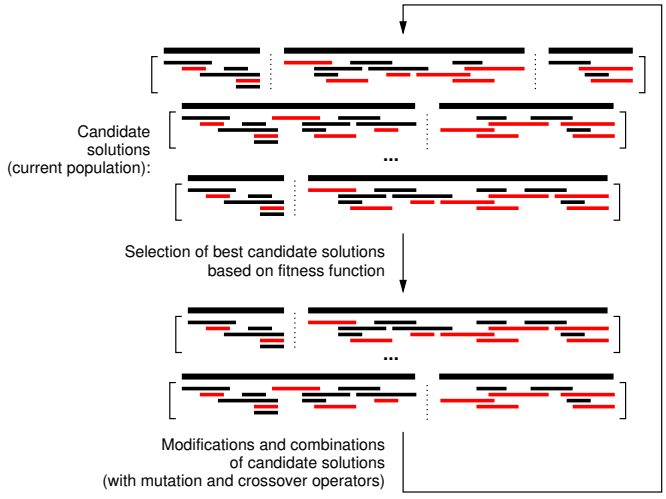

Figure 4: A schematic representation of the iteration loop in the genetic algorithm.

After selection, the candidate solutions are modified randomly using *mutation* and *crossover* operators (listed below with their abbreviations). Figure 4 shows a simplistic iteration of the genetic algorithm.

Mutation or crossover operators are applied to any selected candidate solution in a given order. The operators are summarized below; for a full description see earlier work [6].

- **R** (*Reverse-complement*) A randomly selected contig (and its internal reads) are reverse-complemented.
- **S** (*Split*) A randomly selected contig (of at least two reads) is split at a random point into two contigs.
- **I** (*Inshift*) A randomly selected contig is moved to another position on the scaffold.
- **M** (*Merge*) Two adjacent contigs on the scaffold are merged, if they overlap above a minimum.
- **A** (*Align*) To reduce "chaff", a read extracted from a short contig is realigned completely onto a long contig (of length above a configurable *threshold*, given as a fraction of an estimated optimal assembly size). **A** is new compared to [6].
- **O** (*Order crossover*) Two random "parent" candidate solutions are combined, yielding two "offspring" candidate solutions.

B. Circular assembly

This linear method of minimizing the assembly length, when applied to a genome with long DTRs, naturally leads to assemblies in which the DTRs are overlapped, and the linear assembly reported is shorter in length, and has much shorter DTRs. In essence, this result is correct. Figure 5 shows the equivalence between the linear and circular forms of genomes with long DTRs: the shortest assembly for the linear genome with long DTRs (top left) is the circular genome (bottom). The shortest linear assembly will have the shape at the top right.

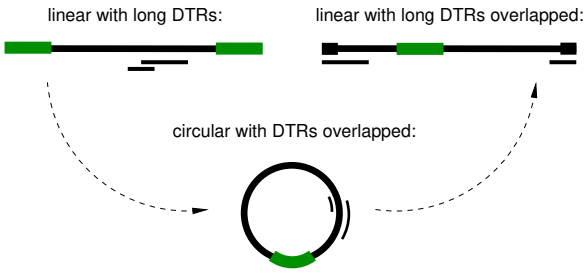

Figure 5: Equivalent linear and circular forms for genomes with DTRs. DTRs are shown in green; only 2 raw reads are depicted alongside the remaining of the genome. The shortest assembly is the circular form.

Our assembler has a "circular" mode, which, if set, will verify whether a single-contig linear scaffold has DTRs of *any length* above the minimum overlap length for adjacent reads (in this study, 20 bp); if a DTR is found, then the assembly is reported as circular, as a string genome plus two integers: the length of the circular genome, and the length of a linear equivalent. The circular length will be constant regardless of the way in which the circular genome was linearized; this fact helps the algorithm converge on a consensus genome in circular mode.

C. Experimental settings

Table II lists the values assigned to the various parameters of the experimental campaign. 100 candidate solutions are refined for every population; to construct the next population, the solutions in the current population are selected via tournament with rounds of size 5: the solution with the best fitness value among all is selected. After selection, a sequence of operators (here, **RASIMOM**) is applied to all solutions. The sequence is chosen based on prior experimentation [6]; essentially, it intersperses those operators which directly attempt to shorten the scaffold (**M** and **A**) with the remaining operators, which will modify the scaffold but are not able to shorten it. Other operator sequences may also work equally well.

Table II: Parameters for the genetic algorithm.

	Value
Population size	100
Termination condition	100 stagnating generations
Selection	Tournament selection, size 5
Operator order	**RASIMOM**
R, S, I, M mutation rates	0.5
A mutation rate / threshold	1 / 0.25
O crossover rate	0.1
Read libraries per test case	5
Assembly runs per read library	20
Minimum overlap (bp)	20

Each operator has a configurable rate (probability) of application. This rate is simply set at the middle value of 0.5 for most mutation operators (**R, S, I, M**), is a very low 0.1 for the crossover **O** (an operator which changes scaffolds drastically), and is maximum for the mutation operator **A** (whose application is the most beneficial among all operators).

Finally, as this computational method is highly stochastic, each of the 5 read libraries sampled from a genome using particular read lengths is assembled 20 times in parallel, and the performance of the method is evaluated statistically using this sample of 100 runs.

V. Experimental results

We evaluate this assembler against strict accuracy metrics: we require that zero misassemblies occur, i.e., that the resulting assembly is a single-contig, accurate, linear or circular genome, depending on the main repeat type. This strong requirement is aided by the idealized evaluation setting with error-free reads, and by having a reference genome to compare against.

We execute the 20 stochastically independent assembly runs, for each of 5 read libraries generated for each given read-length interval, in parallel on a multi-core server with 3-GHz computing cores. By varying the read-length interval, we find empirically that, for this assembly algorithm, the accuracy of linear assemblies varies with the average read length for genomes with DIRs (both tandem and interspersed), but is affected less by the read length when attempting circular assemblies for genomes with long DTRs. Accurate assemblies, when starting from reads of suitable length, are obtained in a large majority of the runs.

In this section, we first give numerical results across assembly runs in order to quantify the likelihood of obtaining accurate genomes, then discuss consensus, runtimes, and make a performance comparison with 3 other assemblers.

A. Likelihood of accurate assemblies

We first generate 5 read libraries per two read-length intervals per genome. The intervals are bounded by "round" read lengths (most often 100, 150, 300, 500, or 1000 bp), chosen so that the length of the main repeat in the underlying genome is either slightly below or slightly above the average length in the read libraries. For example, for the **FrAdV** genome, the read length lies between 100 and 1000 bp in 5 libraries, with an average slightly under the 609 bp DIR length in that genome. After executing the 20 independent assemblies for each of the 5 read libraries, the end result of each is compared against the reference genome. We then report, for each read library in input, three metrics: (a) the lowest contig count computed for a scaffold; (b) in the case when the best scaffold was single-contig, whether this contig is an accurate assembly (✓), i.e., a forward or reverse-complemented, linear or circular copy of the reference, or not (✗); (c) what fraction of the 100 assemblies obtained an accurate assembly. These metrics are listed in Table III, together with comparative runs of other assemblers.

With the exception of assemblies of the **mtDNA** genome from reads of length 100 to 300 bp, the algorithm obtained accurate single contigs for all the libraries in Table III in a large majority of cases. The **mtDNA** case (where the genome contains DIR sequences of lengths 240 and 310 bp, as per Table I) is attributable to the low average length in those read libraries (200 bp, also with no read longer than 300 bp) compared to the DIR length. To gain more insight into these performance factors, in Figures 6 and 7 we plot the likelihood of assembling the correct genome separately, for the 4 genomes where the main repeat is a DIR, and the 3 genomes where that is a DTR. For this, 3 more experimental settings are included, in addition to those listed in Table III, in order to explore more extreme-case average read lengths. These added assemblies are: **FrAdV** with read lengths 100-700 bp, **HIV-1** with read lengths 100-300 bp, and **AAV** with read lengths 75-100 bp.

In both figures, the colour of a data point gives the likelihood (in percentage out of 100 independent assembly runs) that the correct genome is obtained; the size of a data point is proportional to the size of the underlying genome, and the data is plotted against axes which measure the difference between the average read length in the library and the repeat size (the x axis) and the size of the read library (the y axis). The likelihood of an accurate genome is expected to rise with positive x values, and decrease with high y values, since larger libraries pose more challenges to the algorithm.

This hypothesis is confirmed, on our genome set, for genomes with DIRs, where data points with positive x values are accurate in over 80 % of the runs (Figure 6). The exceptional **mtDNA** assemblies with low average read length is the

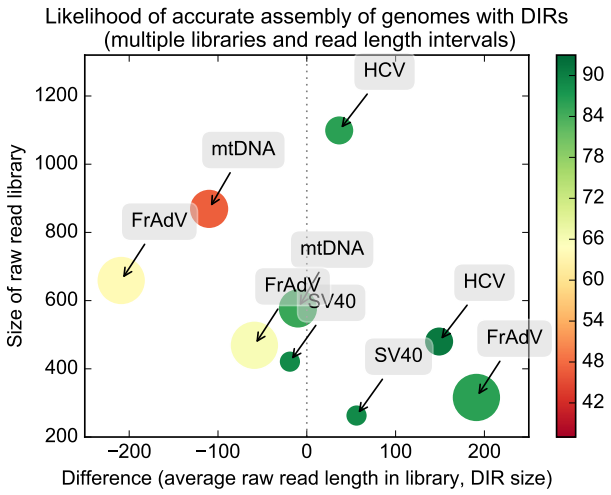

Figure 6: The likelihood of accurate assemblies (shown in colour) for genomes with DIRs. The diameter of a data point is proportional to the size of the genome.

sole case here which failed to obtain an accurate majority; nevertheless, 47 assemblies out of 100 were correct also in that case. We can conclude that parallel runs of the algorithm could build a decisive consensus when assembling genomes with DIRs from libraries with read lengths comparable to the DIR length.

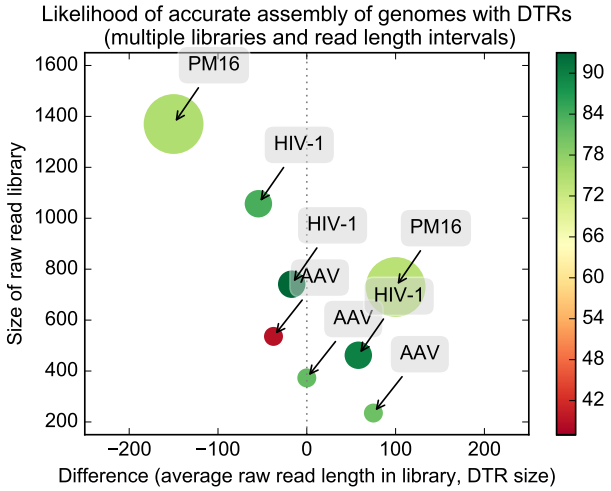

Figure 7: The likelihood of accurate assemblies (shown in colour) for genomes with DTRs. The diameter of a data point is proportional to the size of the genome.

On the other hand, assembling genomes with long DTRs (and only short DIRs present) in circular fashion is less of a challenge for the algorithm, as expected from the design of the circular assembly. All libraries are correctly assembled in at least 70 % of the runs, regardless of the read length, with a single exception: **AAV** assembled from short reads of 75-100 bp. This case is attributable to the fact that, DTR aside, this

Table III: Summary of results, and comparison with the assemblers SSAKE [7], SOAPdenovo [3], and Velvet [4] (the latter two using the option to resolve repeats using long reads). All read libraries have average coverage 10.

				This assembler		SSAKE	SOAPdenovo		Velvet	
Species	**Size (bp)**	**Read length (bp)**	**Lib. size**	**Min contig count**	**% accurate genome**	**Min contig count**	**Min contig count**	**Min-max / N50 contig size (bp)**	**Min contig count**	**Min-max / N50 contig size (bp)**
HCV	9646	75-100	1099	1 ✓	**86 %**	14	11	14-5364 / 5364	3	748-5974 / 5974
		100-300	480	1 ✓	**91 %**	3	14	14-4019 / 1749	1 ✗	9581
mtDNA	17347	100-300	869	1 ✓	**47 %**	8	6	170-5589 / 4720	4	71-15579 / 15579
		100-500	577	1 ✓	**85 %**	4	7	53-15776 / 15776	2	71-17317 / 17317
SV40	5243	100-150	421	1 ✓	**89 %**	3	2	121-5036 / 5036	3	42-4988 / 4988
		100-300	263	1 ✓	**89 %**	3	4	42-5009 / 5009	2	72-5213 / 5213
FrAdV	26163	100-1000	469	1 ✓	**66 %**	2	90	78-731 / 296	2	1888-23624 / 23624
		100-1500	316	1 ✓	**86 %**	2	99	16-624 / 196	2	1888-23624 / 23624
HIV-1	9257	75-100	1057	1 ✓	**84 %**	2	6	670-5284 / 5284	1 ✓	9115
		100-150	741	1 ✓	**93 %**	2	8	386-3045 / 1067	1 ✓	9115
AAV	4679	100-150	373	1 ✓	**82 %**	2	5	14-2596 / 2596	1 ✓	4554
		100-300	235	1 ✓	**81 %**	2	1 ✗	4580	1 ✓	4554
PM16	41268	100-500	1370	1 ✓	**75 %**	2	72	14-2504 / 905	1 ✓	40818
		100-1000	730	1 ✓	**74 %**	2	162	65-748 / 242	1 ✓	40818

genome also contains interspersed repeats in close proximity to terminal repeats, and the total length of this sequence surpasses that of any read in the library.

B. Forming consensus

In principle, a few runs of the assembler on a suitable read library are likely sufficient to recover the genome. However, a high accuracy rate doesn't mean hard guarantees, so a user of the assembler needs a way to distinguish in quality between two different genomes obtained in two different runs on the same library. Since we showed that this assembler is likely to converge on the correct assembly across independent runs, the user can apply this behaviour to determine the correct de-novo assembly by majority call.

For linear assemblies, determining the consensus genome from different runs is straightforward: to "agree", two genomes (a) have the same length, and (b) are either identical strings, or the reverse-complemented form. For circular assemblies, (a) and (b) must hold over the circular assemblies obtained.

Point (a) is in itself informative, given that it is unlikely that different read permutations end up assembled into the same contig length. To support this, Figure 8 shows, for two genomes with DTRs and a single read library in each case, the genome lengths obtained across 20 assemblies (done in both linear and circular mode). While both the linear and the circular assemblies most often yield correct contigs (split at random read boundaries through the genomes), the circular assembly mode has the advantage that it reports a trimmed genome without the terminal overlap, which is then consistent across runs for consensus purposes. In contrast, a linear assembly (although essentially correct) does not verify terminal overlap, and thus the genome length reported to the user falls apparently randomly between the correct reference genome length excluding the DTR, and that genome length including the DTR.

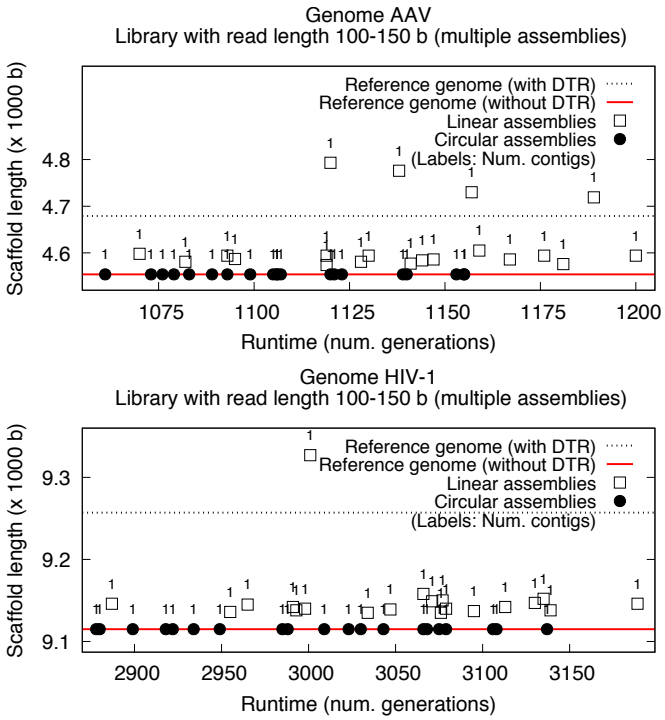

Figure 8: Linear vs. circular assemblies with this algorithm for two read libraries sampled from genomes with DTRs.

C. Runtimes

The main performance problem of this algorithm is its computational complexity, which manifests as long runtimes. The runtime of an assembly execution increases with both the size of the read library (in the number of reads), and the average read size. For the most complex run, with the current single-thread implementation (in Python, over the algorithmic library *inspyred*, `pypi.python.org/pypi/inspyred`) the runtime is 33 hours; for the simplest run, the runtime is

13 minutes.

D. Comparison with other assemblers

We compare this assembler, also in Table III, with 3 other assemblers: the greedy assembler SSAKE [7], SOAPdenovo [3] and Velvet [4] (based on de-Bruijn graphs), using their available options to recover repeats from the graphs using the long reads present in the read libraries. For SOAPdenovo and Velvet, in the cases when a fragmented scaffold was obtained instead of a contiguous genome, we report in the table the classic contig statistics: the number of contigs on the scaffold, and their minimum, maximum, and N50 lengths.

SSAKE was run with the options `-m 20 -x 20 -w 1` for each of the 5 read libraries available per genome and read-length interval; in all cases it obtained a fragmented scaffold. Velvet (executed with the options `-long` and `-exp_cov 10`) outperforms SOAPdenovo (options `-K 13 -R`), with both less fragmented scaffolds across the test cases, and with correct assemblies obtained for the genomes with long DTRs. The genome obtained in the latter cases by Velvet has the length of a circular assembly (excluding any DTR duplicate), like our algorithm. However, only our assembler both correctly signals the circularity of the genome, and deals accurately with interspersed reads.

VI. Conclusions

For applications which need "polished" genomes such as comparative genomics, and given the recent developments in sequencing technology which allows for accurate long reads to be obtained, the machine-learning algorithm underlying our assembler has the requisite power to deliver ready genomes. With low memory complexity (linear in the size of the population) yet high time complexity (affordable for applications which are not time-pressing), this assembler also has the advantage of single-handedly building trust in a genome by consensus.

Future work may improve this assembly method for genomes with repeats in three directions. More repeat types, such as longer inverted terminal repeats, may be treated. A reimplementation of the assembly algorithm in a compiled language, and execution using highly parallel computing architectures such as GPUs is expected to bring on the necessary speed-ups for the method to be more widely applicable. Finally, for application in practice, the algorithm should be evaluated with raw reads in which read errors are present.

References

[1] E. W. Myers, "Toward simplifying and accurately formulating fragment assembly," *Journal of Computational Biology*, vol. 2, no. 2, pp. 275–290, 1995.

[2] C. Alkan, S. Sajjadian, and E. E. Eichler, "Limitations of next-generation genome sequence assembly," *Nature methods*, vol. 8, no. 1, pp. 61–65, 2011.

[3] R. Luo, B. Liu, Y. Xie, Z. Li, W. Huang, J. Yuan, G. He, Y. Chen, Q. Pan, Y. Liu, J. Tang, G. Wu, H. Zhang, Y. Shi, Y. Liu, C. Yu, B. Wang, Y. Lu, C. Han, D. W. Cheung, S.-M. Yiu, S. Peng, Z. Xiaoqian, G. Liu, X. Liao, Y. Li, H. Yang, J. Wang, T.-W. Lam, and J. Wang, "SOAPdenovo2: an empirically improved memory-efficient short-read de novo assembler," *GigaScience*, vol. 1, no. 1, p. 18, 2012. [Online]. Available: http://dx.doi.org/10.1186/2047-217X-1-18

[4] D. R. Zerbino, G. K. McEwen, E. H. Margulies, and E. Birney, "Pebble and rock band: Heuristic resolution of repeats and scaffolding in the velvet short-read de novo assembler," *PLOS ONE*, vol. 4, no. 12, pp. 1–9, 12 2009. [Online]. Available: http://dx.doi.org/10.1371%2Fjournal.pone.0008407

[5] Celera Genomics, "Whole-genome shotgun assembler," https://www.ncbi.nlm.nih.gov/nuccore/KF319020.1, 2017.

[6] D. Bucur, *De Novo DNA Assembly with a Genetic Algorithm Finds Accurate Genomes Even with Suboptimal Fitness*. Springer International Publishing, 2017, pp. 67–82. [Online]. Available: http://dx.doi.org/10.1007/978-3-319-55849-3_5

[7] R. L. Warren, G. G. Sutton, S. J. M. Jones, and R. A. Holt, "Assembling millions of short DNA sequences using SSAKE," *Bioinformatics*, vol. 23, no. 4, pp. 500–501, 2007. [Online]. Available: http://bioinformatics.oxfordjournals.org/content/23/4/500.abstract

[8] M. Pop, "Genome assembly reborn: recent computational challenges," *Briefings in Bioinformatics*, vol. 10, no. 4, p. 354, 2009. [Online]. Available: http://dx.doi.org/10.1093/bib/bbp026

[9] S. L. Salzberg, A. M. Phillippy, A. Zimin, D. Puiu, T. Magoc, S. Koren, T. J. Treangen, M. C. Schatz, A. L. Delcher, M. Roberts, G. Marçais, M. Pop, and J. A. Yorke, "GAGE: A critical evaluation of genome assemblies and assembly algorithms," *Genome Research*, vol. 22, no. 3, pp. 557–567, 03 2012. [Online]. Available: http://www.ncbi.nlm.nih.gov/pmc/articles/PMC3290791/

[10] R. Li, C.-L. Hsieh, A. Young, Z. Zhang, X. Ren, and Z. Zhao, "Illumina synthetic long read sequencing allows recovery of missing sequences even in the finished C. elegans genome," *Nature Scientific Reports*, vol. 5, no. 10814, 2015.

[11] R. C. McCoy, R. W. Taylor, T. A. Blauwkamp, J. L. Kelley, M. Kertesz, D. Pushkarev, D. A. Petrov, and A.-S. Fiston-Lavier, "Illumina truseq synthetic long-reads empower de novo assembly and resolve complex, highly-repetitive transposable elements," *PLOS ONE*, vol. 9, no. 9, pp. 1–13, 09 2014. [Online]. Available: http://dx.doi.org/10.1371%2Fjournal.pone.0106689

[12] K. Bradnam, J. Fass, A. Alexandrov, P. Baranay, M. Bechner, I. Birol, S. Boisvert, J. Chapman, G. Chapuis, R. Chikhi *et al.*, "Assemblathon 2: evaluating de novo methods of genome assembly in three vertebrate species," *Gigascience 2: 10*, 2013.

[13] Y. Cherukuri and S. C. Janga, "Benchmarking of de novo assembly algorithms for Nanopore data reveals optimal performance of OLC approaches," *BMC Genomics*, vol. 17, no. Suppl 7, p. 507, 2016.

[14] NCBI, "Hepatitis C virus genotype 1, complete genome," https://www.ncbi.nlm.nih.gov/nuccore/NC_004102.1, 2017.

[15] ——, "Ciconia ciconia mitochondrial DNA, complete genome," https://www.ncbi.nlm.nih.gov/nuccore/AB026818.1, 2017.

[16] ——, "Simian virus 40 complete genome," https://www.ncbi.nlm.nih.gov/nuccore/J02400.1, 2017.

[17] ——, "Frog adenovirus 1, complete genome," https://www.ncbi.nlm.nih.gov/nuccore/NC_002501, 2017.

[18] ——, "HIV-1 isolate HP-9:03KDE11, complete genome," https://www.ncbi.nlm.nih.gov/nuccore/JQ316128.1, 2017.

[19] ——, "Adeno-associated virus 2, complete genome," https://www.ncbi.nlm.nih.gov/nuccore/AF043303.1, 2017.

[20] ——, "Proteus phage PM16, complete genome," https://sourceforge.net/projects/wgs-assembler/, 2017.

[21] M. Chen, Z. Tan, and G. Zeng, "Microsatellite is an important component of complete Hepatitis C virus genomes," *Infection, Genetics and Evolution*, vol. 11, no. 7, pp. 1646 – 1654, 2011. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1567134811002255

[22] A. J. Davison, K. M. Wright, and B. Harrach, "DNA sequence of frog adenovirus," *Journal of General Virology*, vol. 81, no. 10, pp. 2431–2439, 2000. [Online]. Available: http://jgv.microbiologyresearch.org/content/journal/jgv/10.1099/0022-1317-81-10-2431

[23] F. C. Krebs, T. H. Hogan, S. Quiterio, S. Gartner, and B. Wigdahl, "Lentiviral LTR-directed expression, sequence variation, and disease pathogenesis," *HIV sequence compendium*, pp. 29–70, 2001.

[24] D. Duan, P. Sharma, L. Dudus, Y. Zhang, S. Sanlioglu, Z. Yan, Y. Yue, Y. Ye, R. Lester, J. Yang, K. J. Fisher, and J. F. Engelhardt, "Formation of adeno-associated virus circular genomes is differentially regulated by adenovirus E4 ORF6 and E2a gene expression," *Journal of Virology*, vol. 73, no. 1, pp. 161–169, 1999. [Online]. Available: http://jvi.asm.org/content/73/1/161.abstract