Master's Thesis

University of Rennes 1
Bioinformatics and genomics Master's Degree
(2014 - 2015)

Test and benchmarking of a new scaffolding methodology

Institute for Research in IT and Random Systems, GenScale
263 Avenue General Leclerc, 35000 Rennes, France

Author:
Alexandrina Bodrug

Supervisors:
Pr. Univ. Rennes 1 Rumen Andonov
Dr. CNRS Dominique Lavenier

22 June, 2015

Thanks

Abbreviations & github link

- *Eucalyptus globulus* chloroplastic genome
- *Acorus calamus* chloroplastic genome
- *Atropa belladonna* chloroplastic genome
- *Agrostis stolonifera* chloroplastic genome
- *Cucumis sativus* chloroplastic genome
- *Lecomlella madagascariensis* chloroplastic genome
- *Oenothera elata* chloroplastic genome
- *Pinus koraiensis* chloroplastic genome
- *Euglena gracilis* chloroplastic genome
- *Oryza sativa Japonica* chloroplastic genome

Contents

1 Introduction

1.1 Backgroud

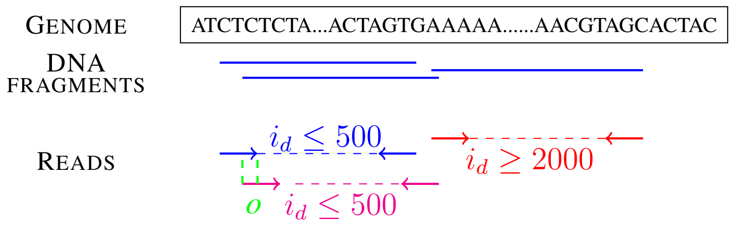
De novo assembly is the process which pieces together overlapping small fragmented DNA sequences produced by Next Generation Sequencing methods into larger sequences. The aim is to obtain complete genomes (or chromosomes) containing gaps of known lengths because the less fragmented the genome is, the easier the downstream analysis are[1]. However an incomplete assembly is still sufficient for most of the analysis performed on DNA which explains why databases mainly contain partially assembled genomes. Nonetheless the uninterrupted genome sequence is a precious information and there has been an important effort made to improve the performance of assembly algorithms and the quality of NGS data. The the detailed process of assembly is described in subsection 1.2 Assembly terminology; the two main steps are building contigs from reads (sometimes referred to as assembly) and scaffolding, the ordering and relative orientation of contigs or unitigs. The 2011 and 2013 Assemblathon projects[2][3] aimed at benchmarking existing assembly tools with high coverage diploid genomes. The studies focused mainly on the contig building step, concluding that although many tools found quality assemblies, the tool and quality criteria should be adjusted to the type of genome and the goal of the assembly project. For example a good N50, an extensively used metric which is the contig length such that using equal or longer contigs produces half the bases of the genome, is not essential in a gene detecting assembly project.
The first stand-alone scaffolder named Bambus[4], originally part of the MetAMOS[5] assembly and analysis pipeline, was published in 2004. Previously the scaffolding step was missing or presented as an option within conting builders, for instance the Velvet[6] assembler *'scaffolding yes or no'* option. In the 2014 comprehensive evaluation of scaffolding tools[7], Hunt *et al* found that no tool identified more than 90% of joins between real-data Velvet assembled contigs, meaning genomes were still fragmented into many scaffolds as joins were missing for a complete and accurate ordering and orientation. The study also used simulated data highlighting the fact that perfect data doesn't always yield perfect results. Despite its simply formulated goal - order and orient contigs - scaffolding is a challenging computational problem. It was first described and modeled in 2002 by Hudson *et al.*[8] which proposed a greedy path-merging strategy, described in subsection 1.3 A history of scaffolding strategies along with other proposed algorithms.

1.2 Assembly terminology

In this report *assembly* will refer to the whole multi-step process which starts from once filtered out-of-the-sequencer data and results, in the best case scenarios, in highly uninterrupted sequence of a genome or chromosome. As previously mentioned, the two main steps are contig/unitig building and contig/unitg scaffolding. The difference between contig and unitig is fundamental to understanding the Genscale scaffolding challenges. Another key point is the construction of joins between contigs/unitigs - also referred to as links, edges, bonds …

1.2.1 Reads, pairing and overlaps

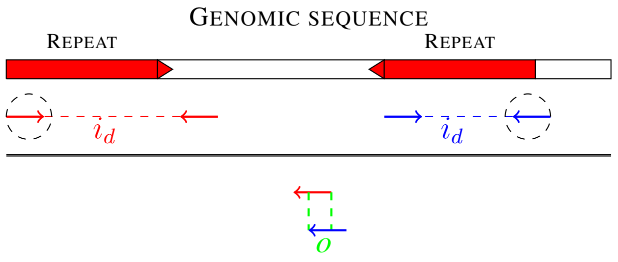
A read is a short ($< 500pb$) copy of a DNA fragment of known length and nucleic acid order. It is produced differently depending of the sequencing technology. Paired reads are copies of the two extremities of a DNA molecule. The DNA sequence between two reads of a pair is called an insert. The size of the insert is variable. Reads with small insert sizes ($< 500bp$) are called paired-end reads. Mate-paired reads are reads whose insert size is very big (up to tens of kilobases). The pairing information and the size of the insert are provided by the sequencer. A collection of reads with their associated insert size is called a a genomic library.
Figure 1 represents three pairs of reads. Within the pairs, reads are facing each other: this configuration is



Each end of a DNA molecule is cloned to produce paired reads. Here is represented a mate-paired pair (red) with a big insert size *(i_d)* and two paired-end pairs (blue and magenta) which slightly overlap *(o)*.

Figure 1: Alignment of paired reads on fragmented DNA

called *Forward-Reverse* read orientation. To be sequenced the genome represented in figure 1 is first amplified by Polymerase Chain Reaction and then fragmented into numerous DNA molecules by sonication or nebulization. Each end of the molecule is then cloned. Overlapping of reads occurs when two reads sequence a portion of the same genomic region, but not only. The overlapping concept implies a common origin but unfortunately overlapping can occur if two reads sequence two different repeated genomic regions. Figure 2 shows how repeated regions create false positive overlaps. Such reads can be detected and filtered out by ignoring high-frequency overlaps (higher than the coverage at which the genome was sequenced). However this can result in false negatives and makes the task of assembling repeated regions very hard.

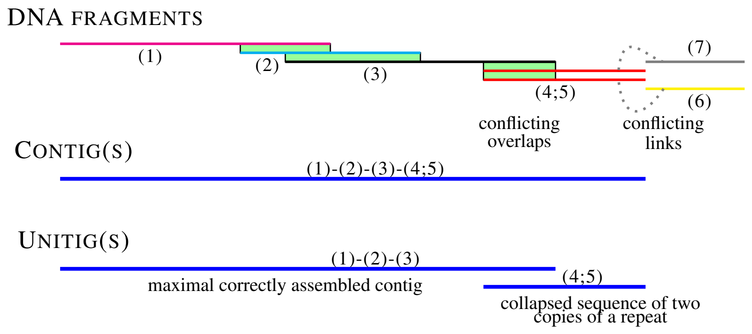


The two circled reads will have a significantly long and accurate overlap to imply a common genomic origin when in fact they come from distant regions.

Figure 2: Overlapping induced by repeated sequences

1.2.2 Unitigs and Contigs

Unitigs are an uniquely assemblable subset of overlapping fragments. At the end of an unitig data shows multiple dubious overlaps as seen in subsubsection 1.2.1 Reads, pairing and overlaps creating joins with multiple other unitigs. Contigs are larger than unitigs, extended through repeat boundaries but are still ungapped sequences. Contigs are interesting to construct because there is a higher chance to detect genes. Taking the example shown in figure 2, a contig will merge the first three DNA fragments and will then be extended though the ambiguous overlaps, merging the red DNA fragments' sequence. Unitigs however will stop at the end of the third DNA fragment and assemble the red fragments separately. In a sense, unitigs are either an unambiguous contig or a compression of several copies of a repeat. The advantage of working with unitigs is that there are less chances of erroneous merging of two far away genomic regions. This feature is used in the Genscale scaffolding strategy, further discussed in subsection 2.1 Genscale scaffolding methodology.

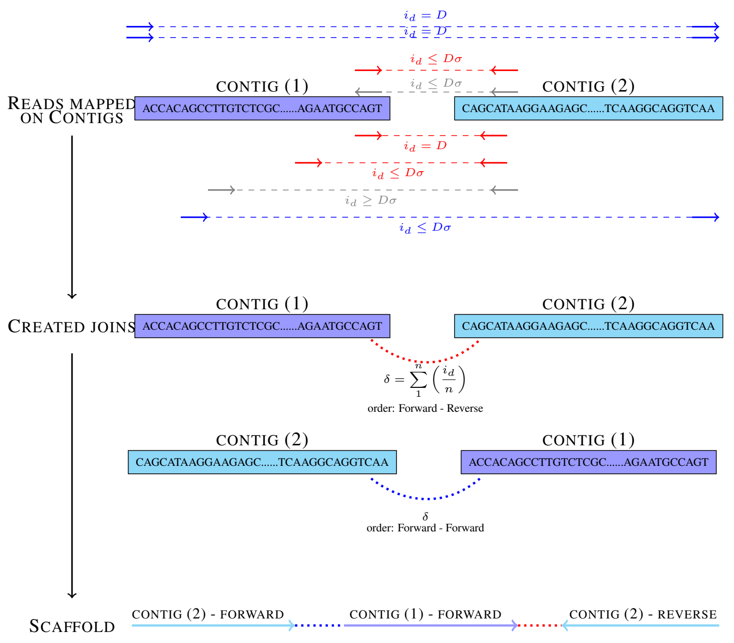


Unitigs end at multiple overlaps indicating a possible repeat. Contigs can be extended through conflicting overlaps. Here, the red DNA fragments are two copies of a repeat. When no more overlaps exist, contigs can be linked (gray dotted line) thanks to information provided by read pairs. This is the scaffolding task. Here alternative paths are possible due to the repeated region.

Figure 3: The difference between unitigs and contigs

1.2.3 Obtaining scaffolds

A scaffold is a linear ordering of contigs (or unitigs). The ordering and relative orientation of contigs is possible thanks to paired reads information. The first step of scaffolding is mapping reads on the previously constructed contigs: the two most used mappers are bwa[9] and bowtie[10;11]. A pair of reads mapping on two different contigs provide a join, which holds the information of distance between the two contigs, and relative orientation (see figure 4). A same contig can have joins with multiple other contigs (see CONFLICTING JOINS figure 3 and CREATED JOINS figure 4). These multiple joins which result in multiple paths when ordering and orienting contigs are solved differently by scaffolders. Most of the time, a choice is made - heuristically - to use one join over another. Another strategy is illustrated in figure 4, where the contig with the two high-confidence joins is duplicated. This is the strategy of the Genscale scaffolders, further described in subsection 2.1 Genscale scaffolding methodology.
The concept of insert size is essential to understand the challenges of scaffolding when dealing with repeated regions in genomes. Take CONTIG (1) and CONTIG (2) in figure 4: the two contigs are separated by a gap of undefined size. Say this gap is caused by a repeated region where reads mapped multiple times and were thus discarded, or the region wasn't sequenced by the sequencer and is all together absent. If the region is bigger than the insert size, no read pair will span over it. The join between contig (1) and contig (2) will not exist. This explains why mate-pair information is extremely useful. When multiple mate-pair libraries with different big insert size are available, genomically distanced regions can directly be ordered and orientated. The scaffolds will potentially be longer, as missing or ambiguous data doesn't impede its construction.



i_d : insert size , D : expected insert size , σ : distance standard deviation , n : number of retained correctly mapped paired reads for the join, δ estimated distance.

Paired reads are mapped on previously assembled contigs. Pairs with reads mapping on different contigs provide linking information. The library represented is Forward - Reverse (reads are facing each other, see red pairs). Additional read pairs with a satisfying i_d but with a different orientation can coexist with the Foward - Reverse pairs (here, blue pairs, Forward - Forward). Pairs which map with a big insert size ($\geq D3\sigma$) are usually discarded (here, gray pairs). Red and blue pairs are retained to create two conflicting joins between CONTIG (1) and CONTIG (2). The conflict can be solve with a scaffolder which duplicated CONTIG (2) allowing both joins to coexist, or ignoring one of the joins (not represented).

Figure 4: Creating joins between contigs thanks to read pair information

1.3 A history of scaffolding strategies

The scaffolding problem was first introduced in 2002 by Hudson *et al.*[8] following the challenges which arose during the human genome clone-by-clone sequencing by *Lander et al.*[12] and the human whole genome shotgun assembly project by Venter *et al.*[13], both published in 2001. The Hudson *et al.* paper defines the problem as follows: "The *Contig Scaffolding Problem* is to order and orientate the given contigs in a manner that is consistent with as many mate-pairs as possible". The problem is modeled as a graph where vertices represent contigs and links represent bundles of pairs of reads joining two contigs (see figure 4, red and blue pairs are bundled into two joins which will be represented by a link in the graph). This model is reused in most of the stand alone scaffolding tools.

1.3.1 Examples of scaffolding tools using heuristics

Hudson et al, SSPACE, Bambus

1.3.2 Examples of scaffolding tools solving the problem exactly

Opera, SGA, SOPRA

1.3.3 Characteristics of the Genscale scaffolding tools

Several models have been developed and tested. Among them: distance based and weighted path models. A new multi-step called flow model is in development. Each processes the same type of data which is different than what is given to the previously described scaffolders.

1.4 Goal of the internship project

The goal of this report is present the performance of the Genscale scaffolding tools (GST) with reliable simulated data and some instances of real data. The first step is to compare the solutions obtained by GST with available solutions. The second step is to benchmark the GST against published tools. In order to achieve this task, a benchmarking workflow was set up and is described in subsection 2.3 Benchmarking of section 2 Material and methods. The data format and data type are described in subsection 2.1 Genscale scaffolding methodology of section 2 Material and methods. ...

2 Material and methods

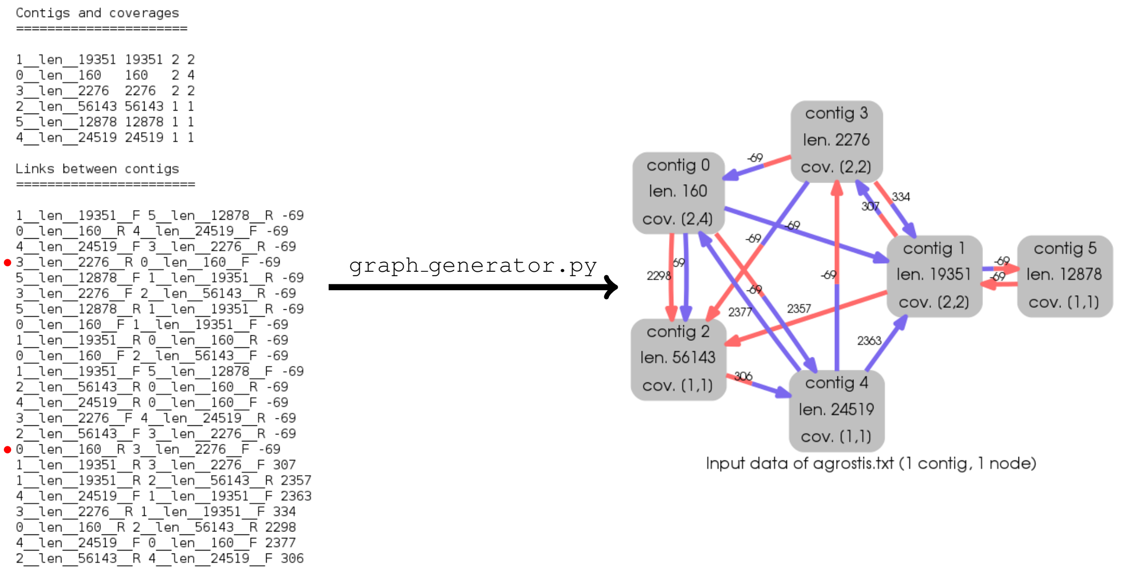
2.1 Genscale scaffolding methodology

2.1.1 Format of the input data for genscale scaffolders

GST model the scaffolding problem as a graph where vertices are unitigs - *not contigs* - and links (directed edges) are bundles of paired reads joining two unitigs. An example of file read by solvers is showed in figure 5 along with its graphical representation generated by the `graph_generator.py` script. The number of links in the *.txt* file is higher than the number of links drawn in the graph because the script merges reverse-equivalent links.

What is a reverse-equivalent link?
Take unitig 0 and unitig 3 of figure 5. In the *.txt* file link list there are two links: `(3R -> 0F -69)` and `(0R -> 3F -69)`, which in fact represents the same link between the two unitigs.

```
                  Reverse-equivalent links
(3 Reverse -> 0 Forward) ≡ (0 Reverse -> 3 Forward)
(3 Forward -> 0 Reverse) ≡ (0 Forward -> 3 Reverse)
(3 Forward -> 0 Forward) ≡ (0 Reverse -> 3 Reverse)
(3 Reverse -> 0 Reverse) ≡ (0 Forward -> 3 Forward)
```



```
Contigs and coverages
======================

1__len__19351 19351 2 2
0__len__160   160   2 4
3__len__2276  2276  2 2
2__len__56143 56143 1 1
5__len__12878 12878 1 1
4__len__24519 24519 1 1

Links between contigs
=======================

1__len__19351__F 5__len__12878__R -69
0__len__160__R 4__len__24519__F -69
4__len__24519__F 3__len__2276__R -69
● 3__len__2276__R 0__len__160__F -69
5__len__12878__F 1__len__19351__R -69
3__len__2276__F 2__len__56143__R -69
5__len__12878__R 1__len__19351__R -69
0__len__160__F 1__len__19351__F -69
1__len__19351__R 0__len__160__R -69
0__len__160__F 2__len__56143__F -69
1__len__19351__F 5__len__12878__F -69
2__len__56143__R 0__len__160__R -69
4__len__24519__R 0__len__160__F -69
3__len__2276__F 4__len__24519__R -69
2__len__56143__F 3__len__2276__R -69
● 0__len__160__R 3__len__2276__F -69
1__len__19351__R 3__len__2276__F 307
1__len__19351__R 2__len__56143__R 2357
4__len__24519__F 1__len__19351__F 2363
3__len__2276__R 1__len__19351__F 334
0__len__160__R 2__len__56143__R 2298
4__len__24519__F 0__len__160__F 2377
2__len__56143__R 4__len__24519__F 306
```

graph_generator.py
contig 0 len. 160 cov. (2,4)
contig 3 len. 2276 cov. (2,2)
contig 1 len. 19351 cov. (2,2)
contig 5 len. 12878 cov. (1,1)
contig 2 len. 56143 cov. (1,1)
contig 4 len. 24519 cov. (1,1)
-69
307
334
2298
2377
2357
2363
306
Input data of agrostis.txt (1 contig, 1 node)

The *.txt* file contains a list of unitigs with their associated length and coverage. When the coverage is > 1 the unitig is a repeated sequence. The advantage of using unitigs and not contigs is that the associated coverage is easier to determine (for long unitigs) and more trustworthy. For small unitigs, the coverage is an interval (here, the 160 base unitig 0). The joins between unitigs are orientated and contain the contig orientation (*Forward* or *Reverse*) information. The distance of each link is negative when two unitigs overlap or positive when two unitigs are separated by a gap. Positive distances (gaps) are obtained though mate-pair information. In the graph orientations are represented by color: red for *Reverse* and blue for *Forward*.

Figure 5: Input data of *Agrostis stolonifera* chloroplast genome in *.txt* format and its graph representation

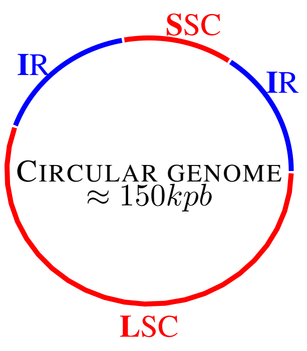
How is the *.txt* file obtained?
The gaps (positive distance links) are obtained by mapping mate-paired reads on the unitigs as seen in figure 4. Overlaps are computed separately. GST are provided a ready list of links while most of the scaffolders (SSPACE, SOPRA ...) join construction is the first step of the scaffolder. Unitig coverage is obtained by mapping paired reads. The bigger the contig, the more robust the mapping information is - hence the intervals for small unitigs.

2.1.2 Features of the assembled genomes

As the aim of the Genscale scaffolding project is to produce a complete genome rising to the challenge of repeated sequences, chloroplastic and bacterial genomes are well suited to test the performances of the GSTools. Moreover, chloroplastic and bacterial genomes are small enough to enable a detailed assessment and benchmarking of found solutions.

Chloroplasts

Chloroplasts are small organelles in plant photosynthetic tissues which possess their own DNA. The chloroplast genomes are small ($\approx 150kpb$), circular and have a large inverted repeated sequence of around $25kpb$. Some instances used in this study lack this repetition. This is the case in *Pinus koraiensis* and *Euglena gracilis*. However these two genomes possess significantly more small ($<$ 20bp) repeated sequences.

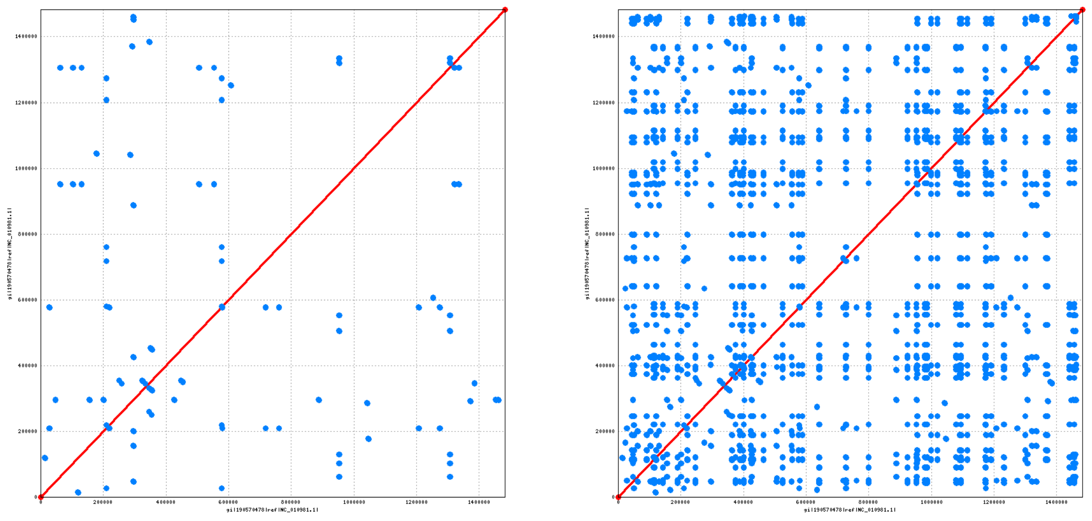


Inverted Repeat (IR $\approx 23kpb$) ; Long Single Copy (LSC); Small Simple Copy (SSC $\approx 85kpb$)

Figure 6: Chloroplast genome structure

Bacteria and other

Bacterial genomes are bigger ($>$ 1Mpb) and contain many small repeated sequences between 500pb and 1000pb.



Notes

Figure 7: Wolbachia endosymbiont genome dotplotted against itself

2.1.3 Genscale scaffolding strategies

As seen in figure 5, the input data can be visualized by the `graph_generator.py` script as a graph where nodes uniquely represent unitigs regardless of their coverage. However, this visualization is useful for a first human assessment of the data. Within models, each contig is multiplied by the number of occurrences. The total number of occurrences is then duplicated to model the *Forward* and *Reverse* orientation. With the number of contig occurrences and orientations, the number of links increases as well. However no duplicate links are allowed (merged) and for each link its reverse equivalent is created. A simple example of the difference between the raw input graph and the processed input graph that the GST solve is presented in figure 8.

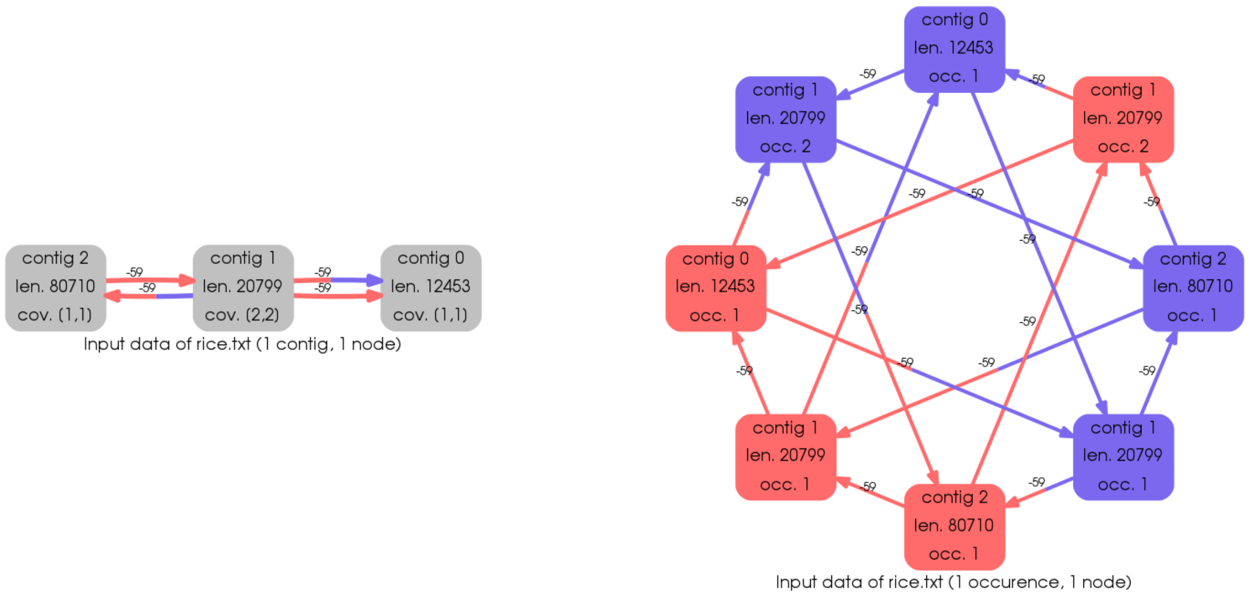


Figure 8: Input graphs of eucalyptus, as observed in the txt file and as processed by the Genscale scaffolding models

2.2 Testing GST and search for the holy grail of explanations

Golden standard et comparison between data etc

2.3 Benchmarking

2.3.1 Published scaffolders chosen for benchmarking

Why i chose SSPACE?

2.3.2 Benchmarking strategy

Draw a workflow Chosen scaffolding tools to benchmark against Benchmarking workflow Comparisons QUAST Comparison function Visualization MUMMER Visualization tool

3 Results

3.1 Comparison between the data sets

3.1.1 Input data sets

Results presented in table 1 are those of the complete model-like graphs.

organism	G type	G size	#nodes	#edges	node degree	betweeness centrality	degree centrality
agrostis	chpl.	size	22	106	min 4, max 13, avg 9	c4R, c4F, c1R	c2F, c2R, c4F
acineto	bacter	size	924	9288	min 0, max 183, avg 20	c121F, c35R, c35F	c62R, c143R, c25R
acorus	chpl.	size	30	204	min 4, max 26, avg 13	c5R, c1F, c3F	c5R, c3F, c3R
atropa	chpl.	size	52	288	min 3, max 19, avg 11	c0R, c14F, c14R	c0R, c2R, c11R
cucumis	chpl.	size	194	1790	min 4, max 72, avg 18	c17F, c17R, c47R	c17F, c17R, c26R
eucalyptus	chpl.	size	8	16	min 2, max 6, avg 4	c2R, c1F, c0F	c2R, c1F, c0F
euglena	chpl.	size	296	11894	min 8, max 225, avg 80	c24F, c26F, c24R	c24F, c36R, c36F
lecomtella	chpl.	size	22	90	min 2, max 17, avg 8	c1R, c4R, c1F	c1R, c0R, c0F
oenothera	chpl.	size	172	3222	min 3, max 82, avg 37	c9F, c13F, c-9R	c28R, c28F, c18R
pinus	chpl.	size	122	658	min 3, max 38, avg 10	c0F, c1R, c0R	c0F, c1R, c0R
rice	chpl.	size	8	16	min 4, max 4, avg 4	c2R, c0F, c1F	c2F, c1F, c1R
sacchar.	chr3	size	370	4416	min 4, max 139, avg 23	c45F, c58R, c21F	c21F, c58R, c18R
wolbachia	bacter	size	-	-	min -, max -, avg -	c-, c-, c-	c-, c-, c-

Table 1: Graph complexity

3.1.2 Input data inconsistencies detector

`input_inspector.py`

3.2 Comparison of GST solutions with the expected solution

organism	expected solution found	time	partial solution found	time	note
agrostis	-	-	-	-	-
acineto	-	-	-	-	-
acorus	-	-	-	-	-
atropa	-	-	-	-	-
cucumis	-	-	-	-	-
eucalyptus	-	-	-	-	-
euglena	-	-	-	-	-
lecomtella	-	-	-	-	-
oenothera	-	-	-	-	-
pinus	-	-	-	-	-
rice	-	-	-	-	-
sacchar.	-	-	-	-	-
wolbachia	-	-	-	-	-

Table 2: Genscale scaffolders solutions

3.2.1 Weighted path method

3.2.2 Distance based

3.2.3 Flow model

3.3 Comparison between best GST solution and SSPACE

3.3.1 QUAST and comparison function

Find similar table for rice, pinus and wolbachia in Annexes.

Assembly	wpm_sol1.fsa	wpm_sol2.fsa	wpm_sol3.fsa	wpm_sol4.fsa	dist_sol	sspace_sol	ref_genome
# contigs ($\geq$ 0 bp)	**1**	**1**	**1**	**1**	2	3	**1**
# contigs ($\geq$ 1000 bp)	**1**	**1**	**1**	**1**	**1**	2	**1**
Total length ($\geq$ 0 bp)	136584	136584	136584	136584	**137641**	115344	136584
Total length ($\geq$ 1000 bp)	136584	136584	136584	136584	**137145**	115184	136584
# contigs	**1**	**1**	**1**	**1**	**1**	3	**1**
Largest contig	136584	136584	136584	136584	**137145**	102256	136584
Total length	136584	136584	136584	136584	**137145**	115344	136584
Reference length	136584	136584	136584	136584	136584	136584	136584
GC (%)	38.45	38.45	38.45	38.45	38.45	37.39	38.45
Reference GC (%)	38.45	38.45	38.45	38.45	38.45	38.45	38.45
N50	136584	136584	136584	136584	**137145**	102256	136584
NG50	136584	136584	136584	136584	**137145**	102256	136584
N75	136584	136584	136584	136584	**137145**	102256	136584
NG75	136584	136584	136584	136584	**137145**	12928	136584
L50	1	1	1	1	1	1	1
LG50	1	1	1	1	1	1	1
L75	1	1	1	1	1	1	1
LG75	**1**	**1**	**1**	**1**	**1**	2	**1**
# misassemblies	**0**	2	1	1	2	1	**0**
# misassembled contigs	**0**	1	1	1	1	1	**0**
Misassembled contigs length	**0**	136584	136584	136584	137145	102256	**0**
# local misassemblies	**0**	**0**	**0**	**0**	1	**0**	**0**
# unaligned contigs	0 + 0 part	0 + 0 part	0 + 0 part	0 + 0 part	0 + 0 part	0 + 0 part	0 + 0 part
Unaligned length	0	0	0	0	0	0	0
Genome fraction (%)	**100.000**	**100.000**	98.657	98.657	91.522	84.590	**100.000**
Duplication ratio	1.000	1.155	1.014	1.014	1.097	**0.998**	1.000
# N's per 100 kbp	**0.00**	**0.00**	**0.00**	**0.00**	218.75	12.14	**0.00**
# mismatches per 100 kbp	0.00	0.00	0.00	0.00	0.00	0.00	0.00
# indels per 100 kbp	**0.00**	**0.00**	**0.00**	**0.00**	5.60	0.87	**0.00**
# genes	132 + 1 part	**133 + 0 part**	131 + 0 part	131 + 0 part	122 + 1 part	114 + 2 part	**133 + 0 part**
Largest alignment	**136584**	81612	82913	134750	102364	81140	**136584**
NA50	**136584**	81612	82913	134750	102364	81140	**136584**
NGA50	**136584**	81612	82913	134750	102364	81140	**136584**
NA75	**136584**	53138	53671	134750	22817	21116	**136584**
NGA75	**136584**	53138	53671	134750	22817	12928	**136584**
LA50	1	1	1	1	1	1	1
LGA50	1	1	1	1	1	1	1
LA75	**1**	2	2	**1**	2	2	**1**
LGA75	**1**	2	2	**1**	2	3	**1**

Table 3: All statistics are based on contigs of size $\geq$ 50 bp, unless otherwise noted (e.g., "# contigs ($\geq$ 0 bp)" and "Total length ($\geq$ 0 bp)" include all contigs).

Assembly	wpm_sol1.fsa	sspace_sol	flow_step1_sol	ref_genome
# contigs ($\geq$ 0 bp)	**1**	21	2	**1**
# contigs ($\geq$ 1000 bp)	1	1	1	1
Total length ($\geq$ 0 bp)	116866	**119654**	116214	116866
Total length ($\geq$ 1000 bp)	116866	**117042**	115838	116866
# contigs	**1**	21	**1**	**1**
Largest contig	116866	**117042**	115838	116866
Total length	116866	**119654**	115838	116866
Reference length	116866	116866	116866	116866
GC (%)	38.80	38.83	38.86	38.80
Reference GC (%)	38.80	38.80	38.80	38.80
N50	116866	**117042**	115838	116866
NG50	116866	**117042**	115838	116866
N75	116866	**117042**	115838	116866
NG75	116866	**117042**	115838	116866
L50	1	1	1	1
LG50	1	1	1	1
L75	1	1	1	1
LG75	1	1	1	1
# misassemblies	11	2	1	**0**
# misassembled contigs	1	1	1	**0**
Misassembled contigs length	116866	117042	115838	**0**
# local misassemblies	4	5	4	**0**
# unaligned contigs	0 + 0 part	0 + 0 part	0 + 0 part	0 + 0 part
Unaligned length	0	0	0	0
Genome fraction (%)	99.471	99.892	92.153	**100.000**
Duplication ratio	1.005	1.025	1.076	**1.000**
# N's per 100 kbp	**0.00**	741.30	6997.70	**0.00**
# mismatches per 100 kbp	5.16	0.86	**0.00**	**0.00**
# indels per 100 kbp	8.60	2.57	1.86	**0.00**
# genes	262 + 7 part	262 + 8 part	249 + 4 part	**270 + 0 part**
Largest alignment	28559	86569	96535	**116866**
NA50	20762	86569	96535	**116866**
NGA50	20762	86569	96535	**116866**
NA75	19567	29474	96535	**116866**
NGA75	19567	29474	96535	**116866**
LA50	3	**1**	**1**	**1**
LGA50	3	**1**	**1**	**1**
LA75	4	2	**1**	**1**
LGA75	4	2	**1**	**1**

Table 4: All statistics are based on contigs of size $\geq$ 50 bp, unless otherwise noted (e.g., "# contigs ($\geq$ 0 bp)" and "Total length ($\geq$ 0 bp)" include all contigs).

3.3.2 Visualization with mummer and graph_generator.py

3.3.2.1 graph_generator.py visualization

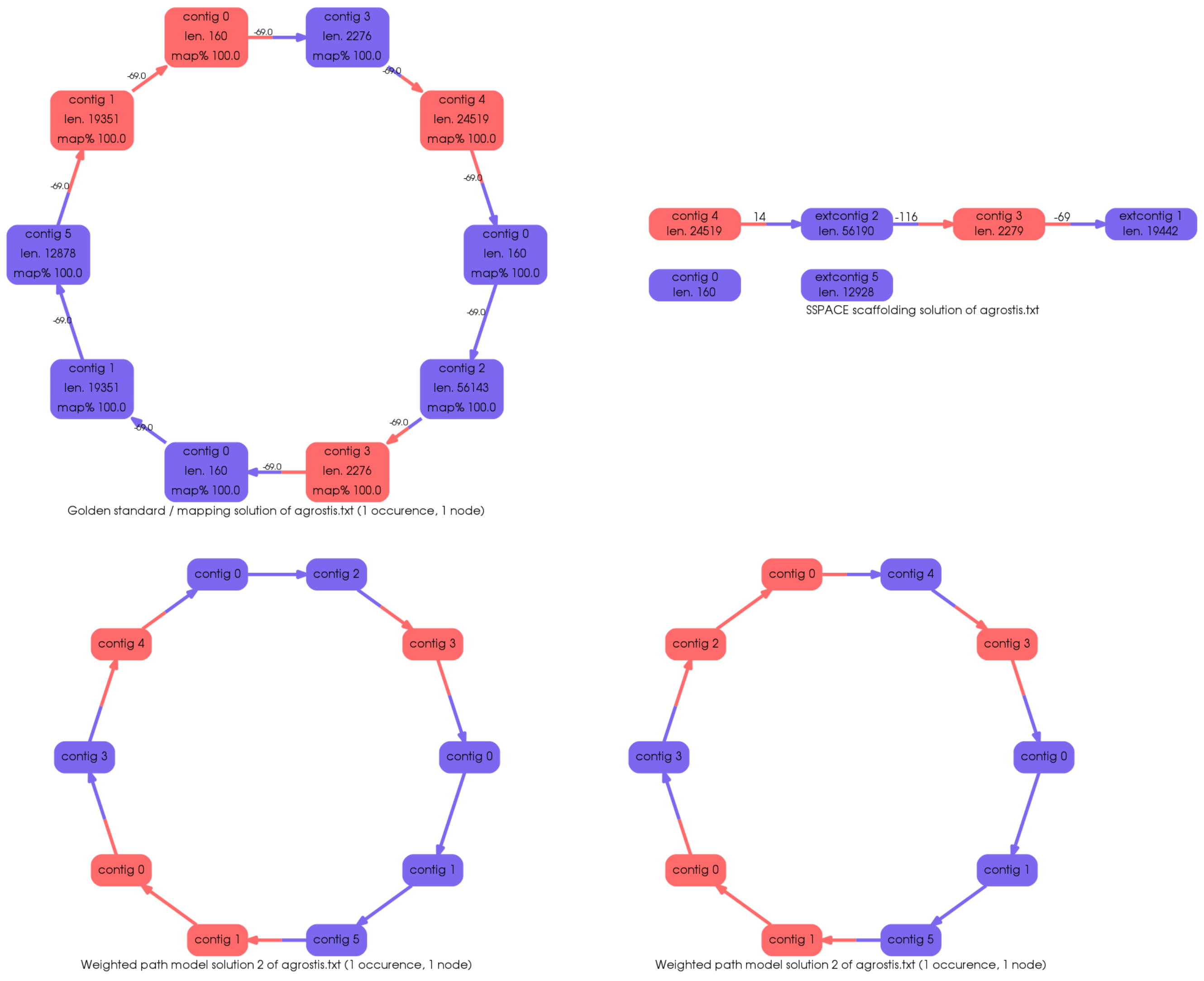

contig 0 len. 160 map% 100.0
-69.0
contig 3 len. 2276 map% 100.0
-69.0
contig 4 len. 24519 map% 100.0
-69.0
contig 0 len. 160 map% 100.0
-69.0
contig 2 len. 56143 map% 100.0
-69.0
contig 3 len. 2276 map% 100.0
-69.0
contig 0 len. 160 map% 100.0
-69.0
contig 1 len. 19351 map% 100.0
-69.0
contig 5 len. 12878 map% 100.0
-69.0
contig 1 len. 19351 map% 100.0
-69.0
Golden standard / mapping solution of agrostis.txt (1 occurence, 1 node)
contig 4 len. 24519
14
extcontig 2 len. 56190
-116
contig 3 len. 2279
-69
extcontig 1 len. 19442
contig 0 len. 160
extcontig 5 len. 12928
SSPACE scaffolding solution of agrostis.txt
contig 0
contig 2
contig 3
contig 0
contig 1
contig 5
contig 1
contig 0
contig 3
contig 4
Weighted path model solution 2 of agrostis.txt (1 occurence, 1 node)
contig 0
contig 4
contig 3
contig 0
contig 1
contig 5
contig 1
contig 0
contig 3
contig 2
Weighted path model solution 2 of agrostis.txt (1 occurence, 1 node)

Figure 9: Expected solution and scaffolding solutions of weighted-path model and SSPACE scaffolders

3.3.2.2 Dotplots with mummer

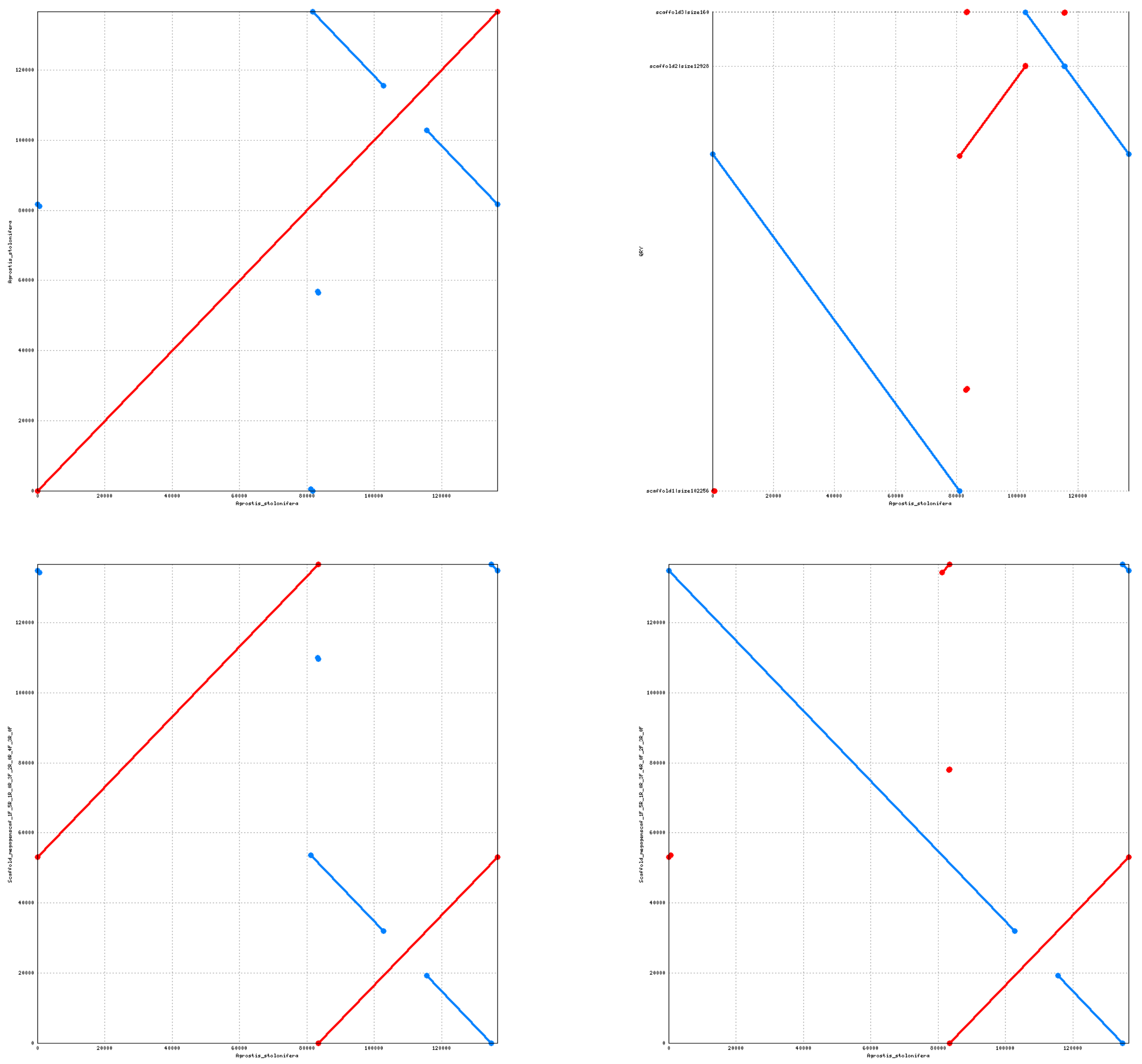


Figure 10: Control dotplot and dotplots of weighted path model and SSPACE scaffolding solution with the reference genome

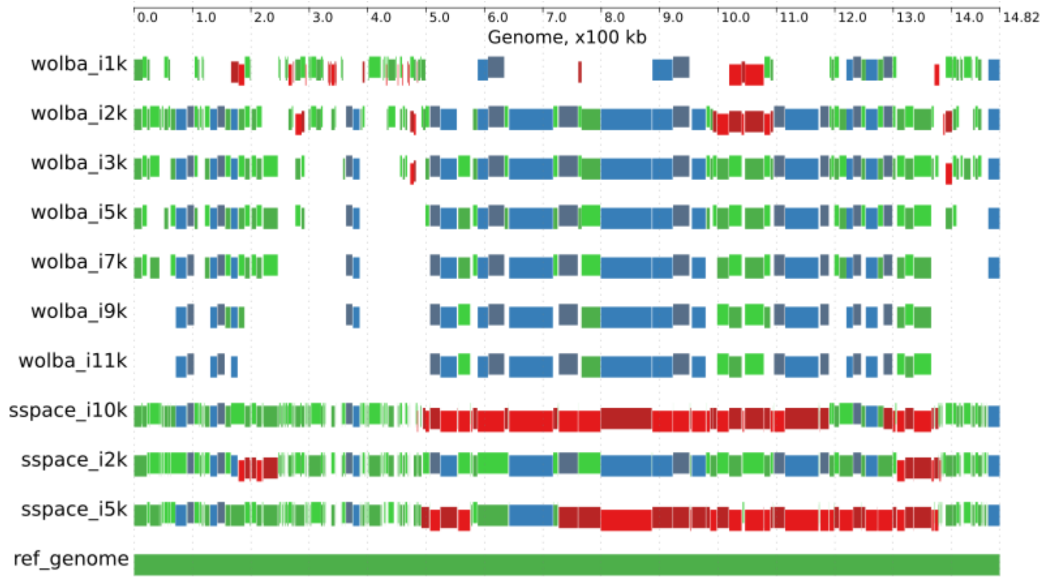
3.4 Partially solved instances

The bacterial genomes are partially assembled by the flow model, which processes big mate-pair connected unitigs first. The wolbachia example highlights the importance of correctly choosing the mate pairs' insert size. The results presented in table 5 show a drastic improvement of the total assembled length when changing the mate paired insert size from 1000bp to 2000bp. This length gradually decreases as the insert size increases (same with the genome size metric). The largest scaffold is also assembled with the 2000bp library (over 2000bp whereas all other libraries yield scaffolds < 500kb). One proposed explanation is the repeats' sequence size of *Wolbachia Endosymbiont*. As previously seen in Figure 7 Wolbachia endosymbiont genome dotplotted against itself section 2.1.2 on page 7 Features of the assembled genomes, the bacterial genome has repeats with sizes mainly between 500bp and 1kbp. These repeats are partially solved during the unitig building step. The issue comes from genomically close repeats bigger than 1.2kbp. There are 25 repeats of this type, among them 3 bigger than 5kbp. The location of these repeats coincide with regions of major scaffolding problems, where unitigs are not scaffolded or so badly scaffolded that they do not map on the reference genome. To see those location see figure 11 and table 6. The insert size must be big enough to overlap all repeats however increasing the insert size too much impacts on the scaffolders' ability to precisely join regions containing smaller overlaps. Using multiple libraries would be a solution however this strategy didn't result in encouraging scaffoldings with SSPACE (lib i2000 and i5000 used simultaneously).

Assembly	unitigs	wolba_i10k	wolba_i11k	wolba_i1k	wolba_i2k	wolba_i3k	wolba_i4k	wolba_i5k	wolba_i6k	wolba_i7k	wolba_i8k	wolba_i9k	ref_genome
# scaffolds ($\geq$ 0 bp)	444	9	7	37	28	19	13	13	13	11	9	9	**1**
# scaffolds ($\geq$ 1000 bp)	138	9	7	37	28	19	13	13	13	11	9	9	**1**
Total length ($\geq$ 0 bp)	1364357	871378	847704	615455	1214243	1197652	1121566	1101921	1084716	1045627	869142	879828	**1482355**
Total length ($\geq$ 1000 bp)	1290990	871378	847704	615455	1214243	1197652	1121566	1101921	1084716	1045627	869142	879828	**1482355**
# scaffolds	444	9	7	37	28	19	13	13	13	11	9	9	**1**
Largest scaffolds	87315	387284	387518	63122	222630	368628	368594	483412	211466	334472	249957	249957	**1482355**
Total length	1364357	871378	847704	615455	1214243	1197652	1121566	1101921	1084716	1045627	869142	879828	**1482355**
Reference length	1482355	1482355	1482355	1482355	1482355	1482355	1482355	1482355	1482355	1482355	1482355	1482355	1482355
GC (%)	34.01	33.92	33.94	34.08	33.89	33.95	33.96	33.96	33.91	33.93	33.95	33.94	34.19
Reference GC (%)	34.19	34.19	34.19	34.19	34.19	34.19	34.19	34.19	34.19	34.19	34.19	34.19	34.19
N50	17458	142935	143187	21300	68630	75831	142496	122998	116953	189499	189823	189972	**1482355**
NG50	14993	68085	68090	-	59359	69791	122971	77779	93415	122899	45001	57433	**1482355**
N75	7665	79427	79468	12604	44181	48864	75838	58047	77742	62360	75545	68114	**1482355**
NG75	5696	-	-	-	14316	22037	15579	-	-	-	-	-	**1482355**
L50	22	2	2	7	6	4	3	2	4	2	2	2	**1**
LG50	25	5	5	**-**	9	6	4	4	6	4	6	6	1
L75	52	4	4	17	12	8	5	5	7	5	4	5	**1**
LG75	65	**-**	**-**	**-**	19	14	13	**-**	**-**	**-**	**-**	**-**	1
# misassemblies	**0**	1	**0**	12	3	1	1	**0**	**0**	**0**	1	**0**	**0**
# misassembled scaffolds	**0**	1	**0**	7	3	1	1	**0**	**0**	**0**	1	**0**	**0**
Misassembled scaffolds length	**0**	142935	**0**	139698	143728	22037	35610	**0**	**0**	**0**	142540	**0**	**0**
# local misassemblies	**0**	23	23	45	66	60	52	47	42	38	30	26	**0**
# unaligned scaffolds	0 + 0 part	0 + 0 part	0 + 0 part	0 + 0 part	0 + 0 part	0 + 0 part	0 + 0 part	0 + 0 part	0 + 0 part	0 + 0 part	0 + 0 part	0 + 0 part	0 + 0 part
Unaligned length	0	0	0	0	0	0	0	0	0	0	0	0	0
Genome fraction (%)	90.341	53.726	51.551	40.034	78.587	76.826	71.970	70.313	68.955	66.047	54.424	54.992	**100.000**
Duplication ratio	1.019	1.094	1.109	1.040	1.042	1.052	1.051	1.057	1.061	1.068	1.077	1.079	**1.000**
# N's per 100 kbp	**0.00**	8603.27	9854.62	1808.26	3969.22	4880.30	4859.72	5402.66	5764.55	6364.03	7177.65	7348.94	**0.00**
# mismatches per 100 kbp	**0.00**	**0.00**	**0.00**	42.97	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**
# indels per 100 kbp	**0.00**	**0.00**	**0.00**	4.72	0.09	0.09	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**
Largest alignment	87315	356813	356813	62424	219709	359164	359164	459850	205359	319611	237693	237693	**1482355**
NA50	17458	89651	115294	16288	59479	71508	134513	111512	111512	116821	99293	119120	**1482355**
NGA50	14993	23733	29168	-	51212	64638	71508	56028	74077	56028	33486	44273	**1482355**
NA75	7665	65191	65191	10269	35490	42401	47594	49146	56028	55817	44360	65191	**1482355**
NGA75	5696	-	-	-	10269	10269	-	-	-	-	-	-	**1482355**
LA50	22	2	2	9	7	4	3	2	4	3	3	3	**1**
LGA50	25	8	7	**-**	9	6	5	5	7	5	8	7	1
LA75	52	5	5	20	13	9	6	6	8	6	6	5	**1**
LGA75	65	**-**	**-**	**-**	24	18	**-**	**-**	**-**	**-**	**-**	**-**	1

All statistics are based on contigs of size $\geq$ 50 bp, unless otherwise noted (all scaffolds are).

Table 5: Scaffolding solutions of the flow model step 1 with mate-pair libraries of different insert sizes compared to the reference genome and the initial unitig sample.



This plot shows alignment of contigs to the reference genome and positions of misassemblies in these scaffolds. Correctly aligned big (> 10kb) scaffolds are blue if the boundaries agree, and green if the boundaries don't agree. Scaffolds containing an important amount of misassemblies are red. Here, SSPACE solution with the 2kbp insert size library shows the less misassembled regions with the most reference coverage. However the solution is broken up in an enormous amount of scaffolds (336), most of them being just the unsucessfully attached small unitigs. SSPACE also fails to produce any valid scaffold for the genomic regions containing the large >5000 repeats (see their exact coordinates in table 6).

Figure 11: Alignment of scaffolding solutions on the reference genome of Wolbachia endosymbiont

[S1]	[E1]	[S2]	[E2]	[LEN 1]	[LEN 2]	[% IDY]
1	1482355	1	1482355	1482355	1482355	100.00
118235	119770	14639	13104	1536	1536	99.67
951662	953006	62728	61384	1345	1345	100.00
13104	14639	119770	118235	1536	1536	99.67
295130	296467	156339	155002	1338	1338	100.00
1044418	1046269	178963	177112	1852	1852	100.00
295131	296471	201307	199967	1341	1341	99.93
346054	354494	259128	250682	8441	8447	99.56
1040287	1042282	286820	284825	1996	1996	100.00
1370666	1372235	293326	291757	1570	1570	99.87
886392	887730	296467	295129	1339	1339	100.00
1459215	1461421	296650	294476	2207	2175	98.50
353564	355681	325239	323122	2118	2118	97.54
346054	351130	332648	327572	5077	5077	99.94
250682	259128	354494	346054	8447	8441	99.56
295130	296467	426552	425215	1338	1338	100.00
353564	355681	449855	447738	2118	2118	97.54
951661	953011	506502	505152	1351	1351	99.93
1305338	1306686	553913	552565	1349	1349	99.93
295129	296467	887730	886392	1339	1339	100.00
505152	506502	953011	951661	1351	1351	99.93
284825	286820	1042282	1040287	1996	1996	100.00
177112	178963	1046269	1044418	1852	1852	100.00
1334495	1335840	1306687	1305342	1346	1346	100.00
1305342	1306686	1320819	1319475	1345	1345	100.00
1305342	1306687	1335840	1334495	1346	1346	100.00
291757	293326	1372235	1370666	1570	1570	99.87
294476	296650	1461421	1459215	2175	2207	98.50

Table 6: Coordinates and length of repeats in the *Wolbachia endosymbiont* genome

4 Discussion

5 Conclusion

6 References

References

[1] Hunt, M., Newbold, C., Berriman, M. & Otto, T. D. A comprehensive evaluation of assembly scaffolding tools **15**, R42.

[2] Earl, D. *et al.* Assemblathon 1: A competitive assessment of de novo short read assembly methods **21**, 2224–2241. URL http://www.ncbi.nlm.nih.gov/pmc/articles/PMC3227110/.

[3] Bradnam, K. R. *et al.* Assemblathon 2: evaluating de novo methods of genome assembly in three vertebrate species **2**, 10. URL http://www.ncbi.nlm.nih.gov/pmc/articles/PMC3844414/.

[4] Pop, M., Kosack, D. S. & Salzberg, S. L. Hierarchical scaffolding with bambus **14**, 149–159. URL http://genome.cshlp.org/content/14/1/149.

[5] Treangen, T. J. *et al.* MetAMOS: a modular and open source metagenomic assembly and analysis pipeline **14**, R2. URL http://genomebiology.com/2013/14/1/R2/abstract.

[6] Zerbino, D. R. & Birney, E. Velvet: Algorithms for de novo short read assembly using de bruijn graphs **18**, 821–829. URL http://genome.cshlp.org/content/18/5/821.

[7] Hunt, M., Newbold, C., Berriman, M. & Otto, T. D. A comprehensive evaluation of assembly scaffolding tools **15**, R42. URL http://genomebiology.com/2014/15/3/R42/abstract.

[8] Huson, D. H., Reinert, K. & Myers, E. W. The greedy path-merging algorithm for contig scaffolding **49**, 603–615. URL http://doi.acm.org/10.1145/585265.585267.

[9] Li, H. & Durbin, R. Fast and accurate short read alignment with burrows-wheeler transform **25**, 1754–1760.

[10] Langmead, B. & Salzberg, S. L. Fast gapped-read alignment with bowtie 2 **9**, 357–359. URL http://www.nature.com/nmeth/journal/v9/n4/full/nmeth.1923.html.

[11] Langmead, B., Trapnell, C., Pop, M. & Salzberg, S. L. Ultrafast and memory-efficient alignment of short DNA sequences to the human genome **10**, R25. URL http://genomebiology.com/2009/10/3/R25/abstract.

[12] Lander, E. S. *et al.* Initial sequencing and analysis of the human genome **409**, 860–921.

[13] Venter, J. C. *et al.* The sequence of the human genome **291**, 1304–1351.

7 Annexes