Evaluation of genome assembly software based on long reads

Laurent Bouri[1,*], **Dominique Lavenier**[2], **Jean-François Gibrat**[3], **and Victoria Dominguez del Angel**[4]

[1]CNRS Engineer/ IFB
[2]CNRS Research Director, GenScale team leader
[3]INRA Research Director/ IFB
[4]ELIXIR Training Coordinator (FRANCE)/ IFB

ABSTRACT

During the last 30 years, Genomics has been revolutionized by the development of first- and second-generation sequencing (SGS) technologies, enabling the completion of many remarkable projects as the Human Genome Project[1,2], the 1000 Genomes Project[3] and the Human Microbiome Project[4].
In the last decade, SGS technologies based on massive parallel sequencing have dominated the market, thanks to their ability to produce enormous volumes of data cheaply. However, often genes and regions of interest are not completely or accurately assembled, complicating analyses or requiring additional cloning efforts for obtaining the correct sequences[5]. The fundamental obstacle in SGS technologies for obtaining high quality genome assembly is the existence of repetitions in the sequences. A promising solution to this issue is the advent of Third-generation sequencing (TGS) technologies based on long read sequencing[6].
TGS technologies have been used to produce highly accurate *de novo* assemblies of hundreds of microbial genomes[7],[8], and highly contiguous reconstructions of many dozens of plant and animal genomes, enabling new insights into evolution and sequence diversity[9],[10]. They have also been applied to resequencing analyses, to create detailed maps of structural variations in many species[11]. Also, these new technologies have been used to fill in many of the gaps in the human reference genome[12].
In this report, we compare and evaluate several genome assembly software based on TSG technology. The experimentation has been performed on 4 reference genomes and the results evaluated with the QUAST software. The 11 software that have been evaluated are: Celera Assembler[13], Falcon[14], Miniasm[15], Newbler[16], SGA Assembler[17], Smartdenovo[18], Abruijn[19], Ra[20], DBG2OLC[21], Spades[22] and Cerulean[23]. The first 8 software use only long reads, while the 3 last software can merge long and short reads

Keywords: Third-generation sequencing, Pacific Biosciences (PacBio), Oxford nanopore MinION, De novo assembly

Contents

1 Introduction

1.1 Background

The TGS technologies developed by the Pacific Bioscience (PacBio) and Oxford Nanopore Technology (ONT) companies are able to produce distributions of read lengths having a median greater than 10,000 bp and whose longest lengths are about 50,000 bp that are very useful to improve genome assembly. Indeed, such long reads allow the method to encompass most of the repetitive regions of the genome. However, these long reads exhibit 10% to 15% sequencing error rates, requiring a preliminary stage of correction before the assembly process.

There are two main families of assemblers based on long reads :

- Long Reads Only assembler (LRO);
- Short and Long Reads combined assembler (SLR).

LRO Assemblers take only long reads as inputs. SLR Assemblers require both long and short reads.

Some LRO assemblers require corrected long reads as input. Several software to correct long reads, based on two strategies, are available. The first strategy consists of aligning long reads against themselves. The second one uses short reads to correct long reads.

This report aims to provide a guide for helping researchers to choose the best assembly software considering:

- the coverage rate of the long reads dataset;
- the availability and quality of supplementary short reads
- the length of the genome to be assembled

We include details for each protocol to facilitate the computational reproducibility for each software approach.

1.2 Evaluated assemblers

1.2.1 LRO assemblers (Long Read Only)

Eight *de novo* assemblers are listed below. Clearly, assembler software prefer previously corrected long reads as input. However most of them can also accept non corrected reads (Table 1). Falcon is the only assembler to have an integrated correction module that can be bypassed.

Assemblers	Accept non corrected reads as input
Celera	no
Falcon	yes
Miniasm	yes
Newbler	no
SGA	no
Smartdenovo	yes
Abruijn	yes
Ra	yes

Table 1: List of *de novo* long reads assemblers and whether they can use non corrected long reads.

In general, these assemblers are based on the Overlap-Layout-Consensus (OLC) algorithm. First, this algorithm produces alignments between long reads, then it calculates the best overlap graph and finally it generates the consensus sequence of the contigs from the graph.
Obviously, the lower the sequencing error rate the more efficient the algorithm.

1.2.2 SLR Assemblers (Short and Long Read)

Until now, 3 hybrid assemblers have been proposed:

- DBG2OLC
- Spades

- Cerulean

Schematically, assembly pipelines that use both long and short reads generate a pre-assembly (production of contigs) using short reads, then the long reads are used to improve the pre-assembly by closing gaps, resolving repetitive regions,...

2 Method

2.1 Evaluated genomes

Genome name	Number of chromosomes	length
Acinetobacter DP1	1 chromosome	3 650 030 pb
Escherichia Coli K12 MG1655	1 chromosome	4 641 652 pb
Saccharomyces Cerevisae W303	16 chromosomes	11 633 571 pb
Caenorhabditis elegans	6 chromosomes	100 272 607 pb

Table 2: The reference genomes used in this report.

2.2 Datasets

The table below shows the 4 reference genomes and the datasets (short and long reads) used for this evaluation :

Genome	length (M bp)	Test	Minion (ONT)	PacBio	Illumina
Acinetobacter	3.9 M	1	10x, 2D, 3.4K reads		211K reads
		2	20x, 2D, 10K reads		211K reads
E.Coli	4.6 M	3		10x, P4C2, 36K reads	11M reads
		4		100x, P4C2, 91K reads	11M reads
		5		10x, P6C4, 8.7K reads	11M reads
		6		100x, P6C4, 87K reads	11M reads
		7	20x, 2D, 22K reads		11M reads
S. Cerevisa	11.6 M	8		10x, P4C2, 26K reads	3.8M reads
		9		100x,P4C2, 261K reads	3.8M reads
		10	20x, 2D, 47K reads		3.8M reads
C. Elegans	100 M	11		10x, P6C4, 92K reads	55M reads
		12		100x, P6C4, 740K reads	55M reads

Table 3: The datasets used in this report. These datasets provide longs reads from Oxford Nanopore technology or Pacbio Science or Illumina short reads.

The available Pacbio datasets have coverage rates from 10x to 100x. inaddition, 2 Pacbio sequencing techniques have been tested, the older polymerase and chemistry P4C2 to the newer P6C4 version. Tested ONT long reads are 2D reads, meaning that they are consensus sequences of both forward and reverse strands generated from the Minion device. Minion and Pacbio assembly cannot be directly compared as they don't have the same coverage in the datasets. Notice also that no Pacbio dataset has been tested for the assembly of the Acinetobacter genome.

2.3 Hardware resources

Software evaluations were done on the Genouest platform cluster https://www.genouest.org/cluster.

Assembler	Number of threads
Celera Assembler	8
Falcon	2 à 24
Miniasm	16
Newbler	1
SGA assembler	1
Smartdenovo	1
Abruijn	1
Ra	1
DBG2OLC	1
Spades	16
Cerulean	8

Table 4: Number of threads used for each software.

Cluster node configuration:

- **Number of CPU**: 40
- **CPU frequency**: 2.6 GHz
- **RAM available**: 256 Gb

2.4 Long reads correction

As some assembly software have an integrated correction step (Falcon) and some assemblers do not accept non corrected long reads (Celera assembler, Newbler and SGA), raw long reads must be previously corrected with a unique correction tool. Furthermore, when it is possible, the correction module integrated to the assembly software is bypassed. So, before each assembly, the *LoRDEC* software[24] is used to correct long reads. *LoRDEC* constructs a de Bruijn graph from the short reads and produces a long corrected read, based on an optimal graph path.

2.5 Evaluation of assemblies

The software *QUAST*[25] (QUality ASsessment Tool) has been used. It evaluates the assembly by calculating various metrics such as the number of contigs, the total length of the assembled genome, the N50 and the fraction of the reference genome found among the contigs. This fraction is deduced by aligning the assembled genome with the reference genome using the MuMmer[26] software.

3 LRO assemblers (*Long Read Only*)

3.1 Celera Assembler

Introduction

Celera Assembler is an assembler using the OLC algorithm, originally developed to produce assembly from Sanger data. It now supports long reads and 454 technology. Canu is a fork of the Celera Assembler, designed for high-noise single-molecule sequencing as Pacbio or ONT.

Website : `http://wgs-assembler.sourceforge.net`

Installation

Celera Assembler Celera Assembler can be downloaded as a source code to install or as precompiled binaries.

Compilation and installation of the source code:

```
$ bzip2 -dc wgs-8.3rc2.tar.bz2 | tar -xf
$ cd wgs-8.3rc2
$ cd kmer && make install && cd ..
$ cd src && make && cd ..
$ cd ..
```

Extraction of pre-compiled binaries files:

```
$ bzip2 -dc wgs-8.3rc2-*.tar.bz2 | tar -xf
```

Input data

Celera Assembler requires a fragment file (.frg) and an optional .spec file that launches the Celera Assembler pipeline. Some programs as **fastatoCA** and **fastqtoCA** are available in the WGS package to convert any common FASTA or FASTQ file to a fragment file (.frg). Converting a FASTA file requires not only the sequence file but also a FASTA quality file:

```
fastaToCA -l libraryname -s seq.fasta -q qlt.fasta > seq.frg
```

- `l` : library name
- `s` : sequence in FASTA format
- `q` : FASTA quality file

```
fastqToCA -libraryname LIB -technology pacbio-corrected -reads seq.fastq > seq.frg
```

- `libraryname` : library name
- `technology` : data type (pacbio, Illumina, 454,...)
- `reads` : FASTQ file

Pipeline

The runCA script divides the assembly pipeline in 9 stages. Each of these stages creates files in dedicated folders. First, *Celera Assembler* checks all input data for errors and load the valid data into the gkpStore database. From each fragment, a histogram of the kmer frequency is generated. The value from which a k-mer seed becomes non-informative and the depth of coverage are calculated. Then, multiple alignments are executed in order to correct any sequencing errors left and the best overlap graph is generated. Finally, The consensus sequences of the contigs are deduced from the graph and *Celera Assembler* ended by a scaffolding step.

```
runCA -d directory -p prefix -s specfile <option=value> ... <input-files>
```

- `d` : folder name to output results
- `p` : prefix used to create output filename
- `s` : optional .spec file

Encountered errors

Overlap job /root/wgs/Linux-amd64/bin/results/1-overlapper/001/000001 FAILED. 1 overlapper jobs failed

solution : change the merSize option to reduce the seed lenght used by the seed and extend algorithm.

---

1 unitig consensus jobs failed;
remove /root/wgs/Linux-amd64/bin/results5/5-consensus/consensus.sh to try again

solution : delete the file consensus.sh, then retry.

BEGIN failed–compilation aborted at
/data/bill.crosby/apps/wgs-8.3rc1/Linux-amd64/bin/caqc.pl http://caqc.pl/; line 18

solution: Install the perl module Statistics::Descriptive : `sudo cpan Statistics::Descriptive`

Output data

The *Celera Assembler* output consists of 9 folders, an ASM file, containing the precise description of the assembly with the generated scaffold sequences and a quality control file, providing statistics on the assembly. The QC file retains informations about generated scaffold (N50,...), long reads (quantity,...) and contigs (gaps quantity,...).

The ASM file is the *Celera Assembler* native output format. It can be converted in a fasta file format. To do so, the WGS-assembly package provide a script named asmOutputFasta:

```
asmOutputFasta -p prefix < output.asm
```

3.2 Falcon

Introduction

Falcon is an assembler using reads from Pacbio and ONT. Long reads are aligned against themselves to build consensus sequences that are subsequently used to generate the genome assembly. *Falcon* is advertised as an assembler of diploid genomes. It is advisable to have a coverage of 100x of Pacbio or ONT reads for a *de novo* assembly.

Website: https://github.com/PacificBiosciences/FALCON

Installation

Falcon can be downloaded as a source code to be installed. It requires a Sun Grid Engine environment. The installation requires gcc (4.8.3+) and python (2.7+). FALCON-integrate is an integration package for FALCON and its dependencies (DALIGNER, DAZZ_DB, pypeFLOW,...)

Create a virtual Python environments and activate it:

```
    $ FC=fc_env
    $ virtualenv -no-site-packages -always-copy $FC
    $ . $FC/bin/activate
```

Download and install Falcon and its dependencies:

```
    $ git clone git://github.com/PacificBiosciences/FALCON-integrate.git
    $ cd FALCON-integrate
    $ git submodule update --init
    $ cd pypeFLOW
    $ python setup.py install
    $ cd ..
    $ cd FALCON
    $ python setup.py instal
    $ cd ..
```

```
$ cd DAZZ_DB/
$ make
$ cp DBrm DBshow DBsplit DBstats fasta2DB $FC/bin/
$ cd ..
$ cd DALIGNER
$ make
$ cp daligner daligner_p DB2Falcon HPCdaligner LA4Falcon LAmerge
  LAsort FC/bin
$ cd ..
```

Input data

Falcon needs a configuration file named ”fc_run.cfg”, listing command lines to be executed, on which node in the cluster, with which resources and with which data. Here is an exemple of *Falcon* configuration file:

```
[General]
\#job_type = local

\# list of files of the initial bas.h5 files
input_fofn = input.fofn
\#input_fofn = preads.fofn

input_type = raw
\#input_type = preads

\# The length cutoff used for seed reads used for initial mapping
length_cutoff = 12000

\# The length cutoff used for seed reads usef for pre-assembly
length_cutoff_pr = 12000


jobqueue = your_queue
sge_option_da = -pe smp 8 -q %(jobqueue)s
sge_option_la = -pe smp 2 -q %(jobqueue)s
sge_option_pda = -pe smp 8 -q %(jobqueue)s
sge_option_pla = -pe smp 2 -q %(jobqueue)s
sge_option_fc = -pe smp 24 -q %(jobqueue)s
sge_option_cns = -pe smp 8 -q %(jobqueue)s

pa_concurrent_jobs = 32
ovlp_concurrent_jobs = 32

pa_HPCdaligner_option =  -v -dal24 -t16 -e.70 -l1000 -s1000
ovlp_HPCdaligner_option = -v -dal24 -t32 -h60 -e.96 -l500 -s1000

pa_DBsplit_option = -x500 -s200
ovlp_DBsplit_option = -x500 -s200

falcon_sense_option = --output_multi --min_idt 0.70 --min_cov 4
--local_match_count_threshold 2 --max_n_read 200 --n_core 6 --output_dformat

overlap_filtering_setting = --max_diff 100 --max_cov 100 --min_cov 20
--bestn 10 --n_core 24
```

The parameters ”input-fofn” and ”jobqueue” must be filled in before launching a job on a node of the cluster. The file with the ”.fofn” extension contains, in each line, a path to the dataset of long reads to assemble (FASTA file). Lastly, if the parameter ”input-type” from the configuration file is set to ”preads” instead of ”raw”, Falcon will consider that the long reads have already been corrected.

Pipeline

Make sure virtualenv is activated, then execute the next command line in order to launch the *Falcon* assembler:

```
$ fc_run.py fc_run.cfg
```

The Falcon pipeline is divided in the following stages :

- Raw sub-reads overlapping for error correction (DALIGNER)
- Pre-assembly and error correction
- Overlapping detection of the error corrected reads
- Overlap filtering
- Constructing graph from overlaps
- Constructing contig from graph

Encountered errors

fasta2DB: Could not find file

Solution : grant access right to fasta2DB.

Pacbio header line name inconsistent

Solution : Use a script to change the header and make them look like header from pacbio data.

Output data

The folder ”2-asm-falcon” holds data about the graph and the contigs generated from the assembly. The final assembly can be found in the file ”p-ctg.fa”.

3.3 Miniasm

Introduction

Miniasm is an assembler based on the OLC algorithm. *Miniasm* is able to generate a genome assembly from Pacbio or ONT long reads, corrected or not, in a short period of time. It is worth mentioning that *Miniasm* pipeline does not contain a consensus stage. It simply concatenates pieces of read sequences to generate the final unitig sequences. Thus the per-base error rate is similar to the raw input reads.

Website : http://github.com/lh3/miniasm

Installation

Miniasm can be downloaded as source code:

```
$ git clone https://github.com/lh3/minimap && (cd minimap && make)
$ git clone https://github.com/lh3/miniasm && (cd miniasm && make)
```

Input data

Miniasm needs the file containing the alignment data (”.paf.gz” file extension), generated by *Minimap*. *Minimap* accept FASTA and FASTQ file.

Pipeline

First of all, *Minimap* find approximate mapping positions between the long reads. Then, *Miniasm* uses the file generated by *Minimap* to build an assembly graph in a file with extension ".gfa".

```
$ minimap/minimap -Sw5 -L100 -m0 -t8 reads.fq reads.fq | gzip -1
 > reads.paf.gz
```

- `S` : Do not take into account reads that map against them
- `w` : Minimizer window length
- `L` : Minimal match length
- `m` : merge two strings if the given percentage is shared between minimizer
- `t` : Number of threads

```
$miniasm/miniasm -f reads.fq reads.paf.gz > reads.gfa
```

Output data

Finally, the following AWK script allows to convert the ".gfa" file to a FASTA file format :

```
$ awk '/^S/{print ">"$2"\n"$3}' input_file.gfa | fold > output_file.fa
```

3.4 Newbler

Introduction

Newbler is a set of scripts specifically designed for assembling data generated by 454 pyrosequencing platforms (454 Life Sciences).

Website : http://www.454.com/products/analysis-software/

Installation

In order to download *Newbler* binary files, a link must be requested on the website by filing out a form. Then download and extract the file "gsNewbler-2.9-1x8664.rpm".

Input data

Newbler does not accepts reads larger than 2000 bp. Reads whose size exceeds 2000bp must be truncated while maintaining a maximum of overlap between the sequences (500 bp overlap length).

Pipeline

First, the project must be created :

```
$ newAssembly projectname
```

Then, Truncated long reads can be loaded in a new project :

```
$ addRun -lib minion projectname run_minion.fasta
```

Lastly, the following command line launch the assembly :

```
$ runProject -mi 96 -ml 60  -sl 22 projectname
```

- `mi` : minimum percentage of identity
- `ml` : minimum overlap length
- `sl` : seed length

The first phase of the assembly consists in finding overlaps between the reads. First, *Newbler* produces 16-mer seeds for each read. When *Newbler* finds an overlap between two reads, it extends the overlap between the reads up to a minimum size (40bp by default), with a minimum percentage of identity (90 by default). *Newbler* then create a graph, and finally extract the contigs.

Newbler can also be executed with one command line:

```
$ runAssembly -o projectname -mi 96 -ml 60 -sl 22 run_minion.fasta
```

Output

The output folder contains the generated genome assembly in a file named ”454AllContigs.fna”.

3.5 SGA Assembler

Introduction

SGA is based on the ”string graph” of *Gene Myers* and uses the Burrows-Wheeler Transform/FM-index to find overlaps between reads.

Website : https://github.com/jts/sga

Installation

SGA assembler requires installation of the following dependencies :

· google sparse hash library (http://code.google.com/p/google-sparsehash/)

· the bamtools library (https://github.com/pezmaster31/bamtools)

· zlib (http://www.zlib.net/)

SGA installation :

```
$ git clone https://github.com/jts/sga.git
$ cd sga/src
$ ./autogen.sh
$ ./configure {with-sparsehash=<chemin vers sparsehash>
    --withbamtools=<pathway to bamtools>
    --prefix=<pathway to SGA install folder> && make && make install
```

Input data

SGA assembler accepts long reads in FASTA or FASTQ format.

Pipeline

SGA assembler is divided in 6 stages :

1. A preliminary step to assembly, removing reads containing letters other than ATGC
2. The construction of an FM-index from the file in Fasta or Fastq format
3. The correction of long reads based on overlaps
4. The removing of duplicated reads
5. The construction of a ”String Graph” from the identified overlaps between reads
6. Assembly from the graph created in the previous step.

```
$ ./sga preprocess reads.fasta
$ ./sga index -a ropebwt reads.fasta
$ ./sga correct reads.fasta
$ ./sga filter reads.fasta
$ ./sga overlap -m 17 reads.fasta
```

- `m` : overlap length (pb)

```
$ ./sga assemble reads.filter.pass.asqg.gz
```

Encountered errors

substring read found during overlap computation

Solution : run the rmdup script

```
$ ./sga rmdup reads.fasta
```

Note that the commands for constructing the graph and then assembling must contain the following input files :

- “reads.filter.pass.rmdup.fa”
- “reads.filter.pass.rmdup.asqg.gz”

Output data

The output of *SGA assembler* provides a ”default-contigs.fasta” file containing the final assembly. The name of this file can be modified with the option ”-o”.

3.6 Smartdenovo

Introduction

Smartdenovo is a fast *de novo* assembler for PacBio and ONT data. It produces a sequence consensus after aligning reads against themselves.

Website : `https://github.com/ruanjue/smartdenovo`

Installation

Smartdenovo requires an unix system.

Smartdenovo can be downloaded as source code

```
$ git clone https://github.com/ruanjue/smartdenovo.git &&
  (cd smartdenovo; make)
```

Input data

Smartdenovo takes long reads sequences in a FASTA file format.

Pipeline

Smartdenovo has several tools to find overlaps between reads, to identify the chimeric or low-quality regions, and then to construct consensus sequences. These tools are launched by the perl script ”smartdenovo.pl” :

```
$ smartdenovo.pl -p prefix reads.fa > prefix.mak
$ make -f prefix.mak
```

Output data

The genome assembly is in a file with ”.dmo.lay.utg” extension.

3.7 Abruijn

Introduction

Abruijn is a *de novo* assembler, currently under development. It creates a de Bruijn graph from long reads.

Website : `https://github.com/fenderglass/ABruijn`

Installation

Abruijn requires an unix system and Blasr.

Blasr installation:

```
$ git clone git://github.com/PacificBiosciences/blasr.git blasr
$ ./configure.py --no-pbbam HDF5_INCLUDE=f1 HDF5_LIB=f2
$ make blasr
```

Abruijn installation:

```
$ git clone https://github.com/fenderglass/ABruijn.git
$ cd Abruijn
$ make
```

Input data

Abruijn requires long reads stored in a FASTA file format, as well as the estimated coverage of long reads.

Pipeline

The *Abruijn* software begins by pre-assembling the long erroneous reads. To do so, *Abruijn* constructs an A-bruijn graph from solid kmers.Then, *Abruijn* looks for a path in the A-bruijn graph, thus giving a pre-assembled genome with errors. Finally, *Abruijn* use the BLASR alignment of the long reads against the preassembled genome in order to obtain a corrected assembly of the genome.

The following command line launch the assembly:

```
$ python abruijn.py -t 8 -k 16 reads.fasta <dossier de sortie>
  <taux de couverture>
```

· `t` : number of threads

· `k` : kmer length

Output data

Corrected sequences are located in a ”polished.fasta” file, stored in the specified output folder.

3.8 Ra

Introduction

Ra is a de novo assembler taking only long reads as input. It has the particularity of using a new mapper named Graphmap. Nevertheless, since the development of this tool is still recent, *Ra* does not include a consensus stage (as Miniasm), so the assembly file has the error rate similar to the input reads.

Website : `https://github.com/mariokostelac/ra-integrate`

Installation

Ra requires a linux system, ruby 2.2, make, g++ (4.8+) et graphviz. It is also possible to launch *Ra* via the use of a Docker container.

Compilation of *Ra*:

```
$ git clone--recursive https://github.com/mariokostelac/ra-integrate.git
$ make
```

Start *Ra* in a Docker container:

```
$ docker pull mariokostelac/ra-integrate:master
```

Input data

Ra only needs Long reads stored in a FASTA file format.

Pipeline

In order to find overlaps between the different long reads used to generated a genome assembly, Graphmap's agorithm is divided into 5 steps:

1. Reduces the search space and get seed hits as a form of coarse alignment using a novel adaptation of gapped spaced seeds.
2. Constructs anchors using a graph-based vertex-centric processing of seeds.
3. Chains anchors using a kmer version of longest common subsequence (LCSk) construction.
4. Refines alignments by chaining anchors in the anchored mode or with a form of L1 linear regression in the semiglobal alignment mode
5. Evaluates the remaining candidates to select the best location to reconstruct a final alignment

The following command line launch the assembly:

```
$ script/run reads.fa
```

Output data

The file containing the assembled sequences (FASTA) is in an output folder named "assembly.number"

4 SLR assemblers (*Short and Long Read*)

4.1 DBG2OLC

Introduction

DBG2OLC is an assembler using Illumina contigs as anchor points to construct an overlap graph with long reads.

Site web : https://github.com/yechengxi/DBG2OLC

Installation

DBG2OLC requires a linux system and the prior installation of *Blasr* and *Sparc*. *DBG2OLC*, *Blasr* and *Sparc* can be downloaded as source code. *Blasr* Installation requires hdf 1.8.12+. It is also possible to install *SparseAssembler* in order to construct contigs from short reads.

DBG2OLC installation

```
$ git clone http://git.code.sf.net/p/dbg2olc/code dbg2olc-code
$ cd dbg2olc-code
$ g++ DBG2OLC.cpp -o DBG2OLC
```

Blasr installation

```
$ git clone git://github.com/PacificBiosciences/blasr.git blasr
$ ./configure.py --no-pbbam HDF5_INCLUDE=f1 HDF5_LIB=f2
$ make blasr
```

- `f1` : fichier header HDF5
- `f2` : fichier librairie HDF5

Sparc installation

```
$ git clone http://git.code.sf.net/p/sparc-consensus/code sparc-consensus-code
$ g++ Sparc.cpp -o Sparc
```

Input data

DBG2OLC Accepts contigs with either long reads (Pacbio, ONT) in FASTA format, or short reads like Illumina in FASTA or FASTQ format. Contigs can previously be constructed using a DBG assembler such as *SparseAssembler*.

Pipeline

DBG2OLC software is divided in 5 stages :

1. Construction of a De-Bruijn graph and creation of contigs from short reads (*SparseAssembler*)
2. Alignment of contigs with each long read. Long reads are compressed into lists of anchors
3. Execution of multiple alignments to suppress chimeric long reads
4. Construction of an overlap graph using long compressed reads
5. Deduction of a consensus sequence from the graph

The following command launch the pipeline:

```
$ ./DBG2OLC k 17 KmerCovTh 2 MinOverlap 20 AdaptiveTh 0.002 Contigs Contigs.txt \\
    RemoveChimera 1 f <fichier_pacbio1> f <fichier_pacbio2>
```

- `KmerCovTh` : fixed k-mer matching threshold
- `MinOverlap` : minimum overlap score between a pair of long reads
- `AdaptiveTh` : adaptive k-mer matching threshold
- `k` : kmer length

- `Contigs` : the contigs file in Fasta format
- `MinLen` : read minimum length
- `RemoveChimera` : suppresses chimeric reads in a data set.

This parameters are critical for performance : KmerCovTh, MinOverlap et AdaptiveTh.

For PacBio data, depending on the coverage, it is advised to use the following values :

Coverage	10x/20x	50x/100x
KmerCovTh	2-5	2-10
MinOverlap	10-30	50-150
AdaptiveTh	0.001 0.01	0.01-0.02

Table 5: Values to use according to the coverage rate of the long reads dataset.

A file named "*backbone_raw.fasta*" contains scaffolds constructed by DBG2OLC. In order to finalize the assembly, the script "*split_and_run*" must be executed. Verify that *Blasr* has been defined in the PATH variable, that the *Sparc* binary is present in the same folder as the script to be executed and the following files:

- The scaffolds file produced by *DBG2OLC* ("*backbone_raw.fasta*")
- The consensus file generated by *DBG2OLC* ("*DBG2OLC_Consensus_info.txt*")
- The contigs file
- The long reads file

Then,It is necessary to merge the long reads and the contigs files into a single file.

```
$ cat Contigs.txt pb_reads.fasta > ctg_pb.fasta
```

Finally, the script "*split_and_run_sparc.sh*" located in the "utility" folder must be run.

```
$ sh ./split_and_run_sparc.sh backbone_raw.fasta  DBG2OLC_Consensus_info.txt \\
ctg_pb.fasta ./consensus_dir > consensus_log.txt
```

Encountered errors

./split_and$_r$un_sparc_r2.sh: 1: eval: blasr: not found

Solution: Edit the script "*split_and_run_sparc.sh*" so that BLASR can be found.

Output file

The script "*spli_and_run*" generates a file named "final_assembly.fasta" in the *utility* folder, containing scaffolds from the finished assembly.

DBG2OLC pipeline

4.2 Spades

Introduction

SPAdes (St. Petersburg genome assembler) is an assembler originally designed for the assembly of small genomes. It supports Illumina / IonTorrent reads, and now has an option to run hybrid assembly with long reads.

site web : http://bioinf.spbau.ru/spades

Installation

SPAdes requires a linux or Mac OS system and python. *SPAdes* can be downloaded as source code or pre-computed binaries.

Download ans extract binaries files:

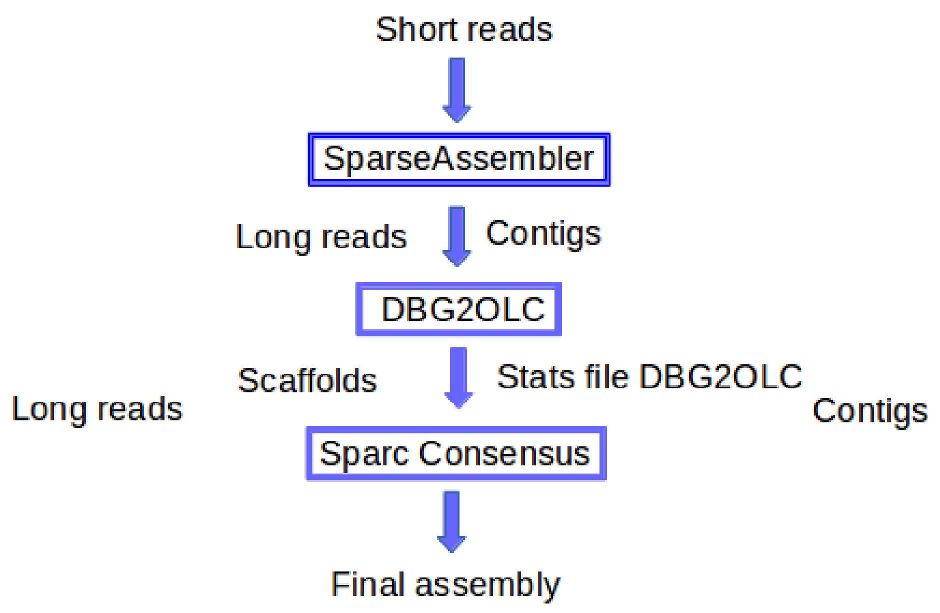


Figure 1. Details of the DBG2OLC pipeline. DBG2OLC uses contigs with either long reads or short reads to generate data such as scaffolds and statistics on the assembly. Sparc will then start from this files to create a final consensus of the assembly

```
$ wget http://spades.bioinf.spbau.ru/release3.6.0/SPAdes-3.6.0-Linux.tar.gz
$ tar -zxf SPAdes-3.6.0-Linux.tar.gz
$ cd SPAdes-3.6.0-Linux/bin/
```

Download and install source code:

```
$ wget http://spades.bioinf.spbau.ru/release3.6.0/SPAdes-3.6.0.tar.gz
$ tar -zxf SPAdes-3.6.0.tar.gz
$ cd SPAdes-3.6.0
$ ./spades_compile.sh
```

Input data

SPAdes supports paired-end reads, mate-pairs and unpaired reads in FASTA or FASTQ format. If available, contigs can be added as input data to enhance the assembly.

Pipeline

SPAdes constructs a de-Bruijn graph from k-mers of different length. Thus, a smaller k value in low-coverage regions minimizes fragmentation, while a higher k-value in heavily covered areas reduces the number of chimeric contigs. Contigs are then deduced from the graph. Finally, long reads are used for closing gaps and identifying repeated regions.

```
$ spades.py  --only-assembler --nanopore <file_name> -s <file_name> \\
     -trusted-contigs  <file_name> -o <output_dir>
```

- `--only-assembler` : start the assembly module
- `--nanopore` : long reads file
- `-s` : short reads file
- `--trusted-contigs` : contigs file
- `-o` : Output folder

Output data

SPAdes generates several files in the output folder:

- ”scaffolds.fasta”: scaffolds created by the assembly.
- ”param.txt”: list of parameters
- ”warning.log”: log file

4.3 Cerulean

Introduction

Cerulean extends the contigs assembled from short reads, such as Illumina, using long reads.

Website : http://sourceforge.net/projects/ceruleanassembler/

Installation

Cerulean requires linux system and the installation of this software:

- Python 2.7.1
- numpy et matplotlib (python libraries)
- Abyss assembler http://www.bcgsc.ca/platform/bioinfo/software/abyss
- Blasr (from SMRT Analysis toolkit): http://pacbiodevnet.com/
- Pbjelly : https://sourceforge.net/projects/pb-jelly/

Download and extract Cerulean scripts:

```
$ wget
http://sourceforge.net/projects/ceruleanassembler/files/Cerulean_v_0_1.tar.gz
$ tar -zxf Cerulean_v_0_1.tar.gz
```

Input data

Cerulean needs the contigs assembled from the short reads assembler *Abyss*, as well as the mapping of the long reads on these contigs, performed by *Blasr*.

Assembly from Illumina reads.

```
$ abyss-pe k=64 n=10 in='reads1.fastq reads2.fastq' name=$<$dataname$>$
```

- `k` : k-mer length
- `n` : minimum number of pairs required for building contigs

2 files are then created :

- ”<dataname>-contigs.fa”: contigs sequences
- ”<dataname>-contigs.dat”: graph structure

Mapping of long reads on contigs with *Blasr* :

```
$ blasr <dataname>_pacbio.fa <dataname>-contigs.fa -minMatch 10
  -minPctIdentity 70 -bestn 30 -nCandidates 30 -maxScore 500
  -nproc <numthreads> -noSplitSubreads
  -out <dataname>_pacbio_contigs_mapping.fasta.m4
```

- `minMatch` : Minimum seed length
- `minPctIdentity` : Minimum pourcentage of identity
- `bestn` : display the n best alignments

- `nCandidates` : display the n best candidates
- `maxScore` : maximum score to display
- `nproc` : number of threads
- `noSpliSubreads` : reads are not splited

Pipeline

All input files must be in the same folder :

- <dossier>/<dataname>-contigs.fa
- <dossier>/<dataname>-contigs.dot
- <dossier>/<dataname>_pacbio_contigs_mapping.fasta.m4

Celurean is started by the following command line :

```
$ python src/Cerulean.py --dataname <dataname> --basedir <basedir> --nproc <numthreads>
```

- `dataname?` datasets name
- `basedir?` pathway to input folder
- `nproc?` number of threads

Output data

Celurean generates the file ”<dossier>_cerulean.fasta”, containing the assembly results. Finally,*Cerulean* authors recommended using *PBJelly* in order to close remaining gaps.

Cerulean pipeline

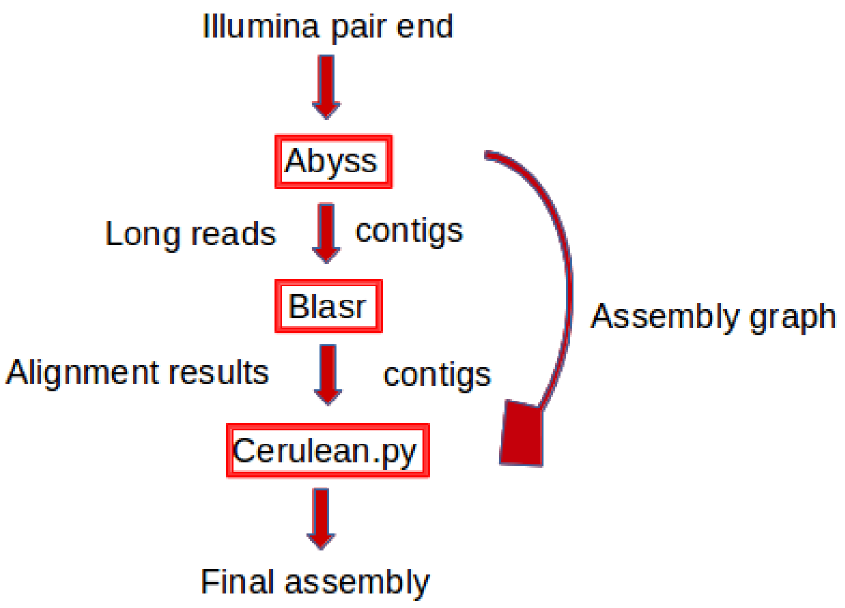


Figure 2. Details of the Cerulean pipeline. Cerulean uses data from the short reads assembler Abyss and the mapping of the long reads (Blasr) on contigs to generate the final assembly

5 Results

5.1 Evaluation of assembly

The reference genome is used to evaluate the quality of the different assemblies, on the basis of the results of the metrics produced by the QUAST software. The different metrics are listed below:

· **# contigs ($>$ 1000bp)** : Total number of contigs exceeding 1000 bp.

· **Largest contig**: The length of the longest contig in the assembly.

· **Total length**: total number of base pairs in the contigs generated by the assembly.

· **N50**: the minimum length of a set of contigs ordered by decreasing length such that the sum of base pairs in this set is larger than or equal to half the total number of base pairs included in the assembly.

· **Genome fraction (%)**: the percentage of aligned bases in the reference genome.

The combination of this metrics defines the assembly quality. Obviously, the best assembly result must take into account fewer number of contigs, higher N50, higher Largest contig, a total length of the assembly as close as possible to the expected genome length and higher genome fraction in a minimum execution time.

5.2 Benefit of long reads in hybrid assembly

Finally, in order to attest the advantages of long reads in an hybrid assembly pipeline, we try to assemble genomes solely with short reads dataset (indicated as ”Run 2” in the result tables and tested for *DBG2OLC* and *Spades*), as opposed to an hybrid assembly executed with a combination of long and short read datasets (indicated as ”Run 1” in the result tables).

5.3 Testing Data Sets

Acinetobacter sp. adp1, Illumina: Internal data

Acinetobacter sp. adp1, Minion run5: http://www.genoscope.cns.fr/externe/nas/datasets/MinION/acineto/acineto_nanopore_2D_run5.fa.gz

Acinetobacter sp. adp1, Minion run6: http://www.genoscope.cns.fr/externe/nas/datasets/MinION/acineto/acineto_nanopore_2D_run6.fa.gz

Escherichia coli k-12, Illumina: ftp://webdata:webdata@ussd-ftp.Illumina.com/Data/SequencingRuns/MG1655/MiSeq_Ecoli_MG1655_110721_PF_R1.fastq.gz
ftp://webdata:webdata@ussd-ftp.Illumina.com/Data/SequencingRuns/MG1655/MiSeq_Ecoli_MG1655_110721_PF_R2.fastq.gz

Escherichia coli k-12, Minion: http://www.genoscope.cns.fr/externe/nas/datasets/MinION/ecoli/Ecoli_LomanAll2D.fa.gz

Escherichia coli k-12, Pacbio (p4c2): http://sourceforge.net/projects/wgs-assembler/files/wgs-assembler/wgs-8.0/datasets/escherichia_coli_k12_mg1655.m130404_014004_sidney_c100506902550000001823076
s1_p0.1.fastq.xz
http://sourceforge.net/projects/wgs-assembler/files/wgs-assembler/wgs-8.0/datasets/escherichi
coli_k12_mg1655.m130404_014004_sidney_c100506902550000001823076808221337_s1_p0.2.fastq.xz
http://sourceforge.net/projects/wgs-assembler/files/wgs-assembler/wgs-8.0/datasets/escherichi
coli_k12_mg1655.m130404_014004_sidney_c100506902550000001823076808221337_s1_p0.3.fastq.xz

Escherichia coli k-12 Pacbio (p6c4): https://github.com/PacificBiosciences/DevNet/wiki/E.-coli-Bacterial-

Saccharomyces cerevisiae W303, Illumina: Accession number: SRR567755

Saccharomyces cerevisiae W303, Minion: http://www.genoscope.cns.fr/externe/nas/datasets/MinION/yeast/W303_ONT_Raw_reads_2D.fa.gz

Saccharomyces cerevisiae W303, Pacbio (p4c2): https://github.com/PacificBiosciences/DevNet/wiki/Saccharomyces-cerevisiae-W303-Assembly-Contigs

Caenorhabditis elegans, Illumina: Accession numbers: SRR065388, SRR065389, SRR065390

Caenorhabditis elegans, Pacbio (p6c4): http://datasets.pacb.com.s3.amazonaws.com/2014/c_elegans/list.html

5.4 Test 1: Acinetobacter sp, ADP1, run5; Minion 10x

Datasets:

- ONT 2D reads corrected by Lordec: 3 427 reads
- Illumina reads (MiSeq) : 211 219 reads of length 150pb, 16x
- Contigs generated by sparse assembler with Illumina reads (233 contigs), needed for hybrid assembly

LRO Assemblers

Metrics	Celera	Falcon	Miniasm	Newbler
# contigs (>=1000pb)	–	237	34	910
N50	–	57 840	112 086	487 000
Largest contig	–	658 902	340 549	24 4182
Execution time	–	1h52m58s	3.6sec	8m20s
Total length	–	10 509 831	2 712 071	5 203 730
Genome fraction	–	96.5	71.63	98.9

Metrics	SGA	Smartdenovo	Abruijn	**Ra**
# contigs (>=1000pb)	–	26	26	**2**
N50	–	164 126	130 182	**2 510 403**
Largest contig	–	363 376	345 439	**2 510 403**
Execution time	–	37s	20m48s	**47s**
Total length	–	3 341 599	3 072 900	**3 383 054**
Genome fraction	–	89.02	84.69	**86.75**

Table 6: Quast results generated from various LRO assemblers with 10x of Minion long reads (Acinetobacter sp. ADP1). The lack of results for Celera and SGA assemblers is caused by the low coverage rate of long reads.

Assemblers SLR

	DBG2OLC		Spades		Cerulean
Metrics	**run 1**	run 2	run 1	run 2	run 1
# contigs (>=1000pb)	**1**	48	3	44	8
N50	**3 636 390**	201 371	3 601 686	191 200	1 474 160
Largest contig	**3 636 390**	323 681	3 601 686	377 379	1 845 480
Execution time	**38s**	1m12s	7m10s	20m12s	12m
Total length	**3 636 390**	3 631 852	3 607 931	3 554 907	10 700 143
Genome fraction	**99**	99	99.99	98.4	99.77

Table 7: Quast results generated from various SLR assemblers with 10x of Minion long reads, contigs generated from short reads and Illumina short reads (Acinetobacter sp. ADP1).

5.5 Test 2: Acinetobacter sp, ADP1, run6; Minion 20x

Datasets:

- ONT 2D reads corrected by Lordec: 10 116 reads
- Illumina reads(MiSeq) : 211,219 reads of length 150pb, 16x
- Contigs generated by sparse assembler with Illumina reads (233 contigs), needed for hybrid assembly

Assemblers LRO

Metrics	Celera	Falcon	**Miniasm**	Newbler
# contigs (>=1000pb)	–	29	**1**	2 056
N50	–	3 651 707	**3 594 439**	2 321
Largest contig	–	356 596	**3 594 439**	325 922
Execution time	–	2h18m3s	**13s**	27m1s
Total length	–	3 651 707	**3 594 439**	7 636 549
Genome fraction	–	90.6	**91.08**	98.9

Metrics	SGA	**Smartdenovo**	**Abruijn**	Ra
# contigs (>=1000pb)	–	**1**	**1**	4
N50	–	**3 622 997**	**3 601 882**	1 215 050
Largest contig	–	**3 622 997**	**3 601 882**	1 717 290
Execution time	–	**1m33s**	**22m9s**	4m29s
Total length	–	**3 622 997**	**3 601 882**	3 648 098
Genome fraction	–	97.22	**100**	88.45

Table 8: Quast results generated from various LRO assemblers with 20x of ONT long reads (Acinetobacter sp. ADP1).

The lack of results for Celera and SGA assemblers is caused by the low coverage rate of long reads.

SLR assemblers

	DBG2OLC		**Spades**		Cerulean
Metrics	**run 1**	run 2	**run 1**	run 2	run1
# contigs (>=1000pb)	**3**	48	**3**	44	7
N50	**2 743 656**	201 371	**3 602 046**	191 200	881 591
Largest contig	**23 681**	323 681	**3 602 046**	377 379	2 302 560
Execution time	**1m27s**	1m12s	**11m25s**	20m12s	33min
Total length	**3 341 152**	3 631 852	**3 608 291**	3 554 907	7 041 048
Genome fraction	**88.35**	99	**99.99**	98.4	99.7

Table 9: Quast results generated from various SLR assemblers with 20x of ONT long reads, contigs generated from short reads and Illumina short reads (Acinetobacter sp. ADP1).

5.6 Test 3: Escherichia coli k-12, reads Pacbio 10x (P4-C2)

Datsets:

- Pacbio reads corrected with lordec (10x coverage): 36 355 reads
- Illumina reads(MiSeq) : 11 458 940 reads of length 150pb, 370x
- Contigs generated by sparse assembler from Illumina reads (1 876 792 contigs), needed for hybrid assembly

Assemblers LRO

Metrics	Celera	Falcon	Miniasm	Newbler
# contigs (>=1000pb)	332	–	161	7 537
N50	47 318	–	19 379	1 999
Largest contig	157 193	–	54 170	31 521
Execution time	1h41m6s	–	15s	21m42s
Total length	5 097 702	–	1 921 945	18 789 586
Genome fraction	97.7	–	55.7	98.69

Metrics	SGA	**Smartdenovo**	Abrijn	Ra
# contigs (>=1000pb)	27 519	**93**	–	152
N50	1 795	**33 149**	–	59 639
Largest contig	20 571	**95 016**	–	265 054
Execution time	45m12s	**18s**	–	3m11s
Total length	91 585 601	**2 787 937**	–	5 638 785
Genome fraction	99.9	**53.04**	–	87.13

Table 10: Quast results generated from various LRO assemblers with 10x of P4C2 Pacbio long reads (Escherichia coli k-12).

The lack of results for Falcon et Abruijn assemblers is caused by the low coverage rate of long reads.

SLR Assemblers

	DBG2OLC		**Spades**		**Cerulean**
Metrics	run 1	run 2	**run 1**	run 2	run1
# contigs (>=1000pb)	167	502	**33**	90	**25**
N50	24 576	729	**423 200**	126 572	**284 387**
Largest contig	66 617	4 730	**584 914**	221 710	**855 359**
Execution time	2h33m16s	1h6m15s	**20h6m40s**	1d14h7m47s	**3h2m9s**
Total length	3 285 170	3 006 518	**4 752 529**	4 551 234	**5 536 433**
Genome fraction	72.2	61	**99.1**	97.9	**99.388**

Table 11: Quast results generated from various SLR assemblers with 10x of P4C2 Pacbio long reads, contigs generated from short reads and Illumina short reads (Escherichia coli k-12).

5.7 Test 4 : Escherichia coli k-12, reads Pacbio 100x (P4-C2)

Datasets:

- Pacbio reads corrected by lordec (100x coverage): 91 394 reads
- Illumina reads (MiSeq) : 11 458 940 reads of length 150pb, 370x
- Contigs generated by sparse assembler from Illumina reads (1 876 792 contigs), needed for hybrid assembly

LRO assemblers

Metrics	Celera	Falcon	**Miniasm**	Newbler
# contigs (>=1000pb)	317	9 070	**1**	16 421
N50	71 566	15 423	**4 685 365**	1 999
Largest contig	281 759	69 760	**4 685 365**	35 033
Execution time	13h50m	2h9min	**2m41s**	2h5m3s
Total length	6 968 300	111 813 089	**4 685 365**	3 6154 946
Genome fraction	99.9	100	**96.98**	99.6

Metrics	SGA	Smartdenovo	**Abrijn**	Ra
# contigs (>=1000pb)	80 568	1	**1**	332
N50	7 366	4 682 708	**4 642 185**	48 453
Largest contig	29 821	4 682 708	**4 642 185**	8 062 545
Execution time	4h43m27s	13m18s	**45m56s**	53m17s
Total length	458 775 476	4 682 708	**4 642 185**	8 062 545
Genome fraction	100	96.69	**99.9**	94.42

Table 12: Quast results generated from various LRO assemblers with 100x of P4C2 Pacbio long reads (Escherichia coli k-12).

SLR assembler

	DBG2OLC		Spades		**Cerulean**
Metrics	run 1	run 2	run 1	run 2	**run1**
# contigs (>=1000pb)	361	502	27	90	**17**
N50	32 616	729	488 422	126 572	**613 288**
Largest contig	206 766	4 730	960 524	221 710	**737 401**
Execution time	2h40m	1h6m15s	12h4m21s	1d14h7m47s	**3h54m47s**
Total length	7 520 958	3 006 518	4 941 241	4 551 234	**5 278 430**
Genome fraction	90.8	61	99.55	97.9	**99.38**

Table 13: Quast results generated from various SLR assemblers with 100x of P4C2 Pacbio long reads, contigs generated from short reads and Illumina short reads (Escherichia coli k-12).

5.8 Test 5: Escherichia coli k-12, reads Pacbio 10x (P6-C4)

Datasets:

- reads Pacbio corrected by lordec (10x coverage) : 8 746 reads
- reads Illumina (MiSeq) : 11 458 940 reads of length 150pb, 370x
- Contigs generated by sparse assembler from Illumina reads (1 876 792 contigs), needed for hybrid assembly

Assemblers LRO

Metrics	Celera	Falcon	Miniasm	Newbler
# contigs (>=1000pb)	60	–	26	2 136
N50	167 523	–	350 193	5 198
Largest contig	357 039	–	703 052	79 937
Execution time	3h1m47s	–	10s	10m
Total length	4 838 203	–	4 585 882	8 129 512
Genome fraction	97.3	–	95.93	98.97

Metrics	SGA	**Smartdenovo**	Abruijn	Ra
# contigs (>=1000pb)	178	**9**	10	12
N50	1 3410	**1 207 236**	857 514	583 331
Largest contig	28016	**1 490,628**	1 717 212	295 905
Execution time	1h22m	**1m23s**	10m9s	3m17s
Total length	1 281 475	**4 650 012**	4 576 044	4 893 126
Genome fraction	16.04	**94.86**	97.2	94.39

Table 14: Quast results generated from various LRO assemblers with 10x of P6C4 Pacbio long reads (Escherichia coli k-12).

The lack of results for the Falcon assembler is caused by the low coverage rate of long reads.

SLR assembler

	DBG2OLC		**Spades**		**Cerulean**
Metrics	run 1	run 2	**run 1**	run 2	**run1**
# contigs (>=1000pb)	68	502	**27**	90	**27**
N50	64 971	729	**472 057**	126 572	**400 533**
Largest contig	208 657	4 730	**708 505**	221 710	**610 814**
Execution time	2h32m55s	1h6m15s	**14h35m23s**	1d14h7m47s	**3h5m24s**
Total length	3 618 860	3 006 518	**5 171 060**	4 551 234	**4 774 395**
Genome fraction	75	61	**99.4**	97.9	**99.28**

Table 15: Quast results generated from various SLR assemblers with 10x of P6C4 Pacbio long reads, contigs generated from short reads and Illumina short reads (Escherichia coli k-12).

5.9 Test 6: Escherichia coli k-12, reads Pacbio 100x (P6-C4)

Datasets:

- Pacbio reads corrected by lordec (100x coverage): 87 497 reads
- Illumina reads(MiSeq) : 11 458 940 reads of length 150pb, 370x
- Contigs generated by sparse assembler from Illumina reads (1 876 792 contigs), needed for hybrid assembly

LRO assemblers

Metrics	Celera	Falcon	**Miniasm**	Newbler
# contigs (>=1000pb)	451	33	**2**	27 179
N50	36 715	291 737	**4 680 508**	1 999
Largest contig	370 387	938 257	**4 680 508**	15 870
Execution time	6d6h51m49s	53m26s	**5m32s**	7h6m
Total length	1 0746 953	4 845 711	**4 734 166**	57 510 328
Genome fraction	99.55	94.5	**96.53**	99.77

Metrics	SGA	Smartdenovo	**Abruijn**	Ra
# contigs (>=1000pb)	1410	12	**1**	120
N50	3 352	535 879	**4 642 563**	130 683
Largest contig	18 357	1 303 952	**4 642 563**	415 320
Execution time	13h19m	30m46s	**1h57m29s**	2h45m4s
Total length	4 425 758	4 840 284	**4 642 563**	7 907 056
Genome fraction	49.6	95.90	**99.9**	92.49

Table 16: Quast results generated from various LRO assemblers with 100x of P6C4 Pacbio long reads (Escherichia coli k-12).

SLR assemblers

	DBG2OLC		**Spades**		Cerulean
Metrics	run 1	run 2	**run 1**	run 2	run1
# contigs (>=1000pb)	134	502	**23**	90	72
N50	81 570	729	**449 002**	126 572	121 474
Largest contig	386 914	4 730	**707 712**	221 710	287 981
Execution time	2h43m30s	1h6m15s	**13h50m18s**	1d14h7m47s	4h25m26s
Total length	6 439 845	3 006 518	**4 948 122**	4 551 234	4 772 924
Genome fraction	91.57	61	**99.56**	97.9	98.9

Table 17: Quast results generated from various SLR assemblers with 100x of P6C4 Pacbio long reads, contigs generated from short reads and Illumina short reads (Escherichia coli k-12).

5.10 Test 7: Escherichia coli k-12, Minion 20x

Datsets:

· ONT 2D reads corrected by lordec (10x coverage): 22 270 reads

· Illumina reads(MiSeq) : 11 458 940 reads of length 150pb, 370x

· Contigs generated by sparse assembler from Illumina reads (1 876 792 contigs), needed for hybrid assembly

LRO assemblers

Metrics	Celera	Falcon	**Miniasm**	Newbler
# contigs (>=1000pb)	189	–	**1**	3 769
N50	87 431	–	**4 665 895**	3 191
Largest contig	313 881	–	**4 665 895**	33 550
Execution time	3h43m50s	–	**21.6sec**	15m40s
Total length	5 758 951	–	**4 665 895**	7 798 978
Genome fraction	99.2	–	**88.85**	98.5

Metrics	SGA	**Smartdenovo**	**Abruijn**	Ra
# contigs (>=1000pb)	17 680	**2**	**5**	7
N50	9 481	**4 650 531**	**4 618 085**	994 753
Largest contig	47 228	**4 650 531**	**2 055 696**	1 509 502
Execution time	2h16m37s	**3m25s**	**7h55m54s**	4m23s
Total length	134 072 452	**4 707 245**	**4 618 085**	4 737 110
Genome fraction	99.99	**91.15**	**99.29**	86.40

Table 18: Quast results generated from various LRO assemblers with 20x of ONT long reads (Escherichia coli k-12).

The lack of results for the Falcon assembler is caused by the low coverage rate of long reads.

SLR assemblers

	DBG2OLC		Spades		**Cerulean**
Metrics	run 1	run 2	run 1	run 2	**run1**
# contigs (>=1000pb)	129	502	26	90	**13**
N50	50 709	729	472 057	126 572	**2 891 290**
Largest contig	135 085	4 730	1 030 461	221 710	**2 891 290**
Execution time	2h35m	1h6m15s	20h35m	1d14h7m47s	**5h6m39s**
Total length	4 107 939	3 006 518	5 264 214	4 551 234	**5 011 011**
Genome fraction	73	61	99.3	97.9	**99.18**

Table 19: Quast results generated from various SLR assemblers with 20x of ONT long reads, contigs generated from short reads and Illumina short reads (Escherichia coli k-12).

5.11 Test 8: Saccharomyces cerevisae W303, Pacbio reads 10x (P4-C2)

Datasets:

- Pacbio reads corrected by Lordec (10x): 26 196 reads
- Illumina reads(MiSeq) : 3 815 678 reads of length 100pb, 65x
- Contigs generated by sparse assembler from Illumina reads(10 055 contigs), needed for hybrid assembly

LRO assemblers

Metrics	Celera	Falcon	Miniasm	Newbler
# contigs (>=1000pb)	576	1 157	158	11 385
N50	50 224	16 175	34 485	1,999
Largest contig	245 675	68 708	86 751	52 210
Execution time	2h23m42s	2m13s	16s	1h42m36s
Total length	14 792 911	11 458 559	4 996 345	31 879 905
Genome fraction	95.3	97.24	40.024	97.4

Metrics	SGA	Smartdenovo	**Abrijn**	**Ra**
# contigs (>=1000pb)	810	174	**44**	**121**
N50	8 396	60 876	**60 553**	**159 052**
Largest contig	21 663	215 286	**118 337**	**427 616**
Execution time	2h57m3s	1m31s	**21m25s**	**17m36s**
Total length	5 073 045	8 456 624	**2 705 209**	**11 991 617**
Genome fraction	20.7	66.95	22.56	**85.61**

Table 20: Quast results generated from various LRO assemblers with 10x of P4C2 Pacbio long reads (Saccharomyces cerevisae W303).

SLR assemblers

	DBG2OLC		**Spades**		Cerulean
Metrics	run 1	run 2	**run 1**	run 2	run1
# contigs (>=1000pb)	**58**	1 157	**256**	564	259
N50	**427 868**	16 175	**83 770**	37 681	153 850
Largest contig	**989 517**	68 633	**369 700**	145 780	444 461
Execution time	**1m32s**	2m22s	**1h4m**	24m30s	3h3m16s
Total length	**11 952 152**	11 458 559	**11 760 318**	11 518 863	12 849 306
Genome fraction,	**92.398**	97.24	**98.44**	98.09	94.8

Table 21: Quast results generated from various SLR assemblers with 10x of P4C2 Pacbio long reads, contigs generated from short reads and Illumina short reads (Saccharomyces cerevisae W303).

5.12 Test 9: Saccharomyces cerevisae W303, Pacbio reads 100x (P4-C2)

Datasets:

- Pacbio reads corrected by lordec (100x): 261 964 reads
- Illumina reads(MiSeq) : 3 815 678 reads of length 100pb, 65x
- Contigs generated by sparse assembler from Illumina reads (10 055 contigs), needed for hybrid assembly

LRO assemblers

Metrics	Celera	Falcon	**Miniasm**	Newbler
# contigs (>=1000pb)	5 599	17 857	**25**	64 900
N50	16 276	18 670	**755 806**	1 999
Largest contig	89 231	78 140	**1 271 021**	42 447
Execution time	1d8h1m12s	39d	**7m24s**	6d22h9m
Total length	52 637 674	267 821 084	**12 022 519**	140 588 558
Genome fraction	88.1	99.28	**94.26**	98.4

Metrics	SGA	Smartdenovo	**Abruijn**	Ra
# contigs (>=1000pb)	251 627	20	**26**	–
N50	8 721	820 758	**750 435**	–
Largest contig	32 571	1 543 855	**1 531 123**	–
Execution time	23h28m23s	51m21s	**10h4m27s**	–
Total length	1 492 957 147	12 209 277	**12 207 471**	–
Genome fraction	99.52	95.26	**97.80**	–

Table 22: Quast results generated from various LRO assemblers with 100x of P4C2 Pacbio long reads (Saccharomyces cerevisae W303).

The absence of results from Ra assembler is caused by the high quantity of input data.

SLR assemblers

	DBG2OLC		**Spades**		Cerulean
Metrics	run 1	run 2	**run 1**	run 2	run1
# contigs (>=1000pb)	**103**	1 157	**188**	564	240
N50	**165 784**	16 175	**129 444**	37 681	190 934
Largest contig	**423 865**	68 633	**372 443**	145 780	638 854
Execution time	**46m35s**	2m22s	**6h9m44s**	24m30s	3h34m24s
Total length	**9 393 403**	11 458 559	**11 944 790**	11 518 863	12 497 420
Genome fraction	**68.8**	97.24	**98.5**	98.09	94.7

Table 23: Quast results generated from various SLR assemblers with 100x of P4C2 Pacbio long reads, contigs generated from short reads and Illumina short reads (Saccharomyces cerevisae W303).

5.13 Test 10: Saccharomyces cerevisae W303, ONT reads 20x

Datasets:

- ONT 2D reads corrected by lordec (20x coverage) : 4 7027 reads
- Illumina reads(MiSeq) : 3 815 678 reads of length 100pb, 65x
- Contigs generated by sparse assembler from Illumina reads (10 055 contigs), needed for hybrid assembly

LRO assemblers

Metrics	Celera	Falcon	Miniasm	Newbler
# contigs (>=1000pb)	894	–	169	35 687
N50	20 733	–	76 389	1 999
Largest contig	89 592	–	291 498	34 166
Execution time	2h54m32s	–	17s	2h5m46s
Total length	13 766 378	–	10 152 501	87 137 754
Genome fraction	87.9	–	68.99	96.8

Metrics	SGA	**Smartdenovo**	Abruijn	**Ra**
# contigs (>=1000pb)	37 932	**156**	–	**155**
N50	7 991	**84 777**	–	**127 765**
Largest contig	42 223	**222 924**	–	**349 350**
Execution time	9h32m51s	**2m18s**	–	**13m21s**
Total length	247 200 021	**10 456 229**	–	**12 573 431**
Genome fraction	98.8	**77.93**	–	**69.06**

Table 24: Quast results generated from various LRO assemblers with 20x of ONT long reads (Saccharomyces cerevisae W303).

The absence of results from Falcon and Abruijn is caused by the low coverage rate of long reads.

SLR assemblers

	DBG2OLC		**Spades**		Cerulean
Metrics	run 1	run 2	**run 1**	run 2	run1
# contigs (>=1000pb)	–	1 157	**268**	564	327
N50	–	16 175	**79 613**	37 681	94 617
Largest contig	–	68 633	**370 360**	145 780	361 438
Execution time	–	2m22s	**1h30m31s**	24m30s	3h6m41s
Total length	–	11 458 559	**11 917 307**	11 518 863	15 628 680
Genome fraction	–	97.24	**98.293**	98.09	93.78

Table 25: Quast results generated from various SLR assemblers with 20x of ONT long reads, contigs generated from short reads and Illumina short reads (Saccharomyces cerevisae W303).

The absence of results from DBG2OLC assembler results is caused by the low coverage rate of long reads.

5.14 Test 11: Caenorhabditis elegans, Pacbio reads 10x (P6-C4)

Datasets:

- Pacbio reads corrected by lordec (10x coverage) : 92 597 reads
- Illumina reads(MiSeq) : 55 070 232 reads of length 150pb, 165x
- Contigs generated by sparse assembler from Illumina reads (1 022 387 contigs), needed for hybrid assembly

LRO assemblers

Metrics	Celera	Falcon	Miniasm	Newbler
# contigs (>=1000pb)	45	–	1056	104079
N50	1166	–	77 770	1 999
Largest contig	2 181	–	499 152	5 415
Execution time	52m45s	–	2m7sec	18h42m21s
Total length	56 811	–	71 910 795	208 178 837
Genome fraction	0.048	–	12.3	45.5

Metrics	SGA	**Smartdenovo**	Abruijn	Ra
# contigs (>=1000pb)	–	**784**	39	–
N50	–	**151 698**	58 948	–
Largest contig	–	**546 464**	137 901	–
Execution time	–	**11m22s**	17m23s	–
Total length	–	**91 256 859**	2 391 267	–
Genome fraction	–	**13.16**	0	–

Table 26: Quast results generated from various LRO assemblers with 10x of P6C4 Pacbio long reads (Caenorhabditis elegans).

The absence of results from Falcon Ra and SGA is caused by the low coverage rate of long reads.

SLR assembler

	DBG2OLC		Spades		Cerulean
Metrics	**run 1**	run 2	run 1	run 2	run1
# contigs (>=1000pb)	**490**	17,569	–	10,330	–
N50	**428 709**	8 680	–	18 309	–
Largest contig	**1 392 481**	115 038	–	142 723	–
Execution time	**4h57m**	6h55m42s	–	15m34s	–
Total length	**103 432 617**	95 943 011	–	96 585 462	–
Genome fraction	**17.94**	88.1	–	93.06	–

Table 27: Quast results generated from various SLR assemblers with 10x of P6C4 Pacbio long reads, contigs generated from short reads and Illumina short reads (Caenorhabditis elegans).

The lack of results for the Spades and Cerulean assemblers is caused respectively by an insufficient available RAM and an excessive execution time.

5.15 Test 12: Caenorhabditis elegans, Pacbio reads 100x (P6-C4)

Datasets:

· Pacbio reads corrected by lordec (100x coverage): 740,776 reads

· Illumina reads (MiSeq) : 55,070,232 reads of length 150pb, 165x

· Contigs generated by sparse assembler with Illumina reads (1,022,387 contigs), needed for hybrid assembly

LRO assemblers

Metrics	Celera	Falcon	Miniasm	Newbler
# contigs (>=1000pb)	6 895	18 031	140	410 532
N50	58 720	18 602	1 921 931	1 999
Largest contig	726 905	82 715	5 921,636	297 726
Execution time	19d21h	37d20h	2h10m	22d20m
Total length	135 257 379	268 424 680	107 385 322	728 306 963
Genome fraction	86.8	0	89.33	98.7

Metrics	SGA	**Smartdenovo**	Abruijn	Ra
# contigs (>=1000pb)	–	**90**	–	–
N50	–	**2 249 996**	–	–
Largest contig	–	**4 716 467**	–	–
Execution time	–	**8h32m43s**	–	–
Total length	–	**107 109 382**	–	–
Genome fraction	–	**91.17**	–	–

Table 28: Quast results generated from various LRO assemblers with 100x of P6C4 Pacbio long reads (Caenorhabditis elegans).

The absence of results from Abruijn and Ra is caused by an error during assembly. The absence of results from SGA results the creation of a file containing all the non assembled long reads.

SLR assemblers

	DBG2OLC		Spades		Cerulean
Metrics	**run1**	run2	run1	run2	run1
# contigs (>=1000pb)	**435**	17 569	–	10 330	–
N50	**644 079**	8 680	–	18 309	–
Largest contig	**2 338 886**	115 038	–	142 723	–
Execution time	**11h2m52s**	6h55m42s	–	15min34s	–
Total length	**115 756 904**	95 943 011	–	96,585,462	–
Genome fraction	**90.96**	88.1	–	93.06	–

Table 29: Quast results generated from various SLR assemblers with 100x of P6C4 Pacbio long reads, contigs generated from short reads and Illumina short reads (Caenorhabditis elegans).

The lack of results for the Spades and Cerulean assemblers leads respectively to an insufficient available RAM and an excessive execution time.

6 Discussion

To measure the quality of the genome assembly, two metrics are fundamental: the number of contigs in the assembly (whose lengths are greater or equal to 1000 bp) and the genome fraction. Ideally, the number of contigs should be equal to the number of chromosomes and the genome fraction should be 100%. Therefore, the closer the number of contigs to the number of chromosomes and the genome fraction to 100% the better. However, notice that these two metrics are not necessarily correlated and that a genome fraction close to 100% can be obtained with a large number of small, badly assembled contigs. For this reason, we consider that the number of contigs must be given the priority when assessing the assembly quality. The assemblers are thus ranked, first, according to this parameter, then using the genome fraction and, finally, according to the execution time.

In this report, for practical purposes, we consider that the genome assembly is a success if the number of contigs is less than or equal to 3 times the number of chromosomes (admittedly, the choice of 3 is somewhat arbitrary) and if the genome fraction is larger than 85%.

Another point to take into consideration, is the number of times the assemblers fail to provide a result (for various reasons that are listed in the ”Results” section).

6.1 LRO assemblers

Table 30 shows the number of tests failed by the LRO assemblers.

LRO Assembler	# failures
Miniasm	0
Newbler	0
Smartdenovo	0
Celera	2
Abruijn	3
RA	3
SGA	4
Falcon	5

Table 30: Number of tests failed by the LRO assemblers.

Table 31 below shows the ranking of the different LRO assemblers for the 12 tests described in the ”Results” section. Assemblers are ranked, as indicated above, first according to the number of contigs, then according to the genome fraction.

rank	Test 1	Test 2	Test 3	Test 4	Test 5	Test 6
1	RA	**Abruijn**	Smartdenovo	**Abruijn**	Smartdenovo	**Abruijn**
2	Smartdenovo	**Smartdenovo**	RA	**Miniasm**	Abruijn	**Miniasm**
3	Abruijn	**Miniasm**	Miniasm	Smartdenovo	RA	Smartdenovo
4	Miniasm	RA	Celera	Celera	Miniasm	Falcon
5	Falcon	Falcon	Newbler	RA	Celera	RA
6	Newbler	Newbler	SGA	Falcon	SGA	Celera
7	*Celera*	*Celera*	*Falcon*	Newbler	Newbler	SGA
8	*SGA*	*SGA*	*Abruijn*	SGA	*Falcon*	Newbler

rank	Test 7	Test 8	Test 9	Test 10	Test 11	Test12
1	**Miniasm**	Abruijn	**Smartdenovo**	RA	Abruijn	Smartdenovo
2	**Smartdenovo**	RA	**Miniasm**	Smartdenovo	Celera	Miniasm
3	Abruijn	Miniasm	**Abruijn**	Miniasm	Smartdenovo	Celera
4	RA	Smartdenovo	Celera	Celera	Miniasm	Falcon
5	Celera	Celera	Falcon	Newbler	Newbler	Newbler
6	Newbler	SGA	Newbler	SGA	*SGA*	*Abruijn*
7	SGA	Falcon	SGA	*Abruijn*	*RA*	*SGA*
8	*Falcon*	Newbler	*RA*	*Falcon*	*Falcon*	*RA*

Table 31: Ranks of the LRO assemblers in the 12 tests. Assemblers in italic are those that did not provide a result for the corresponding test, assemblers in bold are those that provided a ”good” genome assembly (as defined above). Notice: all assemblers that failed the test have the same 8th rank.

In Test1, the 10x coverage provided by the long read is insufficient to assemble the genome whereas a 20x coverage (Test2) allows the best assemblers to obtain good genome assemblies (1 contig with genome fractions >91%). The same is true for the assembly of the *E. coli* k-12 genome, at least when reads from the PBS new chemistry P6-4 or ONT Minion are used (Tests 5, 6 7). A 100x coverage (Test 6 and 9) with the old PBS chemistry (P4-C2) allows the best LRO assemblers to obtain good assembly of the *E. coli* k-12 genome (1 or 2 contigs with genome fraction >96%) and *S. cerevisiae* genome (20-25 contigs with genome fractions >94% – recall that *S. cerevisiae* has 16 chromosomes).

Table 32 below shows the mean ranking and the number of "good" genome assemblies for the 8 LRO assemblers. Clearly, there are 3 groups. The first group {Smartdenovo, Miniasm, Abruijn} provides the best results, the second group {RA, Celera} provides intermediate results and the last one {Newbler, Falcon, SGA} provides the worst results. In the first group, the mean ranking of Abruijn would be better if this assembler did not fail to provide a result for 25% of the tests.

LRO Assembler	mean rank	# successes
Smartdenovo	2.08	3
Miniasm	2.75	5
Abruijn	3.33	4
RA	4.25	0
Celera	4.67	0
Newbler	6.17	0
Falcon	6.33	0
SGA	7.08	0

Table 32: Mean rank of the LRO assemblers and number of "good" genome assembly they provide.

6.2 SLR assemblers

In the tables of the "Results" section, run1 and run2 correspond, respectively, to the genome assembly using both long and short reads, and the same genome assembly using only the short reads. In all observed cases, combining long and short reads improve the results, sometimes considerably. It must be noted that adding the long reads improves coverage. However, this addition is relatively marginal (10x or 20x), except for tests 9 and 12 where a 100x coverage is added with the long reads.

Spades and Cerulean fail to provide a result for the largest genome (*C. elegans*) due to a lack of RAM or excessive running time.

Table 33 shows the results of the 3 SLR assemblers for the 12 tests. SLR assemblers, in terms of genome assembly, are globally less efficient than LRO assemblers (their only "success" is for the assembly of the genome of *Acinobacter* sp. ADP1).

rank	Test 1	Test 2	Test 3	Test 4	Test 5	Test 6
1	**DBG2OLC**	**Spades**	Cerulean	Cerulean	Spades	Spades
2	**Spades**	**DBG2OLC**	Spades	Spades	Cerulean	Cerulean
3	Cerulean	Cerulean	DBG2OLC	DBG2OLC	DBG2OLC	DBG2OLC

rank	Test 7	Test 8	Test 9	Test 10	Test 11	Test12
1	Cerulean	DBG2OLC	DBG2OLC	Spades	DBG2OLC	DBG2OLC
2	Spades	Spades	Spades	Cerulean	*Cerulean*	*Cerulean*
3	DBG2OLC	Cerulean	Cerulean	*DBG2OLC*	*Spades*	*Spades*

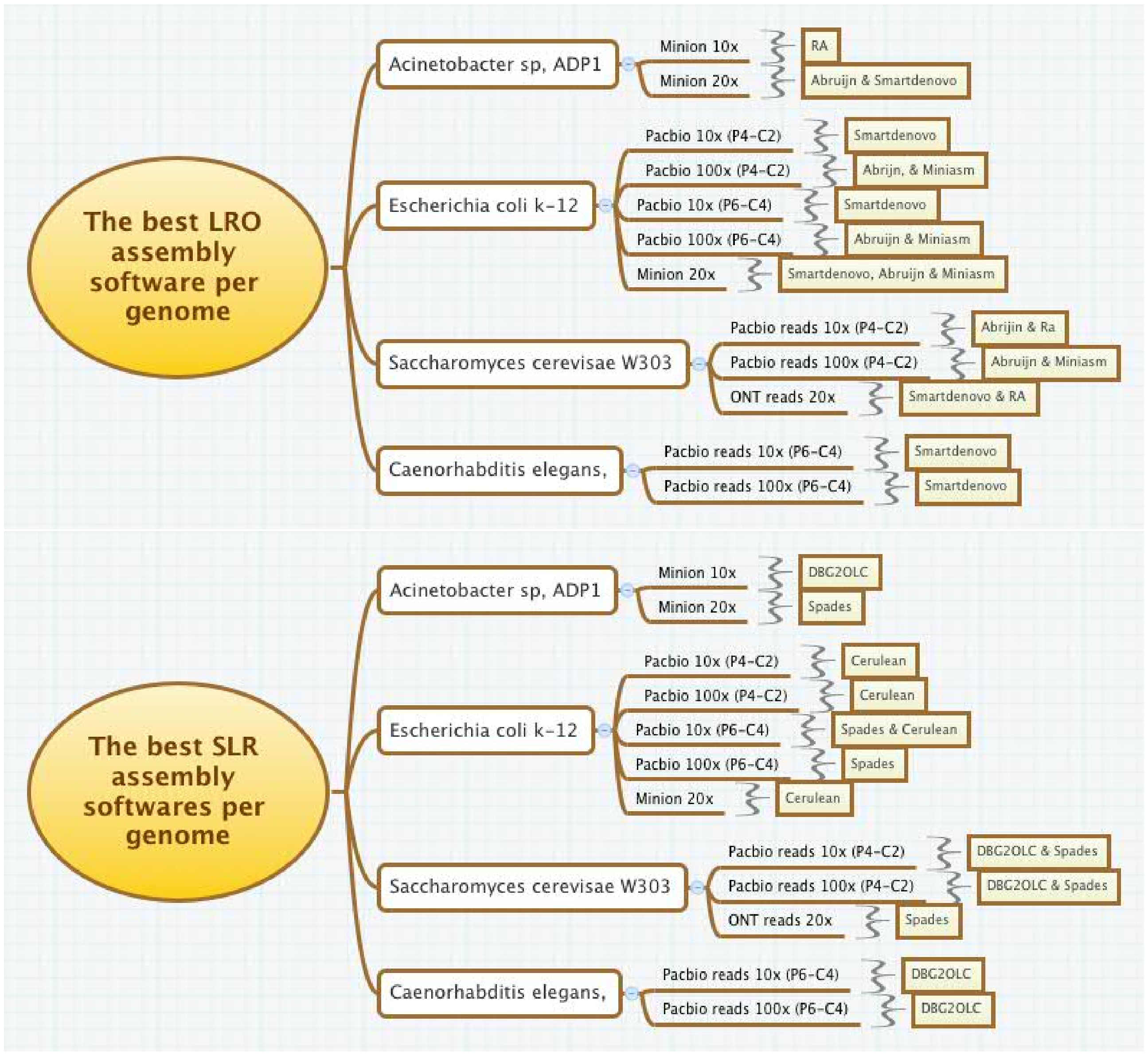
Table 33: Ranks of the SLR assemblers in the 12 tests. Assemblers in italic are those that did not provide a result for the corresponding test, assemblers in bold are those that provided a "good" genome assembly (as defined above). Notice: all assemblers that failed the test have the same 8th rank.

Table 34 shows the mean ranking of the SLR assemblers and their number of successes. Cerulean appears slightly less efficient than Spades or DBG2OLC.

SLR Assembler	mean rank	# successes
DBG2OLC	2.08	2
Spades	2.08	2
Cerulean	2.25	0

Table 34: Mean rank of the SLR assemblers and number of "good" genome assembly they provide.

Figure 3. **Assembly performance per species in different TGS platforms and under different conditions. The best LRO and SLR assemblers are displayed in the boxes for every condition type.**



7 Conclusion

In this report, we have discussed the best practices in long reads assembly. Ideally the results of different assembly performance should help the researchers choose the best software, taking into consideration the nature of the available input data. We chose model organisms with robust previous assembly genomes and we compared them in different conditions: TSG platforms, coverages and polymerase (Figure 3). However, in the case of LRO assemblers when we started with moderate long reads coverage, Smartdenovo software performed the best, fast and often gives low number of contigs and accurate genome result. Nevertheless, in small bacterial genomes, Abrujin software gave also very good results.
As far as SLR are concerned, Spades and Cerulean software performed better in bacterial genomes, but in eukaryote genomes DBG2OLC software gave the best results.
Finally, we should improve the same performance software analysis in other model organisms with a larger genome size and greater content in transposable elements, such as plants or vertebrates, to complete the guideline practices for software genome assemblies. Include others long read correction software before the assembly, it's also strongly recommended.

References

1. The_International_Human_Genome_Sequencing_Consortium. Initial sequencing and analysis of the human genome. *Nature 409:860–921* (2001).
2. Venter JC, *et al.*. The sequence of the human genome. *Science 291(5507):1304-51* (2001).
3. 1000_Genomes_Project_Consortium. An integrated map of genetic variation from 1092 human genomes. *Nature 491:56-65* (2012).
4. MetaHIT_Consortium. An integrated catalog of reference genes in the human gut microbiome. *Nat Biotechnol. 32(8):834-841* (2014).
5. Schatz MC, *et al.*. Assembly of large genomes using second- generation sequencing. *Genome research 20, 1165-1173* (2010).
6. Nagarajan N, P. M. Sequence assembly demystified. *Nat Rev Genet 14:157–167* (2013).
7. Loman NJ, *et al.*. A complete bacterial genome assembled de novo using only nanopore sequencing data. *Nat Methods 12(8):733-5* (2015).
8. Koren S, *et al.*. Reducing assembly complexity of microbial genomes with single- molecule sequencing. *Genome Biology 14(9): R101* (2013).
9. Rhoads A, *et al.*. Pacbio sequencing and its applications. *Genomics Proteomics Bioinformatics; 13(5): 278–289.* (2015).
10. Chen X, *et al.*. The architecture of a scrambled genome reveals massive levels of genomic rearrangement during development. *Cell 158, 1187-1198* (2014).
11. Cao H, *et al.*. Rapid detection of structural variation in a human genome using nanochannel-based genome mapping technology. *Gigascience 3, 34* (2014).
12. Chaisson MJ, *et al.*. Resolving the complexity of the human genome using single- molecule sequencing. *Nature 517, 608-611* (2015).
13. Sergey Koren, *et al.*. Hybrid error correction and de novo assembly of single-molecule sequencing reads. *Nature Biotechnology 30(7):693-700* (2012).
14. Chin, J. Falcon genome assembly tool kit manual. `https://github.com/PacificBiosciences/FALCON/wiki/Manual` (Jan. 2016).
15. Li, H. Minimap and miniasm: fast mapping and de novo assembly for noisy long sequences. *arXiv:1512.01801.* (2015).
16. Margulies M, *et al.*. Genome sequencing in microfabricated high-density picolitre reactors. *Nature 2005, 437: 376–380.* (2005).
17. Simpson JT, D. R. Efficient de novo assembly of large genomes using compressed data structures. *doi: 10.1101/gr.126953.111 Genome Res.* (2011).
18. Ruan, J. Readme. `https://github.com/ruanjue/smartdenovo` (Mar. 2016).
19. Yu Lin, *et al.*. Assembly of long error-prone reads using de bruijn graphs. *http://dx.doi.org/10.1101/048413* (2016).
20. Ivan Sović, *et al.*. Fast and sensitive mapping of nanopore sequencing reads with graphmap. *Nature Communications 7,Article number:11307;doi:10.1038/ncomms11307* (2016).
21. Chengxi Ye, *et al.*. Dbg2olc: Efficient assembly of large genomes using the compressed overlap graph. *arXiv:1410.2801.* (2015).
22. Anton Bankevich, *et al.*. Spades: A new genome assembly algorithm and its applications to single-cell sequencing. *Journal of Computational Biology. 19(5).* (2012).
23. Viraj Deshpande, *et al.*. Cerulean: A hybrid assembly using high throughput short and long reads. *arXiv:1307.7933.* (2013).
24. Salmela L., R. E. Lordec: accurate and efficient long read error correction. *Bioinformatics. 2014;30(24):3506–14. doi: 10.1093/bioinformatics/btu538 pmid:25165095; PubMed Central PMCID: PMC4253826.* (2014).

25. Alexey Gurevich, *et al.*. Quast: quality assessment tool for genome assemblies. *Bioinformatics 29(8), 1072-1075.* (2013).

26. Delcher AL, *et al.*. Alignment of whole genomes. *Nucleic Acids Res. 1999 Jun 1;27(11):2369-76. PMID:10325427 ; PMCID:PMC148804* (1999).