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

I would like to thank the GENSCALE team for making me a part of this project. I thank Pr. Rumen Andonov and Dr. Dominique Lavenier for their guidance and supervision as well as Ivaylo Petrov for taking time to discuss the technical aspects of this project with me. Thanks to Sebastian Francois for his enthusiasm in developing a new approach. I would also like to thank my fellow undergraduates research assistants for their pleasant company and daily good and the PhD students who took time to welcome and advice us.

ABBREVIATIONS

Acorus calamus chloroplastic genome	acorus
Agrostis stolonifera chloroplastic genome	agrostis
Atropa belladonna chloroplastic genome	atropa
Cucumis sativus chloroplastic genome	cucumis
*D*istance based model	dist
Eucalyptus globulus chloroplastic genome	eucalyptus
Euglena gracilis chloroplastic genome	euglena
Genscale Scaffolding Tools	GST
Lecomtella madagascariensis chloroplastic genome	lecomtella
*N*ext Generation Technologies	NGS
Oenothera elata chloroplastic genome	oenothera
Oryza sativa Japonica chloroplastic genome	rice
Pinus koraiensis chloroplastic genome	pinus
*W*eighted path model	wpm or whpm
Wolbachia endosymbiont bacterial genome	wolbachia

GITHUB REPOSITORY

https://github.com/aliecs/Stage2015_gr

Contents

1 Introduction **1**
- 1.1 Context . . . 1
- 1.2 Background . . . 1
- 1.3 Assembly terminology . . . 2
  - 1.3.1 Reads, pairing and overlaps . . . 2
  - 1.3.2 Unitigs and contigs . . . 2
  - 1.3.3 Obtaining scaffolds . . . 3
- 1.4 A history of scaffolding strategies and modeling . . . 5
  - 1.4.1 Examples of scaffolding tools . . . 5
  - 1.4.2 Limits of the scaffolding tools . . . 6
- 1.5 Goal of the internship project . . . 7

2 Material and methods **8**
- 2.1 Genscale scaffolding methodology . . . 8
  - 2.1.1 Format of the input data for genscale scaffolders . . . 8
  - 2.1.2 Features of the assembled genomes . . . 9
  - 2.1.3 Genscale scaffolding problem modeling . . . 10
  - 2.1.4 Differences between Genscale scaffolding tools . . . 11
- 2.2 Input data features and complexity inspection . . . 12
- 2.3 Benchmarking . . . 12
  - 2.3.1 Comparison with the expected solution . . . 12
  - 2.3.2 Comparison with published tools: benchmarking strategy . . . 13

3 Results **15**
- 3.1 Comparison between the data sets . . . 15
- 3.2 Comparison of GST solutions with the expected solution . . . 16
- 3.3 Comparison of the best GST solutions and SSPACE to the reference genome . . . 17
  - 3.3.1 Quality assessment of genome assemblies of *Agrostis stolonifera* chloroplastic genome 17
  - 3.3.2 Quality assessment of genome assemblies of *Pinus koraiensis* chloroplastic genome 20
- 3.4 Partially solved instance - flow model . . . 21

4 Conclusion **23**

1 Introduction

1.1 Context

The nucleic biological molecules of genomes and transcriptomes can be sequenced by many different techniques. Sequencing is the process which determines the nucleic order within DNA and RNA. The first methods involved the extension of location-specific primers, meaning portions of the genome or transcriptome had to be known in advance. An improved and widely known version of this method was developed by Frederick Sanger in 1970. Sanger's sequencing[1] uses radioactive dideoxynucleotide stopping the DNA replication - for each elongated DNA molecule the last nucleic acid is thus determined. These techniques are costly and slow because they involve handling a DNA molecule for each nucleic acid position and genomes can have as many as 150 billion base pairs. With the increasing demand for genomic and transcriptomic sequencing, New Generation Sequencing (NGS) methods were introduced. NGS is faster and sequences the whole genome at once. The Illumina NGS technology does so by fragmenting the genomes into DNA fragments and determining the nucleic order of their extremities thanks to fluorescently-labeled nucleotides. In order to recover the whole genomic sequence the small sequenced parts, called reads, need to be pieced together. The GENSCALE team at the INSTITUTE FOR RESEARCH IN IT AND RANDOM SYSTEMS uses data obtained though the Illumina NGS technique to develop new read assembling strategies.

1.2 Background

De novo assembly is the process which pieces together overlapping reads produced by NGS 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[2]. 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 detailed process of assembly is described in section 1.3 page 2. The two main steps are building large uninterrupted sequences from overlapping reads and scaffolding, the ordering and relative orientation of these large sequences (called contigs or unitigs) . The 2011 and 2013 Assemblathon projects[3][4] 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.

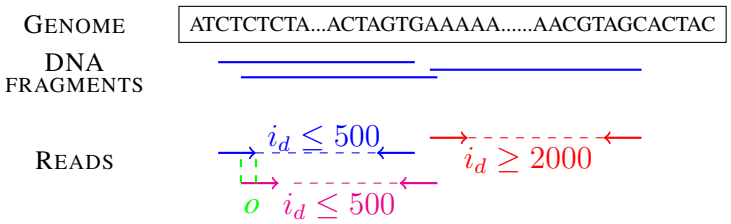
The first stand-alone scaffolder named Bambus[5], originally part of the MetAMOS[6] assembly and analysis pipeline, was published in 2004. Previously the scaffolding step was missing or presented as an option within conting builders, for instance the Velvet[7] assembler *'scaffolding yes or no'* option. In the 2014 comprehensive evaluation of scaffolding tools[8], 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 lead to 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.*[9] which proposed a greedy path-merging strategy, described in section 1.4 page 5 along with other solving methodologies and the distinctive features of the GENSCALE Scaffolding Tools[10] (GST). Concepts surrounding the Illumina NGS technique are explained in section 1.3 page 2.

1.3 Assembly terminology

In this report *assembly* will refer to the whole multi-step process starting from once filtered out-of-the-sequencer data and resulting in a highly uninterrupted sequence of a genome or chromosome, in the best case scenarios. 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 and is explained in section 1.3.2 page 2. Another key point is the construction of joins between contigs/unitigs - also referred to as links, edges or bonds according to the way they are modeled.

1.3.1 Reads, pairing and overlaps

A read is a short ($< 500pb$) copy of a DNA fragment of known length and nucleic acid sequence. It is produced differently depending on the sequencing technology. Paired reads are copies of the two extremities of the same DNA molecule. The DNA molecule size 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, the read size 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.



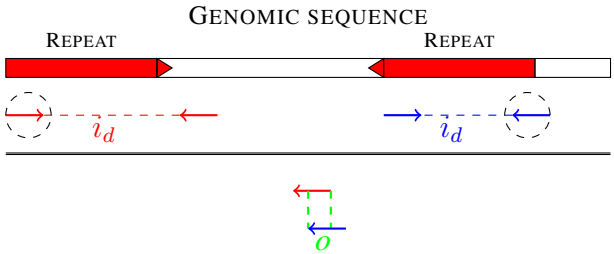
Each end of a DNA molecule is cloned to produce paired reads. Here is represented a mate-paired pair (red) with a big insert size *(i_d)* and two paired-end pairs (blue and magenta) which slightly overlap *(o)*.

Figure 1 – Alignment of paired reads on fragmented DNA molecules obtained from the genome

Figure 1 shows three pairs of reads. Within the pairs, reads are facing each other: this configuration is called *Forward-Reverse* read orientation. To be sequenced the genome represented in figure 1 is first fragmented into numerous DNA molecules by sonication or nebulization then amplified by Polymerase Chain Reaction. Each end of the molecule is then fixed on sequencing wells. 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.

1.3.2 Unitigs and contigs

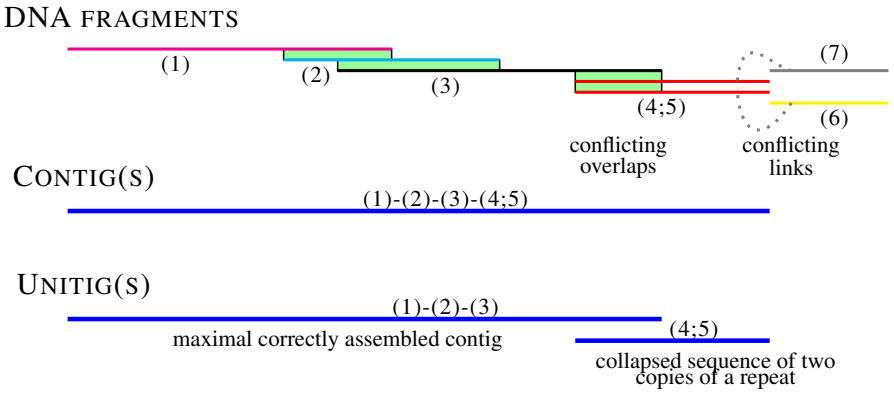
Unitigs are an uniquely assemblable subset of overlapping fragments. At the end of an unitig data shows multiple dubious overlaps as seen in section 1.3.1 page 2 creating joins with multiple other unitigs. Contigs are larger than unitigs, extended through repeat boundaries but are still ungapped sequences. Contigs are interesting to construct because there is a higher chance to detect genes. Taking the example shown in figure 3, 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



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

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 section 2.1 page 8.



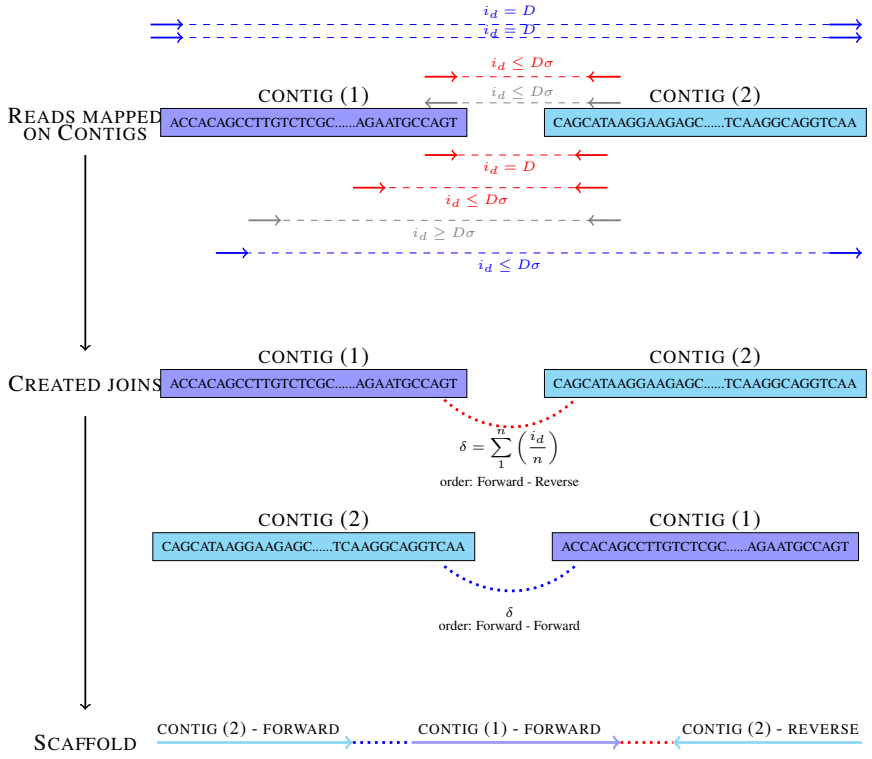
Unitigs end at multiple overlaps indicating a possible repeat. Contigs can be extended through conflicting overlaps. Here, the red DNA fragments are two copies of a repeat. When no more overlaps exist, contigs can be linked (gray dotted line) thanks to information provided by read pairs. This is the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

Figure 3 – The difference between unitigs and contigs

1.3.3 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[11] and bowtie[12;13]. 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 section 2.1 page 8.
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

* * *

The list of contigs or unitigs and the list of joins between them is the minimal data to provide to a scaffolder tool. There are different ways to model the problem computationally. Additional information such as join coverage, contig coverage and join length can be introduced to help solve the scaffolding problem.

1.4 A history of scaffolding strategies and modeling

The scaffolding problem was first introduced in 2002 by Hudson *et al.*[9] following the challenges which arose during the human genome clone-by-clone sequencing by *Lander et al.*[14] and the human whole genome shotgun assembly project by Venter *et al.*[15], 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 most common way in which the problem is modeled is as a graph where nodes 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 links in the graph). Strategies to solve it and find the best path are numerous. It is important to note that Hudson *et al.* regard **the use of mate-pairs as crucial**. There are indeed several scaffolding tools that accept paired-reads libraries with small insert sizes as valid input data - in these cases it is not expected to obtain a quality scaffolding solution. In this section the Hudson *et al.*[9] modeling and scaffolding strategy are detailed, and several scaffolding tools roughly described.

1.4.1 Examples of scaffolding tools

Hudson *et al.* proposed the first heuristic greedy path-merging algorithm for solving the scaffolding problem. Real data can and will be noisy. Inconsistent data is at the basis of the constraints defined for the problem - joins between contigs can be mutually exclusive, erroneous or absent. These features cause the scaffolding problem to be NP-hard. A heuristic approach is a pertinent approach so that scaffolding tools are prepared to quickly take decisions when faced with inconsistent data.

Greedy path merging approach Hudson *et al.*'s scaffolding graph models each contig as two vertexes `Contig - start` and `Contig - end` connected by a `Contig - EDGE` (actually a link, as it is directed `start-> end`). This is a strategy to model the two possible orientations of a contig, read from start to end (*Forward*), or from end to start (*Reverse*). The starting and ending point of a contig are arbitrary as contig building does not provide direction. `Mate - EDGES` are joins between the start (or end) of a contig and the start (or end) of another contig. The weight on these edges is the edge coverage (how many times the created join is supported by data). These edges are undirected. The contig coverage is not taken into account in this model. In fact, suspected repeated regions are removed during the graph solving process. The length between two mate pairs (insert size) is not represented in the final graph either. It is only used to allow edge bundling when several mate-pairs support the same join between contigs. The edge bundling is performed by first greedily choosing a median length of a `Mate - EDGE`, then gathering all `Mate - EDGES` that are within $3(standarddeviation)(medianlength)$ to bundle. The weight of `Mate - EDGES` is 1 if it is a single edge and is the number of mate-pairs participating if the `Mate - EDGES` is a bundle. The information of length between contigs is lost. Transitive reduction decreases the number of edges and leads to the final form of the graph that will be solved by the greedy path merging algorithm. The algorithm constructs a path though the scaffolding graph. The `Mate - EDGES` are processed according to their weight (decreasing order) and the found paths are progressively merged until there are no more `Mate - EDGE` left. As `Mate - EDGE` are processed one by one in a determined order, this method is may not be optimal or perfect.

Algorithm 1 Greedy path-merging algorithm as presented in Hudson *et al.* 2002 paper
$H(Path)$ is the sum of mate edge weights which supports $Path$ and $U(Path)$ is the sum of mate edge weights which contradict $Path$

begin
Select all contig edges
for each mate-edge *e* in descending order of weight:
    if e is not selected:
        Let *v*, *w* denote the two nodes connected by *e*
        Let $P1$ be the selected path incident to *v*
        Let $P2$ be the selected path incident to *w*
        if $P1 \neq P2$ and we can merge $P1$ and $P2$ (guided by *e*) to obtain P:
            if $H(P) - (H(P1) + H(P2)) \geq (U(P) + U(P2))$:
                Replace $P1$ and $P2$ by P
end

SSPACE [16] The SSPACE scaffolding algorithm stores orientation and position of paired contigs by a join during its mapping step. This is list is the starting point of the scaffolding step: SSPACE is one of the few successful scaffolders that does not model the problem as a graph. The scaffolding itself starts with the largest contig and iteratively combines other contigs to it if a minimum number of read pairs exists between them (the default being 5 paired reads joining the two contigs). Alternative connections are dealt with thanks to join coverage as seen in annex **??**. Contig coverage is not taken into account. The scaffolding process stops when the last combined contig has no joins.

Bambus Bambus uses a greedy algorithm to join contigs with the most joins first and ignores subsequent edges that conflict with the formerly used join. This methodology is heavily heuristic as previously accepted joins can be sub-optimal and will cause bias for all the following contig joins.

SOPRA [17] SOPRA was the first tool to try solving the problem exactly. It models the problem as an undirected graph. It removes inconsistent edges and nodes that give rise to spurious links (a choice is presented, ambiguity ensues). This is repeated until no more edges or nodes have to be removes - everything is consistent. Any contig containing repeated regions will thus be discarded.

SCARPA [18] SGA uses a very conservative approach by disallowing any conflicts in the modeled graph, avoiding heuristics at the expense of missing valid joins. Any contig containing repeated regions will thus be discarded.

GST - Genscale Saffolding Tools GST model the problem as a directed graph and try to solve the problem by finding the optimal path. The GST include several different strategies. Their common feature is the fact that they heavily rely on contig coverage and create a node for each contig occurance and orientation. See further explanations about the modeled graph in section 2.1.3 page 10. Conflicting join data will co-exist in the graph and given a choice, the optimal path may be different according to the objective function. Differences between GST are described in section 2.1.4 page 11.

1.4.2 Limits of the scaffolding tools

Erroneous data (fake links due to poorly assembled contigs, low quality libraries), missing data (low quality libraries, unfit insert size, low genome coverage) and inherent genome characteristics (repeated regions, heterozygosity) stand in the way of a perfect and easy scaffolding process. Scaffolders employ strategies to solve the ambiguities heuristically (trusting mate-pair join occurrences, ignoring conflicting links with the best path constructed from an arbitrary chosen starting contig, starting from the biggest contig). Those tools which try to solve the problem exactly are too strict regarding multiple possible paths.

Last but not least, the contig coverage is not their priority. Repeats are not handled, or at best detected and removed. Considering conflicting joins arise mainly from repeated regions, at least in perfect simulated data without sequencing errors, this kind of information ought to be given some importance. Also, scaffolding tools provide a single solution when multiple versions of the same genome can cohabit within the same organism or cell.

1.5 Goal of the internship project

The goal of this report is to present the performance of the GST with reliable simulated data. This is done to study its decision making in case of conflicting joins that will absolutely arise in real life and are allowed to co-exist in the GST problem modeling. The first step is to evaluate the complexity of the scaffolding modeled graph and compare the solutions obtained by GST with the provided genome reference. The methodology is described in section 2.1 page 8. 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 section 2.3 page 12. Results are presented in section 3 page 15. Global metrics are presented for all instances and a detailed study is made for the chloroplastic genomes of *Agrostis stolonifera* and *Pinus koraiensis*, and the bacterial genome of *Wolbachia endosymbiont*. The section 4 page 23 will provide an insight of the context of the GENSCALE scaffolding project in the light of the genome sequencing technique progress. Finally, the conclusion is on page 23.

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 previously obtained bundles of read pairs joining two unitigs. An example of file read by solvers is showed in figure 5 along with its graphical representation generated by the homemade `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. The script has an option to indicate the kind of graph to generate.

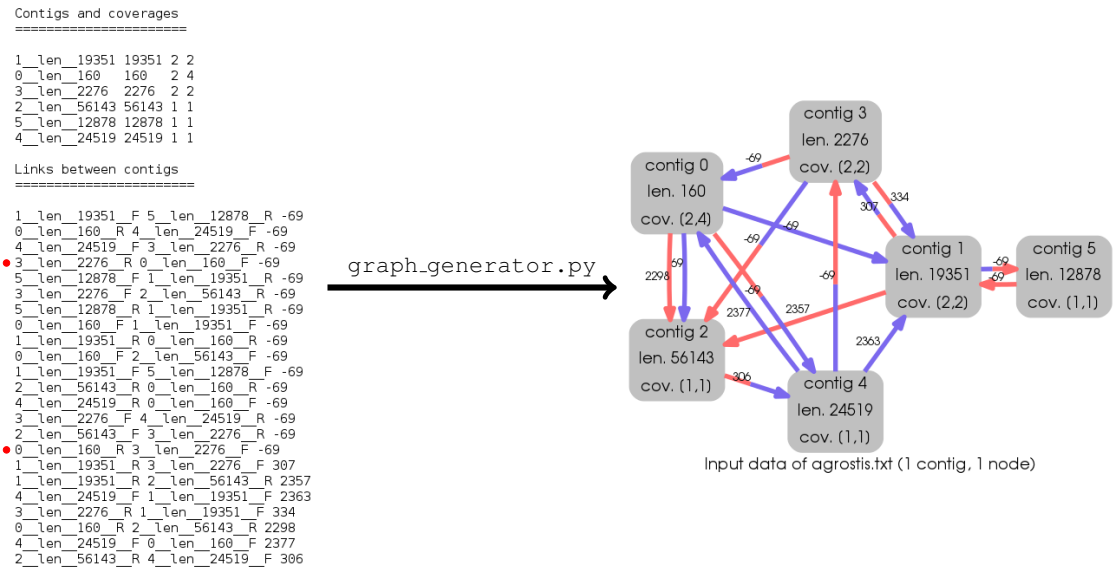
`graph_generator.py -g [whpm,dist,inpt,expt] -f file.txt`

where whpm is weighted path model graph, dist is distance based model graph, inpt is input graph and expt is expected solution graph if it is provided in the file.

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 represent 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)
```



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 data used in this report is artificial. It is simulated from *.fasta* files of reference genomes using two different scripts to generate a paired-end read library and a mated-pair read library. The paired-end read library

is used to built unitigs with MINIA [19;20], a contig building tool developed at GENSCALE which supports an unitig building option. The size of the k-mer (minimal number of overlapping bases between two reads for the overlap to be regarded as valid) is chosen so that MINIA produces the fewest number of unitigs. At this stage published scaffolders can already be used - they require the unitig file and the mated-pair library and perform the read mapping themselves to detect joins. The GST require the mapping process to be separately performed and its input file to roughly resemble the one showed in 5. The GST mapping process has three goals:

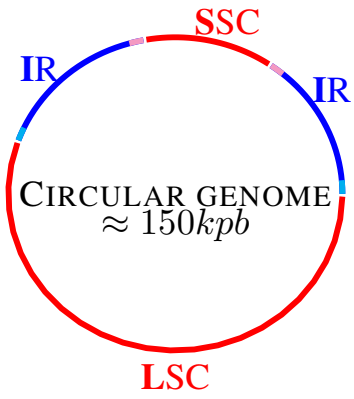
- mapping mate pairs on large unitigs $\mapsto$ insert size
- mapping paired reads on unitigs $\mapsto$ unitig coverage
- mapping mate pairs on different unitigs $\mapsto$ links between unitigs

The bigger the unitig, the more robust the mapping information (homogeneously mapped reads) - hence no exact value 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 fast and detailed assessment and benchmarking of found solutions.

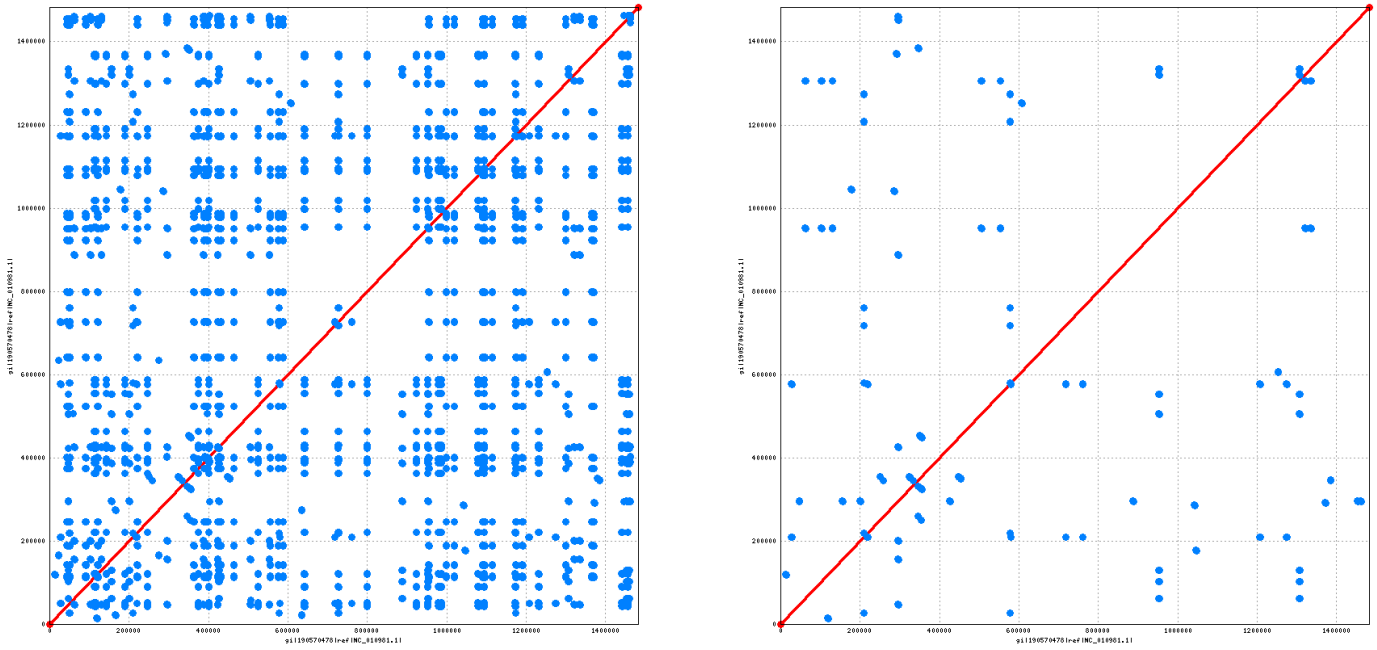
Chloroplasts Chloroplasts are small organelles in plant photosynthetic tissues which possess their own DNA. The chloroplast genomes are small ($\approx 150kpb$), circular and have a large inverted repeated sequence of around $25kpb$. Some instances used in this study lack this repetition. This is the case in *Pinus koraiensis* and *Euglena gracilis*. However these two genomes possess significantly more small ($<$ 20bp) repeated sequences. Figure 6 shows the classic chloroplastic structure, with colored IR extremities to indicate their relative position, *"facing"* each other in the genome.



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

Figure 6 – Chloroplast genome structure

Bacteria Bacterial genomes are bigger ($>$ 1Mpb) and contain many small repeated sequences between 500pb and 1000pb. These genomes have regions which are very hard to assemble because short tandem repeated sequences won't permit any unitig to be built. Repeats as small as 100bp are detected with `nucmer`. 150bp reads covering these regions won't be assembled into any unitig and will not be considered as unitigs on their own. Regions of the reference genome will thus be completely absent when reaching the scaffolding step.

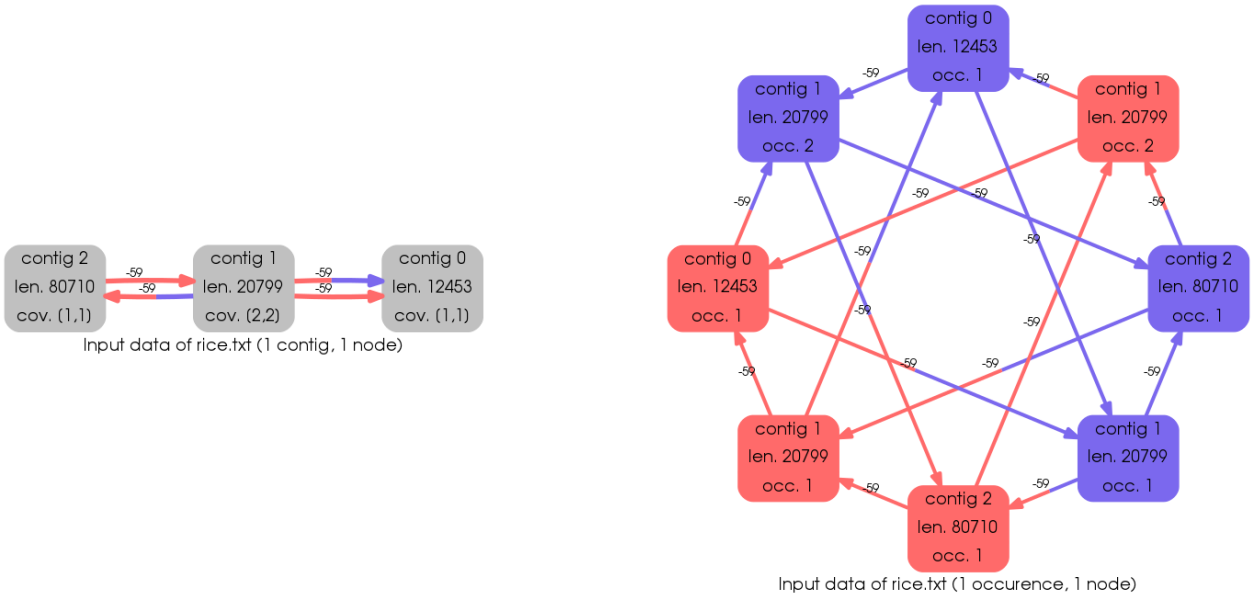


The dot-plots were generated by the nucmer script of the MUMmer[21;22]? sequence fast-alignment tool. (x) and (y) axis are both the *Wolbachia endosymbiont* reference genome downloaded on NCBI (*NC_010981.1*). The blue dots and lines show repetitions in the genome as the DNA sequence matches in several places. The closer the dot is to the red matching axis, the closer the repeats are. We can say, for instance, that at the 1.3Mbp there are several tandem repeats. On the left alignement, repeats longer than 500bp are represented (`nucmer -l 500` option). On the right repeats longer than 1200bp are represented.

Figure 7 – *Wolbachia endosymbiont* genome dotplotted against itself

2.1.3 Genscale scaffolding problem modeling

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 or orientation. However, this visualization is only useful for a first human assessment of the data. Genscale scaffolding tools model each unitig as many times as it occurs. The `unitig 3` with coverage 2 is transformed into `unitig 3 occurance 1` and `unitig 3 occurance 2`, two distinct nodes in the graph. The total number of contig occurrences is then duplicated to model the *Forward* and *Reverse* orientation. `unitig 3 occurance 1` is transformed into `unitig 3 occurance 1 Forward` and `contig 3 occurance 1 Reverse`. The number of links increases consequently. When transforming joins into model graph links, no duplicate links are allowed (merged) and for each link its reverse equivalent is created. Link coverage is not taken into account. Weight link is the estimated distance between two nodes, which is the space left between the two joined tips of two unitig. A distance is negative when two unitig overlap and positive when there is a gap. Gaps are obtained thanks to mated-pair reads information: in this case the distance is the estimated length of the pair's insert size minus the remained sequence length of the unitig following the reads' mapping. A simple example of the difference between the raw input graph and the processed model input graph that the GST solve is presented in figure 8.



Sum of unitig occurrences is 4 so the final number of nodes in the model graph is $4 * 2 = 8$. There are no duplicate links in this example. Flipping the model graph (change the link orientation and node color) produces all the reverse complement links.

Figure 8 – Input graphs of rice, as observed in the *.txt* file (left) and as modeled by the Genscale scaffolding tools (right)

2.1.4 Differences between Genscale scaffolding tools

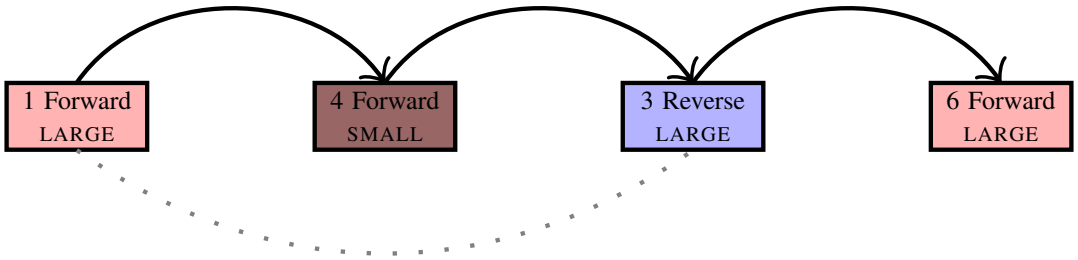
Although all GST try to solve the problem exactly, different models were developed having different objective functions and input information.

Weighted path model Weighted math model (wpm) does not take link weight into account and only aims at solving the order and relative orientation of unitig. The distance between unitig is not known. This model produces several solutions, among them several can be valid or sub-optimal ones. The objective function of this model maximizes the number of links (bundled read pairs) whose orientation corroborates the suggested orientation of unitigs they are linking. All nodes must be visited once because this model doesn't support intervals as unitig coverage. Visually, it means that each node in a solution has $inDegree = outDegree = 1$ and $color(inArrow) = color(outArrow) = color(Node)$.

Distance based model Distance based model (dist) gives weight to links which is the sequence size between two extremities of two unitigs: negative weight in case of an overlapping and positive weight in case of a mated-pair estimated insert size. It produces a single solution. On an axis from 0 to ∞ it tries to find the best position for a unitigs using linear programming so that there are as few as possible erroneous read pairs. The less conflicting data used, the better. The solution provided can contain two consecutive unitigs for which no join exists in the input data if positioning them so helps reducting the number of errors.

Flow model Flow model is currently being developed. The strategy is divided into two steps and relies on the definition of *large unitigs*. This definition evolved in the GENSCALE team: currently large unitigs are unitigs with a sequence length superior to the biggest mated-pair distance in the input data. The first step of the flow model strategy is to link these large unitigs and construct a genome frame containing gaps. The second step, yet to be implemented, aims at filling in the gaps with the remaining unitigs. This strategy

emerged when the wpm was able to construct a genome frame for large datasets (bacterial genomes) once small unitigs were filtered out. Since then the definition of large contigs evolved and linkage challenges were solved for this first step (see figure 9). Additionally, the flow model supports intervals for contigs coverage and confidence intervals for mated-pair distances.



If the only way to link unitig 1 to unitig 3 is though unitig 4, and unitig 4 is considered a small one and filtered out for the first step, then the linkage information between unitig 1 and unitig 3 is lost, yet they have to be consecutive in the genome framework. This unitig organization is found in pinus.

Figure 9 – Example of a linkage challenge when using solely large unitigs for a genome framework construction

2.2 Input data features and complexity inspection

Transforming the input data file into the graph solved by the GST is carried out by the Python interface to the Graphviz[23;24] graph layout and visualization package, pygraphviz. Graphviz provides quick methods to access the features of the graph. The Python implementation uses the networkx[25] package. Features and complexity of the input graph was evaluated with the underlying wish to understand why certain instances were wrongly scaffolded. Automated transformation and inspection of the input data was implemented in the *graph_inspector.py* script. Additionally this script can also provide differences between the expected solution and the input data, detecting missing linkage information in the input data.

2.3 Benchmarking

Comparing the performance of the GST can be done on several levels. As the data is artificially generated from an already known genome, the best solution is already known. The GST solutions can be compared between themselves, with the expected solution and with solutions provided by other published tools. The solution formats are not the same and automated methods of comparing were set up to extract and compare scaffoldings.

2.3.1 Comparison with the expected solution

The expected solution provided by the mapping of unitigs of the reference genome is the one GST aim to obtain in the form of an uninterrupted and accurately gapped scaffold. However in some cases the solution is not unique. In real life several versions of the genome can co-exist, so the GST should aim at providing all valid solutions. In our artificial datasets, only one version of the genome is used, however in the case of chloroplastic genomes the multiple valid solution configuration arises. The Small Single Copy and the Long Single Copy have both extremities next to the same extremities of the Inverted Repeat (see figure 6). In absence of strong mated-pair information overthrowing the overlapping information of these extremities, the SSC and the LSC can be oriented in a *Forward* or a *Reverse* configuration, creating a total of 4 valid

combination. Of course, the genomic structure can be much more complex than that presented in figure 6, providing linkage information that would corroborate an orientation over another. The fact is, a valid solution is not an exact copy of the expected solution. Because it is not known in advance which contigs form the IR of the chloroplastic genome and because in *de novo* assembly no assumption of the genomic structure shall be made at all, these alternative solutions are very hard to detect automatically.
The detection of exact copies of the expected solution has been implemented in the *graph_comparator.py* script. It is a simple list comparison after the retrieval of GST solution in the form of an ordered list of oriented unitigs. Distances between unitigs have been ignored for this comparison. Sub-optimal solutions are also detected. This detection effort has been made especially for the distance based model which provides a single solution which sometimes possesses only one or two errors of orientation. However misplacement of a large set of contigs can only be detected visually with the graph provided by the *graph_generator.py* script. This kind of error did not occur many times in the instances tested as misplacement of one or several contigs often leads to more errors.

2.3.2 Comparison with published tools: benchmarking strategy

A summary of the benchmarking strategy used is presented in figure 10. Based on the comprehensive evaluation of assembly scaffolding tools[2], the SSPACE scaffolding tool was chosen to be compared to GST. SSPACE is not a graph based scaffolder and its strategy of handling repeats is not explained. For all the other considered scaffolders, it is explicitly written that repeated regions are detected and discarded in order to avoid ambiguous joins. Before proceeding to the benchmarking, solution files from SSPACE and GST were converted. The two converters are :

- the *sspace_2_order.py* script which takes the SSPACE *.fasta* scaffolding solution and the SSPACE evidence file tracking the scaffold unitig composition, and outputs an ordered list of orientated (possibly extended) unitigs for each scaffold
- the *gst_order_2_scaf.py* script which takes the GST solution file and the unitig *.fasta* file and constructs the whole uninterrupted GST scaffold, appending N bases when the distance between unitigs is positive.

The *.fasta* files containing the nucleic sequence of scaffolding solution are compared with the Quality Assessment Tool for Genome Assemblies (Quast[26]). The reference genomes and the gene file downloaded from NCBI are given to Quast for additional metrics.

```
> quast.py gst_solution.fasta sspace_solution.fasta unitigs.fasta
-R ref_genome -G gene_file.txt -o quast_outpur_dir
```

Quast provides 31 metrics with this command, some more pertinent than others. The ones and that will be presented in section 3 page 15 are:

- #scaffolds and #scaffolds $\geq 1000bp$: especially interesting to compare it to the #unitigs
- total length of scaffolds: detect if any unitig was duplicated (GST) or elongated (SSPACE)
- largest built scaffold - usually the GST leads to one single large scaffold
- N50: length for which the collection of all contigs of that length or longer covers at least 50% of assembly length. Extensively used metric, but considering the GST solution aims at large unique solution, this metric is a nice indication for SSPACE. It is also useful for the GENSCLAE flow model, which is not finished yet and produces several discontinuous scaffolds.
- #misassemblies, #mismatches and #indels, which will occur at the unitig extremities when overlapping or elongations are wrongly executed.
- genome fraction assembled
- Ns per 100kbp: Ns are added in SSPACE scaffolds, flow and distance based model
- #genes found
- largest accurate alignment with the reference genome

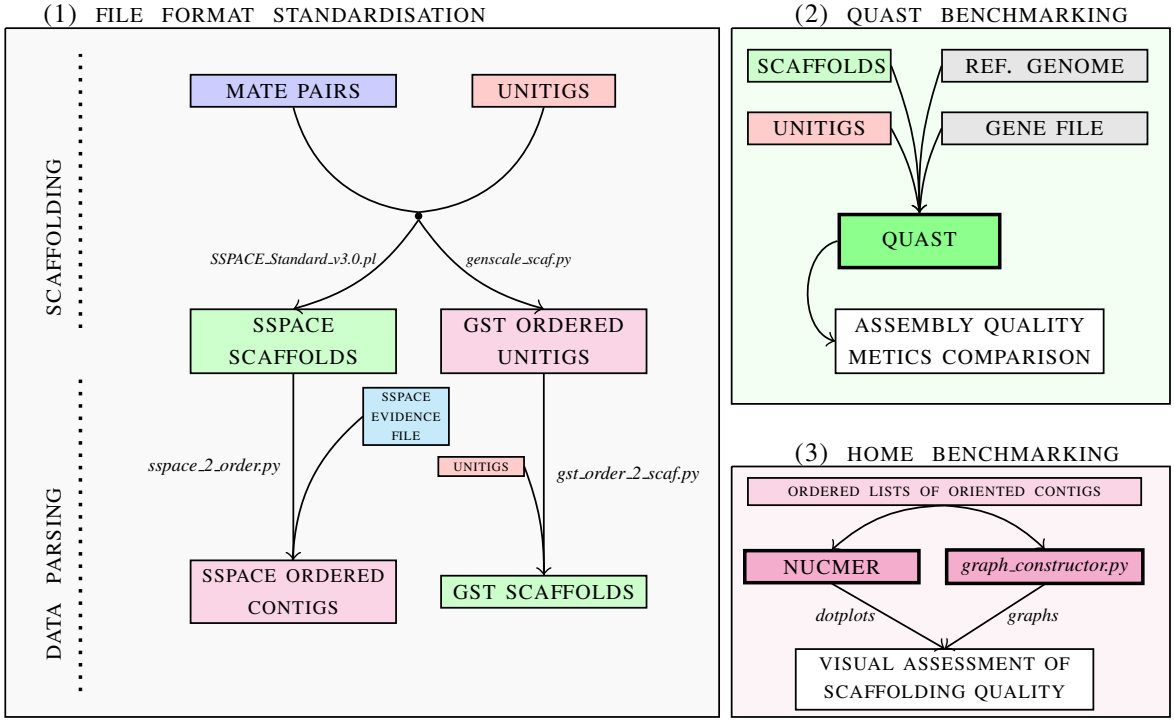
A visualization of the solution is necessary as Quast does not provide information on ordering of unitigs. The presence of misassemblies is an indication of a possible order mistake however it can also be small distance between unitigs errors resulting in wrong overlapping or extra Ns. A visual alignment of the scaffolding sequences with the reference genome is a sturdy method to evaluate the scaffolding continuity. Dotplots are generated with `nucmer`, of the MUMmer software. Two steps are necessary:

- `nucmer scaffolding.fasta ref_genome.fasta -l 500` aligns the two nucleic sequences and outputs a *.delta* file. The `-l` option registers all alignements which are superior to 500bp.
- `mummerplot nucmer_output.delta -t png` outputs a dotplot in *.png* format.

The MUMmer software also provides the `show-coords` tool which detects and outputs the list of coordinates of selected regions of the alignement. For example:

- `show-coords -l -T -L 1200 -q nucmer_alignement.delta` outputs a tab delimited file (`-T`) containing alignements longer than 1200bp (`-L`), including the sequence length information (`-l`).

Sequence continuity is evaluated thanks to this methodology, it does not however show the unitigs that board the eventuam misassemblies or continuity rupture. This is done with the graphical output of `graph_constructor.p` Unscaffolded unitigs and wrongly scaffolded unitigs are detected. Their environment in the input data is analyzed in order to understand the reason behind the scaffolding failure.



(1) SSPACE provides a *.fasta* file with built scaffolds. The `sspace_2_order.py` script converts it to the GST solution format. The GST do not compute *.fasta* sequences of unitigs. The scaffolding final result is an ordered list of unitigs. The `gst_order_2_scaf.py` script converts the solution to a *.fasta* format. If distances between unitigs are not available, the k-mer size with which the unitigs were built is used.
(2) The reference genome and gene file provide additional metrics for assembly comparison with Quast (number of found genes, percentage of genome assembled, N50 ...). All files in *.fasta* format are compared with Quast.
(3) Further inspection of scaffolding solutions is done thanks to `nucmer` alignment tool. `nucmer` dotplots scaffolding solutions against the reference genome so challenging regions can be detected. The `graph_constructor.py` aids the detection of unitigs which were wrongly assembled.

Figure 10 – Benchmarking workflow

3 Results

3.1 Comparison between the data sets

Results presented in table 1 are obtained with the `graph_inspector.py` script. The analysed graph is the one modeled by the GST and built by the `graph_generator.py` script (see rice example figure 8).

organism	G type	G size (bp)	#nodes	#edges	node degree	graph density	diameter	#periphery pn.	radius	#central pn.	status
agrostis	chpl.	136584	22	98	min 4 , max 13 , avg 8	0.212	6	1	4	6	solved
acineto	bacter	3598621	924	7984	min 0 , max 114 , avg 17	0.009	gnc:ipl	gnc:ipl	gnc:ipl	gnc:ipl	unsolved
acorus	chpl.	153821	30	204	min 4 , max 26 , avg 13	0.234	5	13	4	17	solved
atropa	chpl.	156687	52	268	min 5 , max 16 , avg 10	0.101	8	12	6	6	solved
cucumis	chpl.	155293	194	1446	min 4 , max 50 , avg 14	0.038	11	30	8	6	solved
eucalyptus	chpl.	160286	8	16	min 4 , max 4 , avg 4	0.285	4	8	4	8	solved
euglena	chpl.	143171	296	11894	min 12 , max 176 , avg 80	0.136	gnc:ipl	gnc:ipl	gnc:ipl	gnc:ipl	unsolved
lecomtella	chpl.	139073	22	90	min 4 , max 13 , avg 8	0.194	6	3	4	5	solved
oenothera	chpl.	163935	172	3086	min 4 , max 70 , avg 35	0.104	14	1	9	47	solved
pinus	chpl.	116866	122	628	min 4 , max 29 , avg 10	0.042	13	7	7	1	almost solved
rice	chpl.	134525	8	16	min 4 , max 4 , avg 4	0.285	4	8	4	8	solved
sacchar.	chr3	316613	370	4416	min 4 , max 113 , avg 23	0.032	13	1	6	37	unsolved
wolbachia	bacter	1482355	4270	616804	min 5 , max 2582 , avg 288	0.033	9	23	5	198	unsolved

G.: genome, *pn.*: pairs of nodes, *chpl.*: chloroplast, *bacter.*:bacterial, *chr3*: chromosome 3, *gnc:ipl*: "graph not connected: infinite path length".
node degree: the number of edges incident to the vertex. Self-loops are not allowed in this graph.
graph density: $D(G) = \frac{|L|}{|N|(|N|-1)}$ with $|L|$ the link cardinal and $|E|$ the node cardinal. A dense graph has a number of links close to the maximal number of link, and a density value close to 1.
graph diameter: greatest geodesic distance (shortest path between two nodes) for any node in the graph.
graph periphery: nodes with maximal eccentricity (geodesic distance = diameter).
graph radius: lowest geodesic distance for any node in the graph.
graph center: nodes with minimal eccentricity (geodesic distance = radius).

Table 1 – GST modeled graph features

The minimal degree for a node in the graphs should be 4 because a node has at least an in-going link and an outgoing link ($indegree = outdegree = 1$). The graph also models their reverse equivalents. Acineto has a node degree equal to 0. It means an unitig is completly disconnected from the rest of the genome. This explains the "graph not connected" error and why it is unsolvable. Furthermore the density of the graph is very low. Euglena however has a minimum node degree of 12. There are no nodes with indegree or outdegree equal to 0 (or 12). This could indicate that the euglena graph holds two disconnected sub-graphs. The two smallest graphs have the highest graph density: the graphs have 8 nodes with $indegree = outdegree = 1$, ($\times 2$ because of the reverse equivalent links). All links are overlaps. All degrees are all equal to 4. Lower densities indicate that not all consecutive unitigs have joins between them and their consecutive ordering need to be deduced from mated-pair information with other non consecutive unitigs. The three solved instances with the most nodes (cucumis, oenothera and pinus) have low graph density and also have the highest diameter and radius. Distance between nodes are big. Graph periphery in these instances contains mainly short repeat nodes which lack mated-pairs. These nodes are poorly connected to the rest of the graph despite the need for them to be used multiple times.

At the beginning of the project the idea of detecting inconsistencies in the input data by comparing input links with the expected solution links was tested. These results are presented for the chloroplastic instances in table 2. Links in the input solution are links between consecutive unitigs (no mated-pair reads obviously). The high number of links in the input absent from the expected solution comes from the fact that mated-pairs usually link non consecutive unitigs. This strategy has lead to some explanations for GST' difficulties in scaffolding pinus. A link in the expected solution (link between two consecutive unitigs) was absent in

the input data. As previously seen, consecutive unitigs are not always joined in the input data. However in pinus, the unitigs 14 and 15 were linked in the `14 Forward -> 15 Reverse` order when the expected order was `14 Reverse -> 15 Forward` (these two links are not reverse equivalent). This lead to a wrong graph model. The pinus data was eventually re-simulated and solved with the flow model.

organism	links in I and not in ES	links in ES and not in I
eugalyptus	0 (0%)	0
atropa	6 (42.8%)	0
lecomtella	11 (64.7%)	0
oenothera	18 (43.9%)	0
cucumis	79 (69.9%)	0
pinus	23 (60.5%)	2
euglena	79 (73.1%)	0

I:input data, ES:expected solution

Table 2 – Comparison between the input linking data and the expected solution's links

3.2 Comparison of GST solutions with the expected solution

Valid solutions were detected with the `graph_comparator.py` script. Other sub-optimal solutions were looked for manually. The GST performance are presented in table 3.

organism	expected solution found	time	partial solution found	time	note
agrostis	wpm	0.5s	-	-	1 valid solution among 4, sub-optimal found with dist and flow
acineto	-	-	wpm	0.6s	ordering big unitigs correctly
acorus	wpm	0.5s	-	-	1 valid solution among 2, also found with dist and flow
atropa	wpm	1s	-	-	1 valid solution among 2, also found with flow
cucumis	wpm	35s	-	-	several valid solution among 18
eucalyptus	wpm	0.2s	-	-	1 valid solution among 2, also found with dist and flow
euglena	-	-	-	-	instance never fully or partially solved
lecomtella	wpm	0.6s	-	-	1 valid solution among 4, sub-optimal found with dist
oenothera	wpm	163s	-	-	1 valid solution among 4
pinus	-	-	flow	0.2	instance never solved with wpm or dist, good frame found with flow
rice	flow	0.1s	-	-	also found with wpm and dist
sacchar.	-	-	-	-	instance never fully or partially solved
wolbachia	-	-	flow	0.4s	built frame with large unitigs resulting in 11 disconnected scaffolds

Table 3 – Genscale scaffolders solutions

Pinus and euglena are the two chloroplastic genomes which were not fully solved by first generation GST tools. They are incidentally the two chloroplasts not to possess an inverted repeat. Pinus has a significantly shorter genomic length. As noticeable in table 1, the number of nodes is not linked to the genome size but rather to the amount of repeated sequences assembled into separate unitigs. Pinus is partially solved by the first step of the flow model (92.1% of the genome assembled). Bacterial genomes are very challenging to scaffolds for reasons described in section section 2.1.2 page 9. Building a genomic framework is now possible with the flow model, but the full genome has yet to be scaffolded. Results for wolbachia are further discussed in section section 3.4 page 21.

* * *

Next sections present results obtained with the benchmarking workflow. Promising results are detailed using the agrostis example, but the excellent performance of GST for agrostis is also valid for other chloropastic genomes (except pinus and euglena). The improvement in scaffolding made with the flow model will be detailed for pinus. But one must keep in mind that for now only the first frame building step of the flow model was benchmarked.

3.3 Comparison of the best GST solutions and SSPACE to the reference genome

As a reminder, the reason why SSPACE was chosen is because it performs very well according to the comparative scaffolding study[?] and it does not explicitly discard repetitive regions.

3.3.1 Quality assessment of genome assemblies of *Agrostis stolonifera* chloroplastic genome

QUAST *Agrostis stolonifera* chloroplastic genome was successfully assembled with the Weighted path model. The wpm provides 4 solution, among them two are valid ones (1^{st} and 2^{nd}) and one (1^{st}) is the exact copy of the expected solution (the reference genome). The wpm scaffold solution was rebuilt using the size of the k-mer used during the unitig building step (60bp) in place of unitig distance information. Each unitig overlaps its unitig neighbors for 59bp. This method resulted in an exact solution (100% of the genome was assembled, same size). The distance solution is also a satisfying solution however the sizes of the distances between contigs are false and cause the scaffold to be wrongly gapped. The added Ns in the found false gaps increases the size of the genome by 561bp. This is a negligible amount and is not detected by the *N's per 100kpb* metric. The largest alignment is quite low for the distance based model (102364bp before making a misassembly). This length comprises the LSC and an IR, meaning that distance based fails to find the correct distance between the SSC and the IRs.
All of the GSTs performed better than the SSPACE scaffolder. Among the 6 unitigs of the agrostis input data, SSPACE did not scaffold the small 160bp unitig and the big 12928bp extended unitig. The three SSPACE scaffolds are:

- the 160bp contig (0) which is part of the IR ($IR = c3 + c0 + c1$)
- the LSC ($LSC = c4 + c0 + c2$) and an incomplete IR copy lacking the 160bp contig
- the SSC ($SSC = c5$)

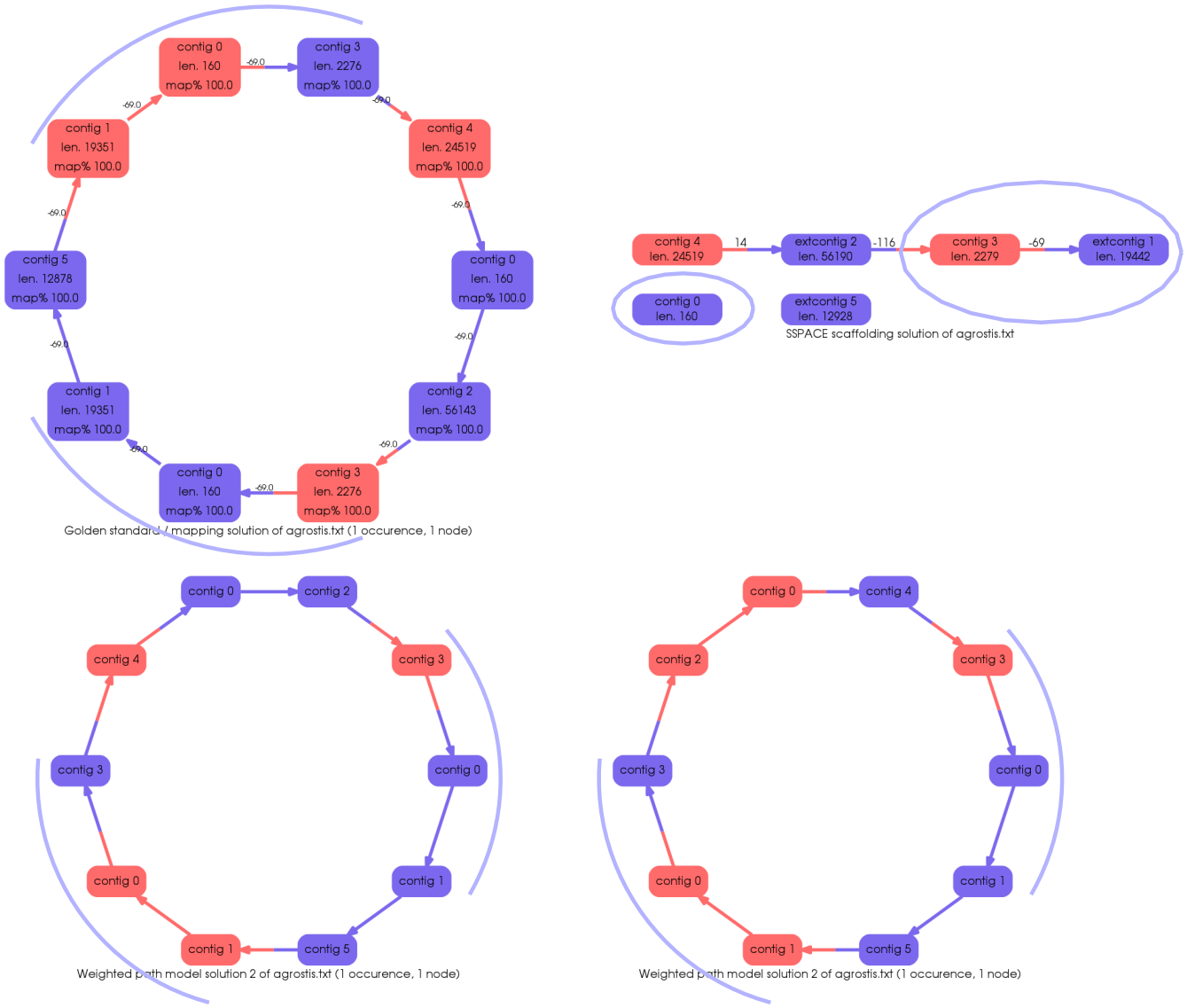
The 160bp contig and the SSC possess conflicting links, and SSPACE is not able to take a decision so it does not take the risk of scaffolding them. It does not find the whole genome: only 84.5% is assembled, which is a 0.3% more than the sum of all unitig sizes. This slight increase occurs because the extension though unused reads was permitted ($-x\ 1$ option). However SSPACE can not go much beyond this percentage as it does not compute unitig coverage and is not meant to duplicate any given unitig.

Assembly	unitigs	wpm_sol1.fsa	wpm_sol2.fsa	wpm_sol3.fsa	wpm_sol4.fsa	dist_sol	sspace_sol	ref_genome
# scaffolds (≥ 0 bp)	6	**1**	**1**	**1**	**1**	**1**	3	**1**
# scaffolds (≥ 1000 bp)	5	**1**	**1**	**1**	**1**	**1**	2	**1**
Total length (≥ 0 bp)	115327	136584	136584	136584	136584	**137145**	115344	136584
Total length (≥ 1000 bp)	115167	136584	136584	136584	136584	**137145**	115184	136584
# scaffolds	5	**1**	**1**	**1**	**1**	**1**	2	**1**
Largest scaffolds	56143	136584	136584	136584	136584	**137145**	102256	136584
Total length	115167	136584	136584	136584	136584	**137145**	115184	136584
Reference length	136584	136584	136584	136584	136584	136584	136584	136584
GC (%)	37.39	38.45	38.45	38.45	38.45	38.45	37.40	38.45
Reference GC (%)	38.45	38.45	38.45	38.45	38.45	38.45	38.45	38.45
# misassemblies	**0**	**0**	2	1	1	2	1	**0**
# misassembled scaffolds	**0**	**0**	1	1	1	1	1	**0**
Misassembled scaffolds length	**0**	**0**	136584	136584	136584	137145	102256	**0**
# local misassemblies	**0**	**0**	**0**	**0**	**0**	1	**0**	**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
Unaligned length	0	0	0	0	0	0	0	0
Genome fraction (%)	84.219	**100.000**	**100.000**	98.657	98.657	91.522	84.590	**100.000**
Duplication ratio	1.001	1.000	1.155	1.014	1.014	1.097	**0.997**	1.000
# N's per 100 kbp	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	218.75	12.15	**0.00**
# mismatches per 100 kbp	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**	**0.00**	**0.00**	5.60	0.87	**0.00**
# genes	110 + 4 part	132 + 1 part	**133 + 0 part**	131 + 0 part	131 + 0 part	122 + 1 part	114 + 2 part	**133 + 0 part**
Largest alignment	56143	**136584**	81612	82913	134750	102364	81140	**136584**
NA50	24519	**136584**	81612	82913	134750	102364	81140	**136584**

All statistics are based on contigs of size ≥ 50 bp. wpm: weighted path model, dist: distance based model.

Table 4 – QUAST metrics for several unitig scaffoldings of *Agrostis stolonifera* with GST (wpm and dist) and SSPACE

graph_generator.py Figure 11 is a *graph_generator.py* visualization of the scaffolding solutions compared with Quast in table 4. The circled or lined unitigs are those forming the IR of agrostis. This technique visually confirms that the unitigs that SSPACE found challenging to scaffold are the repeated ones. The difference between the two valid solutions found by wpm is also highlighted. The LSC (*c2-c4-c4*) region is reversed but the SSC (*c5*) has the same orientation in both solutions.



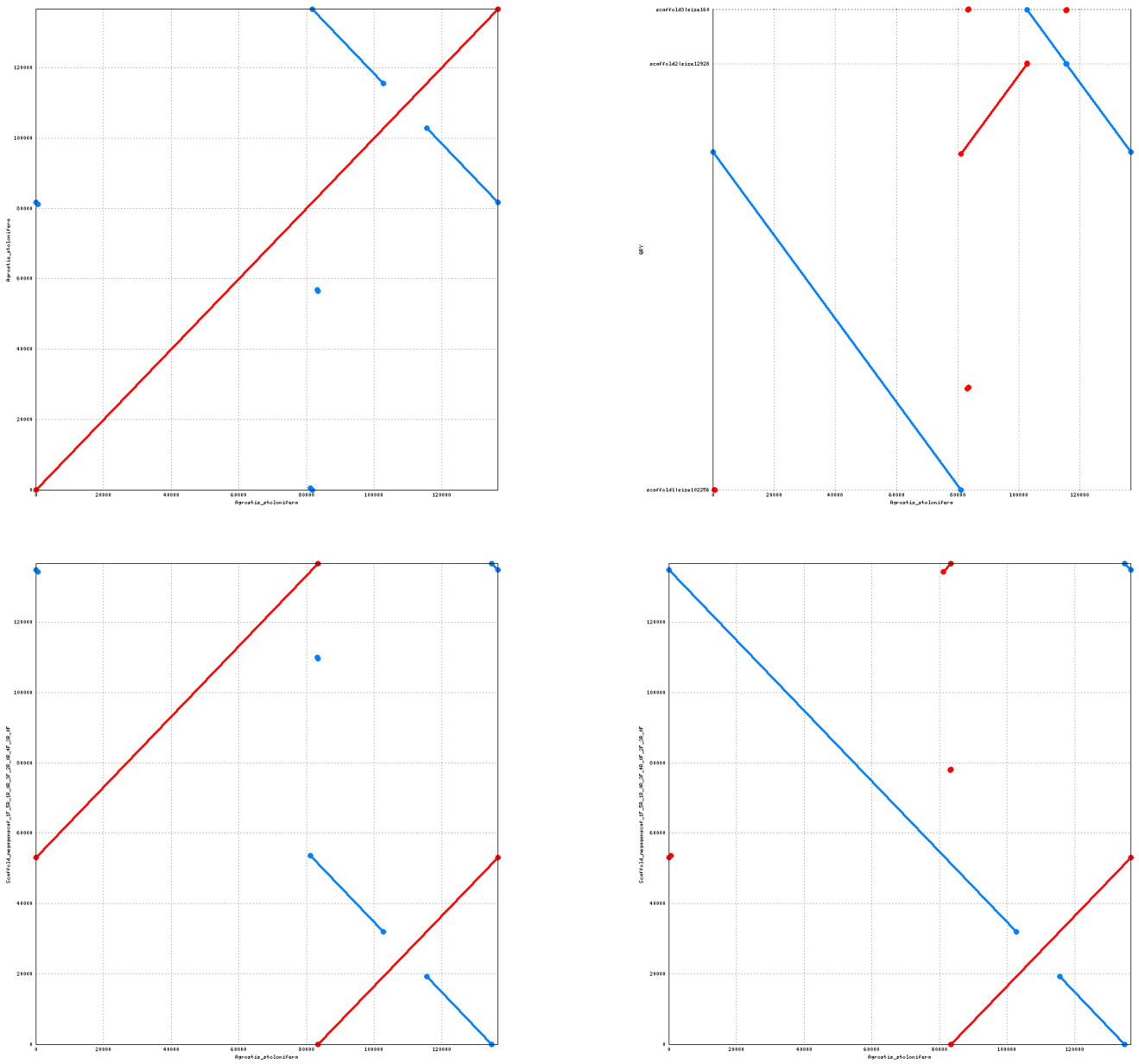
The expected solution graph (top left) has the contigs correctly ordered, displays their length and their mapping score. The expected solution is obtained by mapping unitigs on the reference genome. The two bottom graphs are two valid wpm solutions and the top right graph is the SSPACE scaffolder solution.

```
> graph_constructor.py -g [expt,whpm,sspace] -f solution_file.txt
```

Figure 11 – Expected solution and scaffolding solutions of weighted-path model and SSPACE scaffolders

MUMer The dotplots further illustrates the differences between scaffolding solution. Dotplots do not visualize unitigs but rather the whole scaffolding sequence. If the sequence on the x axis matches the sequence of the y axis in the "$right$" order (both same orientation), a red matching line is drawn. For the control dotplot, where the reference genome is dotplotted against itself a perfect match can be seen. Additionally the inverted repeats are seen as an additional match in the "$wrong$" order (as they are inverted

and the orientation is different) and appear as blue lines. Additional little inverted regions are detected. The control dotplot and the first solution of the wpm display the exact same dotplot profile (just shifted, as the fasta files do not start at the same genomic position). In the second solution, a reversed match exists but the small single copy is reversed: the reverse match is interrupted.
The SSPACE solution has three scaffolds (3 scaffold names on the side of the graph) and only one occurrence of the IR. Additionally the LSC and IR are scaffolder in the wrong relative orientation which indicates the coordinate of the misassembly detected bu Quast (around the 81000bp).



Control (top left), sspace (top right), wpm solutions (bottom). The control is the reference aligned to itself.

```
> nucmer ref_genome.fasta [solution, reference, unitigs] > align.delta
> mummerplot align.delta
```

Figure 12 – Control dotplot and dotplots of weighted path model and SSPACE scaffolding solutions with the reference genome on the x axis

3.3.2 Quality assessment of genome assemblies of *Pinus koraiensis* chloroplastic genome

QUAST As a reminder, *Pinus koraiensis* chloroplastic genome does not possess an inverted repeat. The genome is smaller than the average size for a chloroplatic genome. The pinus instance has 33 unitigs, among them only 10 are bigger than 1000bp. The input graph is shown in annexe A. This instance possesses many small repeats and is difficult to assemble. Some unitigs are as small as 76bp. Knowing that the contig builder `minia` built pinus unitigs with a k-mer size of 60, this is dissapointingly small, and smaller than the read size (150bp). These unitigs are solely linked by overlaps (as the chance of having a mated-pair on them is low, or null for unitigs smaller than the read size). Knowing these unitigs are repeats, their overlaps are hard to trust.

Pinus is never correctly solved by first generation GST (wpm and dist) and only partially solved by the first step of the flow model. Quast results suggests that the best assembly is obtained with SSPACE as there are only 2 misassemblies for a 99.254% of assembled genome. However SSPACE assembles 12 large unitigs into a single scaffold then ignores all other unitigs (the remaining 21 which appear alone). SSPACE extends the unitigs through reads and uses 886 N to achieve the 99.254% assembled genome score. This scaffold also contains two missassemblies, indels and mismatches. All these metrics are worse than those for the 1^{st} flow model step.

Only 1% of the genome are repeats (unitig genome fraction $= 99.079$). Wpm duplicates some contigs and achieves a 99.4% score however has 11 misassemblies which is extremely disappointing. Hopes for the flow model are high despite it assembling only 92% of the genome: it has the high NA50, meaning that for 96535 nucleotides the assembly is perfect. SSPACE's first missassembly occurs at the 86569bp. Nucmer alignements for these scaffolding solutions are shown in annexe B.

Assembly	unitigs	flow_step1_sol	wpm_sol1.fsa	sspace_sol	ref_genome
# contigs (≥ 0 bp)	33	2	**1**	21	**1**
# contigs (≥ 1000 bp)	10	**1**	**1**	**1**	**1**
Total length (≥ 0 bp)	118629	116214	116866	**119654**	116866
Total length (≥ 1000 bp)	115186	115838	116866	**117042**	116866
# contigs	11	**1**	**1**	**1**	**1**
Largest contig	20834	115838	116866	**117042**	116866
Total length	115888	115838	116866	**117042**	116866
Reference length	116866	116866	116866	116866	116866
GC (%)	38.77	38.86	38.80	38.79	38.80
Reference GC (%)	38.80	38.80	38.80	38.80	38.80
N50	14862	115838	116866	**117042**	116866
# misassemblies	**0**	1	11	2	**0**
# misassembled contigs	**0**	1	1	1	**0**
Misassembled contigs length	**0**	115838	116866	117042	**0**
# local misassemblies	**0**	4	4	5	**0**
# unaligned contigs	0 + 0 part	0 + 0 part	0 + 0 part	0 + 0 part	0 + 0 part
Unaligned length	0	0	0	0	0
Genome fraction (%)	99.079	92.153	99.471	99.254	**100.000**
Duplication ratio	1.001	1.076	1.005	1.010	**1.000**
# N's per 100 kbp	**0.00**	6997.70	**0.00**	757.85	**0.00**
# mismatches per 100 kbp	**0.00**	**0.00**	5.16	0.86	**0.00**
# indels per 100 kbp	2.59	1.86	8.60	2.59	**0.00**
# genes	261 + 6 part	249 + 4 part	262 + 7 part	262 + 6 part	**270 + 0 part**
Largest alignment	20834	96535	28559	86569	**116866**
NA50	14862	96535	20762	86569	**116866**

All statistics are based on contigs of size ≥ 50 bp. wpm: weighted path model, flow_step1: step 1 of flow model.

Table 5 – QUAST metrics for several unitig scaffoldings of *Pinus koraiensis* with GST (wpm, flow model) and SSPACE

3.4 Partially solved instance - flow model

Wolbachia is partially assembled by the first step of the flow model. Table 6 highlights the importance of correctly choosing the mated-pair reads' insert size. Results 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). One proposed explanation is the repeats' sequence size of *Wolbachia Endosymbiont*. As previously seen in figure 7, the bacterial genome has repeats with sizes mainly between 500bp and 1.2Kbp. 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 the exact coordinates of those location see annexe C. 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 repeats. Using multiple libraries would be a solution however this strategy didn't result in encouraging scaffolding solutions with SSPACE (library i2000 and i5000 used simultaneously). Another interesting metric to notice is the genome fraction covered by unitigs, which is very low (90.3%) considering that the genome is 1.4Mbp and repeats rarely exceed 1.2Kb . This is also noticeable in figure 13, where rather larger portions of the reference genome are not covered by unitigs. The unitig building step failed to produce large sequences covering these areas because they contain many very small ($\leq$ read size) tandem repeats. Unless filled with Ns, these areas will never be scaffolded. Obtaining a whole circular genomes appears to be a very difficult task, but obtaining larger scaffolds should be possible.

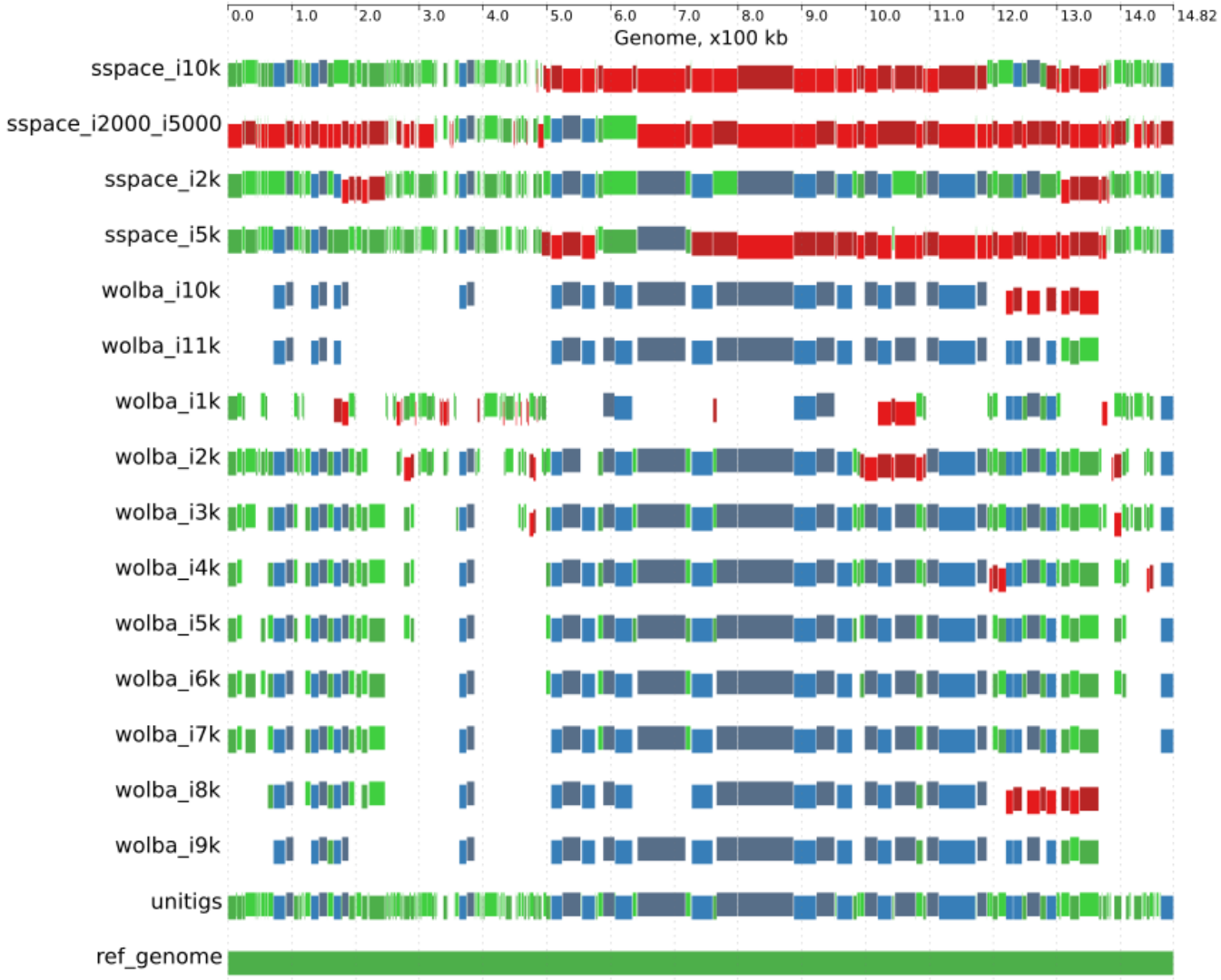
Assembly	unitigs	wolba_i10k	wolba_i11k	wolba_i1k	wolba_i2k	wolba_i3k	wolba_i4k	wolba_i5k	wolba_i6k	wolba_i7k	wolba_i8k	wolba_i9k	ref_genome
# scaffolds ($\geq$ 0 bp)	444	9	7	37	28	19	13	13	13	11	9	9	**1**
# scaffolds ($\geq$ 1000 bp)	138	9	7	37	28	19	13	13	13	11	9	9	**1**
Total length ($\geq$ 0 bp)	1364357	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**
# 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**

All statistics are based on contigs of size $\geq$ 50 bp.
Wolbachia has 444 unitigs, among them 306 are smaller than 1000bp. In this comparison, the higher the number of scaffolds the better because it indirectly depicts the number of unitigs the 1^{st} step of the flow model considers as big unitigs. It impacts the total length of the assembled genome which is the largest with an insert size of 2000bp (1.2Kb). The largest perfect alignement found is with an insert size of 5000bp (459Kb) indicating this insert size is the upper limit to use for wolbachia. For all solution the number of missassemblies is very high and due to the fact that they are heavily gapped and the gap size not exact.

Table 6 – Scaffolding solutions of the flow model step 1 with mated-pair reads libraries of different insert sizes compared to the reference genome and the initial unitig set.

Figure 13 shows the alignement of all scaffolding strategies tested for wolbachia to the reference genome. Additionally the unitigs were also aligned to the reference genome to highligh the heavily uncovered 350Kbp - 500Kbp and 1380Kpb - 1440Kpb regions. SSPACE solution with the 2Kbp insert size library

shows the less misassembled regions with the most reference coverage however this scaffolder still has more misassembled regions than the worst of the flow model solutions. At the end of the SSPACE scaffolding process, there are 338 separate scaffolds left (not shown). This means small unitigs were not used and SSPACE mainly extended big existing unitigs with reads. SSPACE also fails to produce any valid scaffold for the genomic regions containing the large (>Kbp) repeats (see their exact coordinates in annexe C).



This plot shows alignment of unitigs or scaffolds to the reference genome and positions of misassemblies in the scaffolds. Correctly aligned big (> 10kb) scaffolds are blue if the boundaries are exact, and green if the boundaries are different from the reference genome region it was mapped to. Scaffolds containing an important amount of misassemblies are red.

Figure 13 – Alignment of scaffolding solutions on the reference genome of Wolbachia endosymbiont

The repeated unitigs should be introduced in the 2^{nd} step of the flow model. The fraction of the assembled genome should be at least as good as SSPACE. However the strategy used for this second step will decide if the order and orientation will be satisfactory. As of now, two strategies are considered. The first is an interative gap by gap approach, which is faster but whose solution depends heavily of the order in which the gaps are treated. The second approach is filling all gaps at once. This approach seems more sensible but will result in a combinatorial problem with many possible solutions.

4 Conclusion

The GENSCALE scaffolding methodology basis is the computing and pre-processing of unitig coverage. Different modeling strategies are tested and evaluated to solve the scaffolding problem. Adding weight on arcs to model the distance between unitigs does not lead to better results. These additional constraints are too demanding. An evaluation strategy was set up to understand why some data sets are especially challenging. The repeat content impacting on the unitig number rather than the size of the genome is the cause of complex input data. Some explanations are provided for problematic data sets (disconnected graph or missing link) however the main source of difficulties is the size of the modeled graph. A new two step scaffolding modeling strategy is in development. It tries to break the graph complexity by first solving a graph containing only large unitigs - building something that can be compared to a trustworthy genomic frame. The benchmarking workflow brings together several sequence comparing tools: a tool for assessing assembly quality (Quast), a sequence aligner (MUMmer) and homemade visualization and comparison scripts (`graph_generator.py` and `graph_comparato.py`). Although compared to a single published scaffolder (SSPACE), this methodology can be applied for any scaffolding solution obtained with other tools. Comparing the GST to recent publications such as the ScaffMatch scaffolder or the Integer Linear Programming approach developed at the Montpellier Laboratory of Informatics, Robotics and Microelectronics (LIRMM) which explicitly discusses its repeated sequence processing will be insightful. Their article was the motivation behind the study on *Wolbachia Endosymbiont*, one of their tested organism. The benchmarking of the GST highlights the major advantage of processing unitig coverage but also the limitation of the models which have difficulties with bigger graphs with high degree nodes. Overall the project succeeded more in the standardisation of evalution and benchmarking strategies than on providing precise explanations for unsuccessful scaffoldings. The immediate perspective is to help guide the development of the flow model.

Future applications

MinION[27]

References

References

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

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

[3] Earl, D. *et al.* Assemblathon 1: A competitive assessment of de novo short read assembly methods **21**, 2224–2241.

[4] Bradnam, K. R. *et al.* Assemblathon 2: evaluating de novo methods of genome assembly in three vertebrate species **2**, 10.

[5] Pop, M., Kosack, D. S. & Salzberg, S. L. Hierarchical scaffolding with bambus **14**, 149–159.

[6] Treangen, T. J. *et al.* MetAMOS: a modular and open source metagenomic assembly and analysis pipeline **14**, R2.

[7] Zerbino, D. R. & Birney, E. Velvet: Algorithms for de novo short read assembly using de bruijn graphs **18**, 821–829.

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

[9] Huson, D. H., Reinert, K. & Myers, E. W. The greedy path-merging algorithm for contig scaffolding **49**, 603–615.

[10] Petrov, I. & Andonov, R. New scaffolding methodologies (2014).

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

[12] Langmead, B. & Salzberg, S. L. Fast gapped-read alignment with bowtie 2 **9**, 357–359.

[13] Langmead, B., Trapnell, C., Pop, M. & Salzberg, S. L. Ultrafast and memory-efficient alignment of short DNA sequences to the human genome **10**, R25.

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

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

[16] Boetzer, M., Henkel, C. V., Jansen, H. J., Butler, D. & Pirovano, W. Scaffolding pre-assembled contigs using SSPACE. *Bioinformatics (Oxford, England)* **27**, 578–579 (2011).

[17] Dayarian, A., Michael, T. P. & Sengupta, A. M. SOPRA: Scaffolding algorithm for paired reads via statistical optimization. *BMC Bioinformatics* **11**, 345 (2010).

[18] Donmez, N. & Brudno, M. SCARPA: scaffolding reads with practical algorithms. *Bioinformatics (Oxford, England)* **29**, 428–434 (2013).

[19] Salikhov, K., Sacomoto, G. & Kucherov, G. Using cascading bloom filters to improve the memory usage for de brujin graphs. In *Algorithms in Bioinformatics*, 364–376 (Springer, 2013).

[20] Chikhi, R. & Rizk, G. Space-efficient and exact de bruijn graph representation based on a bloom filter. In *WABI*, vol. 7534 of *Lecture Notes in Computer Science*, 236–248 (Springer, 2012).

[21] Delcher, A. L. *et al.* Alignment of whole genomes. *Nucleic Acids Research* **27**, 2369–2376 (1999).

[22] Delcher, A. L., Phillippy, A., Carlton, J. & Salzberg, S. L. Fast algorithms for large-scale genome alignment and comparison. *Nucleic Acids Research* **30**, 2478–2483 (2002).

[23] Gansner, E. R. & North, S. C. An open graph visualization system and its applications to software engineering. *SOFTWARE - PRACTICE AND EXPERIENCE* **30**, 1203–1233 (2000).

[24] Ellson, J., Gansner, E. R., Koutsofios, E., North, S. C. & Woodhull, G. Graphviz and dynagraph – static and dynamic graph drawing tools. In *GRAPH DRAWING SOFTWARE*, 127–148 (Springer-Verlag, 2003).

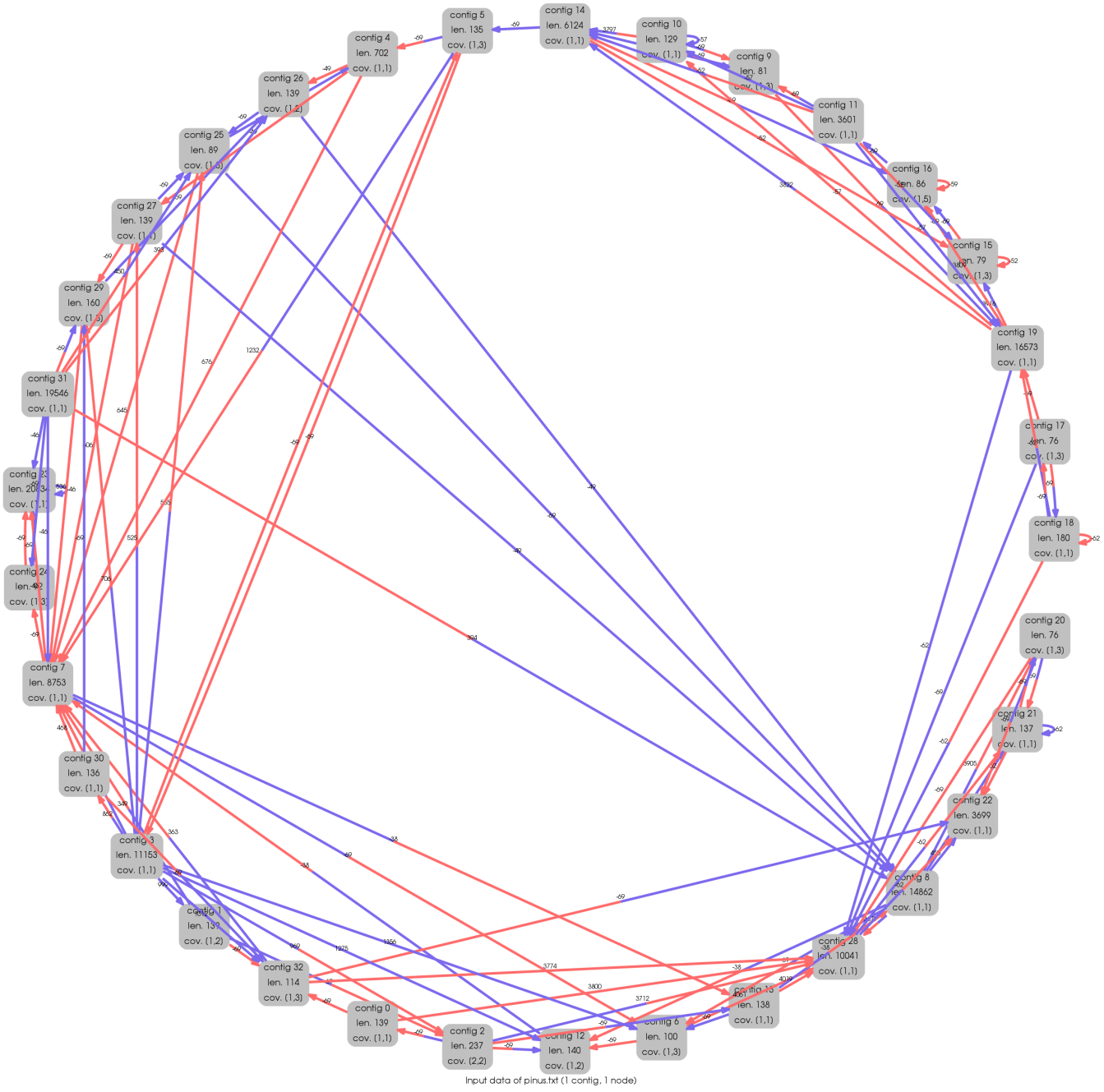
[25] Hagberg, A. A., Schult, D. A. & Swart, P. J. Exploring network structure, dynamics, and function using NetworkX. In *Proceedings of the 7th Python in Science Conference (SciPy2008)*, 11–15 (Pasadena, CA USA, 2008).

[26] Gurevich, A., Saveliev, V., Vyahhi, N. & Tesler, G. QUAST: quality assessment tool for genome assemblies. *Bioinformatics (Oxford, England)* **29**, 1072–1075 (2013).

[27] Ashton, P. M. *et al.* MinION nanopore sequencing identifies the position and structure of a bacterial antibiotic resistance island. *Nature Biotechnology* **33**, 296–300 (2015).

Annexes

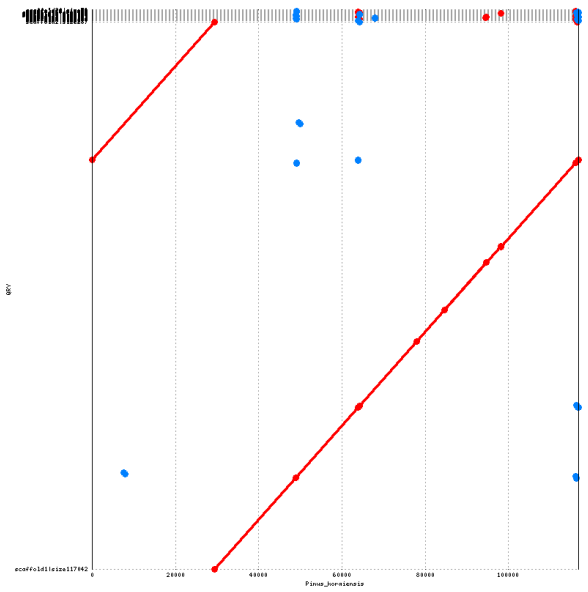
Figure A – Input graph of pinus

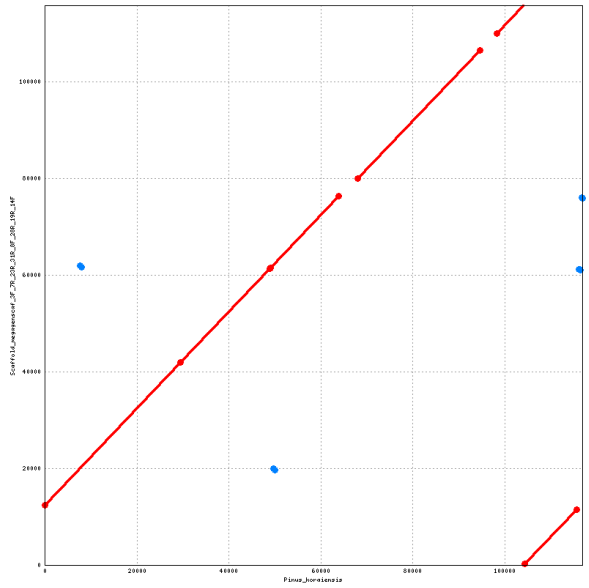


This graph is a graphical transcription of the *.txt* input file of pinus. The graph as modeled by GST is by far bigger. Here we see several self-loops and multiple equivalent links (see 3 identical links between unitig 30 and unitig 7). These will be merged when once the GST graph models the problem. A strategy would be to keep the link coverage as additional constrains to hopefully help find the optimal path. Most of the contigs are shorter than 1000bp (22) and some as small as 76bp. Knowing that the contig builder `minia` built pinus unitigs with a k-mer size of 60, this is dissapointingly small, and smaller than the read size (150bp). These unitigs are solely linked by overlaps (as the chance of having a mated-pair on them is low, or null for unitigs smaller than the read size). Knowing these unitigs are repeats, their overlaps are hard to trust . . .

Figure B – Dotplots of weighted path model, flow model 1st step and SSPACE scaffolding solutions with the reference genome on the x axis

Top left graph is the SSPACE solution, top right the flow model solution and bottom left the weighted path solution. The weighted path solution single scaffold contains many misassemblies - a lot of unitigs are scaffolded in the wrong order and orientation. The SSPACE solution contains many scaffolds because it did not scaffold small unitigs (overlapping unitig names on the left top of the graph). SSPACE extended the big unitigs and introduced Ns to be able to built the big scaffold matching the reference genome - though it contains misassemblies, mismatches and indels visualized as buldges (or little circles) on the main red matching line. The flow model dotplot also shows an almost perfect match, with gaps to be filled in with the second step of the model. The aim is to obtain a scaffold with less N and errors than SSPACE, as well as a better genome assembled fraction.





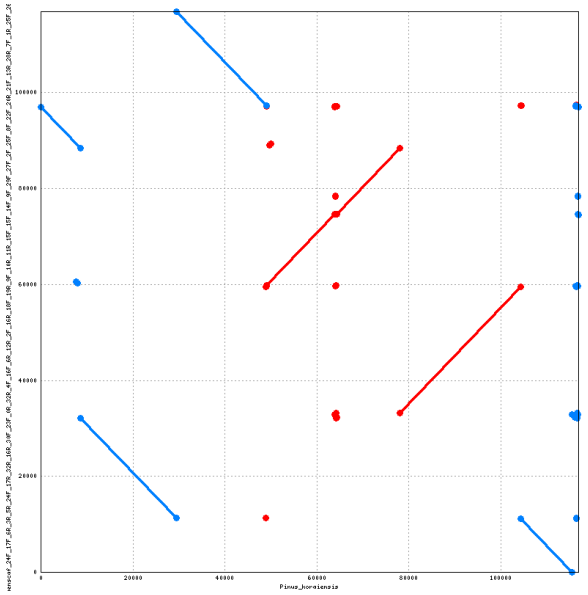


Figure C – Coordinates and length of repeats longer than 1.2Kbp in the *Wolbachia endosymbiont* genome

Obtained with the `show-coord` tool of the MUMmer software after the `nucmer` alignement of the reference genome to itself. Colored rows are coordinates of repeats longer than 2Kbp. *S1*: start coordinate of sequence of x axis
E1: end coordinate of sequence of x axis
S2: start coordinate of sequence of y axis (here, same sequence, the reference genome)
E2: end coordinate of sequence of y axis (here, same sequence, the reference genome)
LEN: length of the matched region
IDY: indentity percentage of the matched region

[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

BIOINFORMATICS AND GENOMICS MASTER'S DEGREE
UNIVERSITY RENNES 1

ALEXANDRINA BODRUG - JUNE 22, 2015

GENSCALE - IRISA

Abstract [eng] *Next Generation Séquencing, scaffolding strategy, repeated sequence, data complexity assessment, benchmarking*

The last step of the NGS data assembling process is scaffolding - the ordering and relative orientation of large genomic sequences called contigs. The scaffolding process remains a challenging computational work because of the possible errors of NGS data and the intrinsic characteristics of genomes, such as repeated regions. The GENSCALE team develops new scaffolding methodologies which rely on contig coverage to solve the scaffolding problem of highly repeated genomes. The detection, processing and integration of repeated sequences in the scaffolding solution is a defining feature of these methodologies. This report describes the strategies used for benchmarking these tools with published scaffolders and the evaluation of the solutions found given the known data and the expected solution. The data used for this project is simulated from small genomes containing many repeats - namely chloroplastic genomes and bacterial genomes. Limits and ways to optimise the GENSCALE scaffolders are outlined.

Résumé [fr] *Séquençage Haut Débit, stratégie d'échafaudage, séquence répétée, , estimation de la compléxité des données, analyse comparative*

La dernière étape du processus d'assemblage de données NGS est le scaffolding - l'ordonnancement et l'orientation relative de longues séquences génomiques appelées contigs. Le scaffolding est un processus difficile à cause d'erreurs possible dans les données NGS et des caractéristiques intrinsèques des génomes, telle que les régions répétées. L'équipe GENSCALE développe de nouvelles méthodologies de scaffolding qui se basent sur la couverture des contigs pour résoudre les problèmes de scaffolding posées par les génomes hautement répétés. La détection, le traitement et l'intégration des régions répétées dans la solution de scaffolding est une particularité de ces méthodologies. Ce rapport décrit les stratégies utilisées pour l'analyse comparative de ces outils avec des scaffoldeurs publiés et l'évaluation des solutions obtenues en connaissance de la nature des données en entrée et de la solution attendue. Les données utilisées dans ce projet sont simulées à partir de petits génomes contenant beaucoup de répétitions - à savoir des génomes chloroplastiques et des génomes bactériens. Sont également exposées dans ce rapport les limites et les possibilités d'optimisation des scaffoldeurs GENSCALE.