Bioinformatics, 31(15), 2015, 2443–2451
doi: 10.1093/bioinformatics/btv171
Advance Access Publication Date: 25 March 2015
Original Paper

OXFORD

Genome analysis

MeDuSa: a multi-draft based scaffolder

Emanuele Bosi[1,2,†], Beatrice Donati[3,4,5,†], Marco Galardini[6], Sara Brunetti[7], Marie-France Sagot[3,4,8], Pietro Lió[9], Pierluigi Crescenzi[5], Renato Fani[1,2] and Marco Fondi[1,2,*]

[1]Department of Biology, ComBo, Florence Computational Biology Group, [2]Department of Biology, LEMM, Laboratory of Microbial and Molecular Evolution Florence, University of Florence, I-50019 Sesto F.no, Italy, [3]INRIA Rhône-Alpes, Villeurbanne Cedex, France, [4]Université de Lyon, F-69000 Lyon, France, [5]Dipartimento di Ingegneria dell'Informazione, University of Florence, I-50139 Firenze, Italy, [6]EMBL-EBI - European Bioinformatics Institute, Wellcome Trust Genome Campus, Hinxton, CB10 1SD Cambridge, UK, [7]Dipartimento di Ingegneria dell'Informazione e Scienze Matematiche, University of Siena, Siena I-53100, Italy, [8]Université Lyon 1, CNRS,UMR5558, 69622 Villeurbanne Cedex, France and [9]Computer Laboratory, University of Cambridge, CB3 0FD Cambridge, UK

*To whom correspondence should be addressed.
[†]The authors wish it to be known that, in their opinion, the first two authors should be regarded as Joint First Authors.
Associate Editor: Inanc Birol

Received on July 23, 2014; revised on February 2, 2015; accepted on March 19, 2015

Abstract

Motivation: Completing the genome sequence of an organism is an important task in comparative, functional and structural genomics. However, this remains a challenging issue from both a computational and an experimental viewpoint. Genome scaffolding (i.e. the process of ordering and orientating contigs) of *de novo* assemblies usually represents the first step in most genome finishing pipelines.

Results: In this article we present MeDuSa (Multi-Draft based Scaffolder), an algorithm for genome scaffolding. MeDuSa exploits information obtained from a set of (draft or closed) genomes from related organisms to determine the correct order and orientation of the contigs. MeDuSa formalizes the scaffolding problem by means of a combinatorial optimization formulation on graphs and implements an efficient constant factor approximation algorithm to solve it. In contrast to currently used scaffolders, it does not require either prior knowledge on the microrganisms dataset under analysis (e.g. their phylogenetic relationships) or the availability of paired end read libraries. This makes usability and running time two additional important features of our method. Moreover, benchmarks and tests on real bacterial datasets showed that MeDuSa is highly accurate and, in most cases, outperforms traditional scaffolders. The possibility to use MeDuSa on eukaryotic datasets has also been evaluated, leading to interesting results.

Availability and implementation: MeDuSa web server: http://combo.dbe.unifi.it/medusa. A standalone version of the software can be downloaded from https://github.com/combogenomics/medusa/releases. All results presented in this work have been obtained with MeDuSa v. 1.3.

Contact: marco.fondi@unifi.it

Supplementary information: Supplementary data are available at *Bioinformatics* online.

1 Introduction

The *de novo* assembly of short-read sequencing data usually leads to a fragmented set of genomic sequences (contigs). Ordering and orientating such contigs (scaffolding) represents the first, non-trivial step towards genome finishing and usually requires extensive processing and manual editing of large blocks of sequence (Barton and Barton, 2012).

The preferred approach to genome scaffolding is currently based on assembling the sequenced reads into contigs and then using paired-end information to join them into scaffolds. Most of the software based on such approach have several preparatory steps in which read and contig libraries are first converted to a specific format, then mapped against each other by means of an external aligner [e.g. BWA, (Li and Durbin, 2009) or BOWTIE, (Langmead *et al*., 2009)] and finally used to possibly join contigs together. At the end of this pipeline, a scaffolding graph is usually constructed and a plethora of different methods can be used to analyse the graph and produce the resulting scaffold structure. Currently available methods/software include SOPRA (Dayarian *et al*., 2010), SCARPA (Donmez and Brudno, 2013), MIP (Salmela *et al*., 2011), Opera (Gao *et al*., 2011), GRASS (Gritsenko *et al*., 2012) and SSPACE (Boetzer *et al*., 2011). A recent survey (Hunt *et al*., 2014) analyses and benchmarks most of these recent and sophisticated scaffolding software. The authors showed that, in general, they are not satisfying either in terms of usability or in terms of the quality of the solution, leading to the conclusion that there is still scope for improvements in this area.

An alternative approach for scaffolding genomes relies on the use of a complete (closed) reference genome to guide the ordering and the orientating of the contigs. Many available methods exist for mapping (and then scaffolding) the generated draft contigs (Galardini *et al*., 2011; Darling *et al*., 2010; Silva *et al*., 2013; van Hijum *et al*., 2005; Kolmogorov *et al*., 2014; Kim *et al*., 2013; Husemann and Stoye, 2010). This approach is also used in some specific contexts, such as for ancient DNA fragments reconstruction (Rajaraman *et al*., 2013), where read information is not available or reliable.

These software differ in terms of their overall strategy and implementation but, in general, (i) they allow for only a single reference genome (Galardini *et al*., 2011); (ii) when multiple genomes are allowed, generally these have to be closed and (iii) a reference phylogeny accounting for the evolutionary relationships among the selected taxa is to be provided to guide a multi-reference genomes-based scaffolding (Kolmogorov *et al*., 2014). None of the aforementioned approaches is capable of ignoring all of these constraints that, taken together, represent important practical limitations. Indeed, with the exception of model organisms, reliable closed reference genomes are not always available. Moreover, especially in the case of bacteria, genomic rearrangements among closely related organisms may introduce important structural differences, hampering the scaffolding procedure based on a single genome as reference. Finally, the requirement of a reliable phylogenetic reconstruction can pose a significant challenge, since it is not always straightforward for some bacterial taxa for which the large genetic variability in gene content inside a same species can lead to different phylogenies depending on which molecular marker and/or approach is used.

To overcome the difficulties that characterize currently available methods, we developed MEDUSA (Multi-Draft based Scaffolder), an algorithm for scaffolding draft genomes by ordering and orientating a set of *de novo* obtained contigs and thus speeding up genome finishing. Unlike most of the other software, MEDUSA: (i) formalizes the scaffolding problem by means of a combinatorial optimization formulation on graphs and implements an efficient constant factor approximation algorithm to solve it; (ii) allows for multiple reference genomes to be used during scaffolding; (iii) does not require prior knowledge on the evolutionary relationships (i.e. a phylogenetic tree) among the reference set of organisms and (iv) can handle both draft and complete reference genomes. This latter point is of great importance in practice since, in current public databases, the availability of draft genomes greatly exceeds that of completely sequenced ones (Reddy *et al*., 2014). Moreover, since retrieving the additional information needed by the aforementioned scaffolders can be a challenging task, an algorithm that does not rely on such prior knowledge is of great interest and allows the inclusion of a larger set of genomes for the scaffolding process. The strategy of MEDUSA is based on the intuition that a set of genomes related to the target one can be used for assigning a relative position to each contig, and that this kind of information is easily available in practice. Specifically, those contigs mapping on adjacent regions in these other genomes are considered to be neighbours in the resulting scaffold. MEDUSA formalizes such scaffolding problem as a path cover problem in a graph and solves it with *ad hoc* optimization techniques. The underlying algorithm has been implemented both in the form of a command line software and a web server.

Testing MEDUSA on bacterial and eukaryotic datasets revealed that our software performs very well in comparison to others currently available and answers some of the implicit requests pointed out by Hunt *et al*. (2014) in their review, i.e. usability and accuracy of the obtained results.

2 System and methods

2.1 Definitions and notation

A *contig* is a fragment of a source target genome. Let T be the *target genome* consisting of a set of n contigs $c_0, \ldots, c_{n-1}$ of various lengths.

An *ordering* of T corresponds to finding the true relative positions of the contigs c_i in the source sequence. The *orientation* of a contig indicates which strand of the source sequence it belongs to. We denote the reverse and complement of a contig c by $\overline{c}$. By duality, if c belongs to one strand, then $\overline{c}$ belongs to the other strand.

Consecutive oriented contigs in the ordering can be joined into a longer (gapped) supercontig called a *scaffold*. Informally, the scaffolding problem consists in inferring the order and orientation of the contigs in T. The (ideal) solution of the problem is one scaffold per chromosome of the source DNA sequence.

Therefore, if the source DNA sequence contains more chromosomes, then T contains contigs from more than one chromosome, and the solution consists in a set of scaffolds, i.e. in a partial ordering of the contigs of T. Moreover, due to possible errors in the assembly of the sequenced reads, even if the source DNA sequence contains one chromosome, the solution of the problem may be a set of scaffolds.

Consider in addition a collection $\mathrm{D} = \{D_0, \ldots, D_{k-1}\}$ of *comparison* genomes, where $D_0, \ldots, D_{k-1}$ are sets of contigs. Our algorithm is designed to determine a set of scaffolds on T and an orientation of its contigs by making use of the additional information provided by D.

Let T and D be given. We map the contigs of T on the contigs of D_h, for all D_h in D.

A contig $c_i \in T$ *hits* a contig $d \in D_h$ if c_i or its reverse and complement $\overline{c_i}$ aligns to d. We call *hit* the subsequence between the first and the last matching positions of c_i on d.

We use the software nucmer from MUMmer (Kurtz *et al.*, 2004) to align the contigs of T on the contigs of D_h, for all D_h in D and recover similar hits.

If c_i hits more than once the contigs of D_h, we call *best hit* of c_i on D_h the hit with maximum coverage, and we call *first position* of the best hit the minimum between the start and end coordinates of the best hit on the contig of D_h as assessed by MUMmer.

Let us denote the first position of the best hit of c_i on D_h by p_h^i.

We also define two variables $\texttt{forw}_h^i$ which is *true* if c_i maps forward and $\texttt{back}_h^i$ if it maps reverse (Obviously $\texttt{forw}_h^i = \neg\texttt{back}_h^i$).

Observe that the value p_h^i is defined if and only if the contig c_i hits D_h.

We make use of two kinds of information in different steps: the first one to determine an order of the contigs and the second to assign an orientation to each of them. Our method is composed of three main computational steps: graph construction, order determination and orientation assignment.

2.2 Graph construction

In the first step, we construct an undirected weighted graph $G = (V, E)$ as follows.

Let us associate a vertex to each contig, regardless of its orientation. Therefore, $V = \{v_1, \ldots, v_n\}$, and we assume that every vertex has associated an index (from 1 to n).

We list all the best hits for every contig of the target genome on each contig of any comparison genomes in increasing order of their first positions.

If the best hit of c_i and the best hit of c_j are in the same contig of D_h, then p_h^i and p_h^j are both defined, and they can be compared. In this case, if $p_h^i < p_h^j$, and there is no $l \in \{0, ..., k-1\}$ so that $p_h^i < p_h^l < p_h^j$, we say that c_i and c_j are *h-adjacent*. Let us define $A(c_i, c_j) = \{h : c_i \text{ is } h - \text{adjacent to } c_j\}$. There is an edge between v_i and v_j if $A(c_i, c_j) = \emptyset$, i.e. $E = \{(v_i, v_j) : \ c_i \text{ is } h\text{−adjacent to } c_j \text{ for some } h \in A(c_i, c_j)\}$.

The weight of an edge is given as $w(v_i, v_j) = |A(c_i, c_j)|$; since the cardinality of D is k, the weights range from 1 to k.

We call *Scaffolding Graph* the so obtained undirected weighted graph $G = (V, E)$.

2.3 Order determination

In the second step, the Scaffolding graph is used to find an order of the contigs. A *path* in G is a finite sequence of edges which connect a sequence of distinct vertices. A *path cover* P of G is a set of vertex-disjoint paths $P_1, \ldots, P_s$ that cover all the vertices of G. In a weighted graph, cover P has total weight $w(P) = \sum_{e \in P} w(e)$. From an optimization point of view, a cover can be characterized by two values: s (its cardinality) and w (its weight). The path cover having minimum cardinality in general does not coincide with the cover having maximum weight. This means that it is not possible to optimize both values at a same time. Since edges encode information about contiguity, and the weights support the existence of the edges, a natural choice is to find a path cover of maximum weight. We therefore formulate the problem as follows: given a scaffolding graph G, determine a maximum weight path cover of G. Unfortunately the problem is NP-complete since finding a Hamiltonian path can be seen as a sub-problem. We therefore opted for an approximation algorithm.

In Moran *et al.* (1990), three approximation algorithms are presented having a complexity-performance trade-off. We implemented the most efficient algorithm that gives an approximation of 1/2. The complexity of this method is $O(|E| \cdot \log |E|)$. The solution is unique if the weights of the edges are all different, but in general, more solutions are possible. This is due to the fact that the order in which the edges with same weight are processed influences the solution. MeDuSa uses a stable sorting for the edges of the graph to output the same solution at every run with same input.

Let us consider a path cover solution. The traversal of any path starting from one of its endpoints establishes an increasing total order of the contigs of T in the path. Without loss of generality, we start the traversal of any path from the endpoint vertex with lower index. By duality, starting from the greater index corresponds to traversing the path in the opposite direction, and thus to reading the contigs on the other strand, where the order and orientations of the contigs are reversed.

After the order assignment, the *cover* so constructed can be seen as a set P of directed paths.

2.4 Orientation assignment

In the third step, we take the orientation of the contigs into consideration.

Let us first consider any arc $\langle v_i, v_j \rangle$ in a path of P meaning that $v_i < v_j$ in the order. For every $h \in A(c_i, c_j)$, several relative orientations for c_i and c_j can occur. Our goal is to determine, relatively to $\langle v_i, v_j \rangle$, unique orientations for the two vertices. This is done by a majority rule, taking into account the fact that each relative orientation on one strand has a dual on the other one. More precisely, let us define the following quantities:

$$\begin{aligned}\mathrm{FF}(i,j) = \quad & |\{h : (p_h^i < p_h^j \wedge (\texttt{forw}_h^i \wedge \texttt{forw}_h^j)) \vee \\ & (p_h^i > p_h^j \wedge (\texttt{back}_h^i \wedge \texttt{back}_h^j))\}|\end{aligned}$$

$$\begin{aligned}\mathrm{FB}(i,j) = \quad & |\{h : (p_h^i < p_h^j \wedge (\texttt{forw}_h^i \wedge \texttt{back}_h^j)) \vee \\ & (p_h^i > p_h^j \wedge (\texttt{back}_h^i \wedge \texttt{forw}_h^j))\}|\end{aligned}$$

$$\begin{aligned}\mathrm{BF}(i,j) = \quad & |\{h : (p_h^i < p_h^j \wedge (\texttt{back}_h^i \wedge \texttt{forw}_h^j)) \vee \\ & (p_h^i > p_h^j \wedge (\texttt{forw}_h^i \wedge \texttt{back}_h^j))\}|\end{aligned}$$

$$\begin{aligned}\mathrm{BB}(i,j) = \quad & |\{h : (p_h^i < p_h^j \wedge (\texttt{back}_h^i \wedge \texttt{back}_h^j)) \vee \\ & (p_h^i > p_h^j \wedge (\texttt{forw}_h^i \wedge \texttt{forw}_h^j))\}|\end{aligned}$$

For the sake of simplicity, we assume that these four values are all distinct (in practice, this is almost always the case). However, it is easy to extend the following procedure to the case in which this assumption is not true (see Supplementary File S1). We denote the orientation of c_i (respectively, c_j) relative to the arc $\langle v_i, v_j \rangle$ by *tail*(v_i, v_j) (respectively, *head*(v_i, v_j)). We then have that *tail*(v_i, v_j) is *forward* if $\max\{\mathrm{FF}(i,j), \mathrm{FB}(i,j), \mathrm{BF}(i,j), \mathrm{BB}(i,j)\} \in \{\mathrm{FF}(i,j), \mathrm{FB}(i,j)\}$ and it is *backward* otherwise. Analogously *head*(v_i, v_j) is then *forward* if $\max\{\mathrm{FF}(i,j), \mathrm{FB}(i,j), \mathrm{BF}(i,j), \mathrm{BB}(i,j)\} \in \{\mathrm{FF}(i,j), \mathrm{BF}(i,j)\}$ and it is *backward* otherwise. Consider now two consecutive arcs $\langle v_1, v_2 \rangle$ and $\langle v_2, v_3 \rangle$ in a path. We say that the orientation assignment of c_2 is *consistent* if and only if *head*$(v_1, v_2) =$ *tail*(v_2, v_3), that is, if the two arcs propose a *consistent* orientation for c_2. The orientation for the contigs of T in a same scaffold is given by consistent orientation assignments. In detail, we start to analyse any path in P. We initialize an empty scaffold. Then, if the orientation assignment for any two consecutive arcs is consistent, we add the contigs corresponding to the arcs in the scaffold with the orientation suggested; otherwise, if it is not consistent, we add the vertices of the first arc to the scaffold, then we cut the second arc, and start

Table 1. Microbial datasets used for benchmarking

Dataset name	Organism	No. of replicons	No. of contigs (Mb)	Reads	No. of drafts	Genome length (Mb)	GC%	N50	Assembly coverage (%)
BCEN	*B.cenocepacia* j2315	4	1223(7.97)	In-house Illumina HiSeq	4	8.05	65.91[a]	7619	87.8
RSPH	*R.sphaeroides* 2.4.1	7	564(4.38)	SRR522246	2	4.60	67.4[a]	11 552	95
ECOL	*E.coli* K12	1	451(4.4)	SRR001665 + SRR001666	25	4.64	50.79	15 570	95.9
MTUB	*M.tuberculosis*	1	116(4.38)	In-house Illumina HiSeq	13	4.41	65.61	67 226	98.4
SAUR	*S.aureus*	3	170(2.8)	GAGE Short jump library	35	2.9	32[a]	47 016	96.5

[a]Average over all replicons

to traverse a new path. We refer the reader to the Supplementary File S1 for examples on the orientation assignment. Here we point out that in the case in which the maximum among $FF(i,j), FB(i,j), BF(i,j), BB(i,j)$ is not unique, we can have a multiple assignment for the orientation of v_i or v_j. In our method, both orientations are considered, thereby reducing possible inconsistency in the traversal of the path for the orientation assignment. This step can be easily carried out, since we treat separately the orientation and the ordering. The complexity of the entire procedure is linear in the number of vertices.

2.4.1 MeDuSa output

The algorithm then produces a scaffold by merging the oriented contig sequences, using 100 undetermined bases (*N*) as a spacer. Accordingly, the final output of MeDuSa is a FASTA file, where each sequence represents a scaffold of ordered and oriented contigs separated by stretches of 100 'N's. Alternatively, MeDuSa can infer the distance among the joined contigs based on their distance on the comparison genomes set (see Supplementary File S1 for details on this step). The user can choose between these two options when launching MeDuSa.

3 Results and discussion

In this section, we present the results we obtained when applying our software to benchmarks (Table 1). Two of them were retrieved from the SRA archive database (ECOL, RSPH), the SAUR dataset was obtained from the GAGE benchmark study (Salzberg *et al.*, 2012), whereas the BCEN and MTUB datasets were obtained from *in-house* performed Illumina HiSeq sequencing runs with a fragment size of 500 bp. To assess the reliability of the *in-house* datasets, a quality check was performed using the FastQC suite (available at www.bioinformatics.babraham.ac.uk/projects/fastqc) and calculating the proportion of reads correctly mapping to the assembled contigs. The results obtained (reported in Supplementary File S1) revealed no major issues for these sequencing runs. Information on the main features of the publicly available GAGE reads dataset can be found in Salzberg *et al.* (2012).

More in detail, we first analysed how MeDuSa performs on real genome scale datasets in terms of errors, completeness and number of reconstructed scaffolds, and how the choice of the draft genomes used for scaffolding influences the results. Then, we compared the performance of MeDuSa with those of five other scaffolders. To evaluate the reliability of the solutions generated by our algorithm, we have chosen real bacterial datasets for which (at least) one whole genome had already been completed, that is 'closed', and used this as a positive reference. From now on we will refer to the following metrics to evaluate the results of our tests: (i) number of correct joins, i.e. the number of true positives (those joins correctly predicted according to the comparison with the corresponding reference genome); (ii) accuracy, the number of true positives divided by the number of proposed joins (i.e. all the joins in the computed solution). For the sake of completeness, it should be mentioned that, unlike the score introduced by Hunt *et al.* (2014), our accuracy index does not include the estimation of the distance among contigs. (iii) recovered information, the number of true positives divided by the expected number of joins; (iv) overall number of reconstructed scaffolds; (v) N50 and NG50 metrics and (vi) total length of joined fragments. Observe that the expected joins correspond to the number of contigs minus the number of chromosomes. Moreover a join between two contigs is considered correct if and only if (i) the contigs are directly consecutive in the genome (no other contig appears in between and they belong to the same replicon) and (ii) the orientation of the two fragments is correct.

3.1 Genome-scale datasets

MeDuSa was tested on datasets of genomes from five microbial representatives (Table 1), each of which is composed as follows:

- a target genome (the draft genome to be scaffolded).
- a set of draft genomes from (more or less) closely related strains (named comparison genomes) to be used in the scaffolding pipeline of MeDuSa.

Except for the SAUR dataset for which we used the contigs from the benchmark work of Hunt *et al.* (2014), for each of the tested datasets, the target genome was obtained from the sequencing reads using ABySS V. 1.3.7 (Simpson *et al.*, 2009). Several *k*-mer values were tried for each dataset. The one leading to the best assembly [as described in Fondi *et al.* (2014)] was chosen and used as input for MeDuSa afterwards. Although genome assembly information is not necessary to use our method, we preferred building the target genome from reads to use exactly the same instance as input for the other programs during benchmarking (see Section 3.4).

The results of these tests are summarized in Table 2. The general goal of reducing the fragmentation of the set of contigs is achieved well. The number of fragments obtained after MeDuSa is applied is significantly smaller than the initial number of contigs. Also, in most cases, the majority of the scaffolds is composed of more than one original contig, that is, is multi-contig. Remarkably, in the case of the MTUB dataset, for which a complete genome (that of *Mycobacterium tuberculosis* KZNV2475) was available among the comparison ones, the result is a single scaffold with an overall length close to the one of the input draft. It is to be noticed that, when referring to multi-contig scaffolds, we are referring to scaffolds generated by joining two or more original contigs together, without taking into consideration the possible presence of internal breakpoints (usually represented by 'Ns') inside a contig. In other words, the possible gaps in the input contigs (as a result of sequencing ambiguities and/or placed by the *de novo* assembler) have not been considered in the calculation of the number of multi-contig scaffolds. Finally, to

Table 2. Accuracy and completeness statistics on the five microbial datasets

Dataset	Contigs	Scaffolds (multi-contig)	Proposed joins	No. of correct joins(accuracy)	No. of wrong joins (inverted contigs)	Recov. info	**Sequence in scaf folds (bp)**	**Overall length (bp)**	**N50**
BCEN	1223	31(25)	1192	1142(96%)	50(11)	94%	7 024 733	7 179 921	756 745
RSPH	564	81(46)	483	389(81%)	94(10)	70%	4 196 732	4 430 833	146 850
ECOL	451	9(6)	442	321(72%)	121(39)	71%	4 442 534	4 486 586	2 406 641
MTUB	116	1	115	105 (91%)	10(3)	91%	4 338 452	4 349 052	4 349 052
SAUR	170	16(7)	154	104(68%)	50(1)	62%	2 830 037	2 864 402	990 064

Table 3. Influence of phylogenetic distance between target and comparison drafts for the target *E.coli*

Target: *E. coli* Genus of organisms for comparison	No. of Scaffolds	No. of joins	No. of correct joins	No. of wrong joins	Recovered info (%)
E. coli	9	442	321	122	71
Escherichia (other species)	46	405	312	93	70
Shigella	32	419	307	112	68
Vibrio	439	12	2	10	0,4
Pseudomonas	441	10	1	9	0.2
Acinetobacter	-	0	0	0	0
E. coli + *Escherichia* other species + *Shigella*	12	439	323	116	71
Escherichia + *Shigella*	30	421	316	105	70
All	12	439	323	116	71

Table 4. Influence of phylogenetic distance between target and comparison drafts for the target *M.tuberculosis*

Target: *M. tuberculosis* Genus of organisms for comparison	No. of scaffolds	No. of joins	No. of correct joins	No. of wrong joins	Recovered info (%)
M.tuberculosis	1	115	105	10	91
Mycobacterium (other species)	53	63	29	34	29
Streptomyces	95	21	15	20	0
Lactobacillus	-	0	0	0	0
M. tubercolosis + *Mycobacterium* other species	1	115	105	10	91
All	1	115	105	10	91

further test the robustness of our approach, we also evaluated (i) the influence on the overall scaffolding procedure of contigs with multiple hits among the comparison set, (ii) the reliability of the estimated gap lengths, (iii) the possibility to integrate sequence similarity information between target and comparison contigs when giving a weight to the edges and (iv) the performances of our approach on two eukaryotic datasets, namely *Saccharomyces cerevisiae* S288c and *Drosophila melanogaster*. The results of this analysis are reported in Supplementary File S1 (Section 7–10) and revealed that MeDuSa is capable of producing good results even when trying to scaffold large and complex genomes.

3.2 Influence of the taxonomical distance

The choice of a set of comparison genomes is left to the user and depends mostly on the organism under study. Nevertheless, some guidelines can be extracted from experimental analyses on the present datasets. The results displayed in Tables 3–7 clarify how the phylogenetic distance between target and comparison genomes influences the scaffolding procedure. We repeated the same test for five different target genomes: *Escherichia coli* (Table 3), *M. tuberculosis* (Table 4), *Staphylococcus aureus* (Table 5), *Burkholderia cenocepacia* (Table 6) and *Rhodobacter sphaeroides* (Table 7)

For each target genome, we have created a series of different sets of comparison draft genomes, in increasing order of phylogenetic distance from the target (from stains belonging to the same species up to strains belonging to unrelated genera). After that, some sets of comparison genomes are created by merging the different groups. These results are interesting for many reasons. First, as expected, the information provided by the comparison drafts tends to decrease with the increase of the distance, and becomes totally insufficient after a certain taxonomical distance (roughly the genus level). The solutions at this level become very poor (the number of scaffolds is close to the initial number of contigs). On one hand, this means that the comparison drafts should be chosen as close as possible to the target. In microbial genomics, this is usually not a problem because some more or less closely related draft genomes are likely to be present for (virtually) each newly sequenced genome. On the other hand, this phenomenon means that the method is robust to noise (false positives are very rare). This aspect is confirmed by the second

Table 5. Influence of phylogenetic distance between target and comparison drafts for the target *S. aureus*

Target: *S. aureus* Genus of organisms for comparison	No. of scaffolds	No. of joins	No. of correct joins	No. of wrong joins	Recovered info (%)
S. aureus	16	154	104	50	62
Staphylococcus (other species)	97	73	29	44	17
Clostridium	0	0	0	0	0
Lactobacillus	169	1	0	1	0
S. aureus and *Staphylococcus* other species	13	157	103	54	60
All	13	157	103	54	60

Table 6. Influence of phylogenetic distance between target and comparison drafts for the target *B. cenocepacia*

Target: *B. cenocepacia* Genus of organisms for comparison	No. of Scaffolds	No. of joins	No. of correct joins	No. of wrong joins	Recovered info (%)
B. cenocepacia	31	1192	1142	50	94
Burkholderia (other species)	330	893	608	285	50
Ralstonia	928	295	91	204	10
Neisseria	1220	3	0	3	0
B. cenocepacia and *Burkholderia* other species	89	1134	1048	86	85,6
All	105	1118	1030	88	84,4

Table 7. Influence of phylogenetic distance between target and comparison drafts for the target *R. sphaeroides*

Target: *R. sphaeroides* Genus of organisms for comparison	No. of scaffolds	No. of joins	No. of correct joins	No. of wrong joins	Recovered info (%)
R. sphaeroides	81	483	389	94	70
Rhodobacter (other species)	392	172	53	119	9
Sinorhizobium	500	64	4	60	0,7
Rickettsia	564	0	0	0	0
R. sphaeroides and *Rhodobacter* other species	78	486	397	89	70
All	109	455	356	99	63

set of experiments, where the set of comparison drafts does not belong to a single taxonomic group. Notably, the quality of the scaffolds obtained by adding the whole set of comparison genomes (distant and closer ones) is usually not too far from the one obtained with the closest possible genomes. This means that the user could potentially add many draft genomes, without the risk of introducing much misleading noise. Of course, in the case where the comparison drafts are numerous, the required contig mapping phase (nucmer in our implementation) will sensibly increase the algorithm running time.

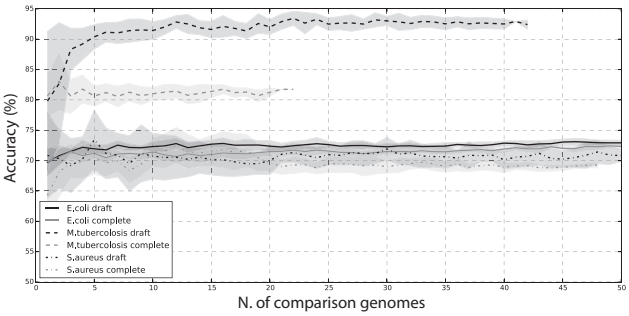

Fig. 1. Variation of accuracy relative to the number of comparison genomes. The grey shade along the lines represents the 95% confidence interval across all the performed permutations

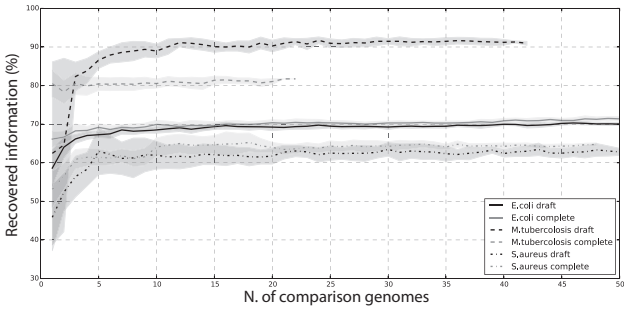

Fig. 2. Recovered information with respect to the number of comparison genomes. The grey shade along the lines represents the 95% confidence interval across all performed permutations

3.3 Varying the number of draft and complete comparison genomes

The second parameter in the choice of the comparison dataset is the number of reference genomes to use. This aspect has been investigated using the ECOL, MTUB and SAUR datasets. This choice relies on the fact that, for these organisms, a considerable number of complete genomes are available to perform the tests described later. All the available draft and complete genomes from representatives of the *M.tuberculosis* species were retrieved from the NCBI database, in addition to 50 draft and complete genomes from representatives of the *E.coli* and *S.aureus* species. A number of different instances equal to the number of the retrieved genomes for each dataset (N) were built, with an increasing number of comparison genomes (from 1 to N) used during each test. This increase was performed consistently only adding new drafts to the previous set. Since the choice of the order in which the drafts are added could influence the solution, all the tests were repeated 10 times, each time varying the relative order of the comparison genomes. Moreover, since MeDuSa allows mixing closed and draft genomes in the comparison set, we tested how the presence of closed genomes affected the behaviour of the algorithm. To do this, another set of tests was performed using closed genomes instead of drafts in the comparison set. For each dataset, the following values are presented: accuracy (Fig. 1), recovered information (Fig. 2) and number of scaffolds (Fig. 3).

The results obtained showed a similar trend in all the datasets and for each of the metrics computed. After an initial improvement, the performances (in terms of accuracy, recovered information and number of scaffolds) stabilize with respect to the increase of the number of comparison genomes included in the dataset.

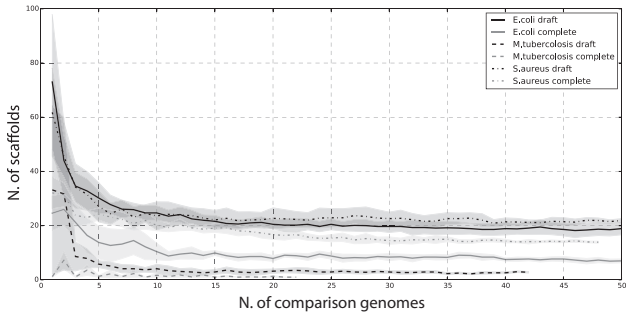

Fig. 3. Number of scaffolds in relation to the number of (draft or complete) comparison genomes used. The grey shade along the lines represents the 95% confidence interval across all the performed permutations

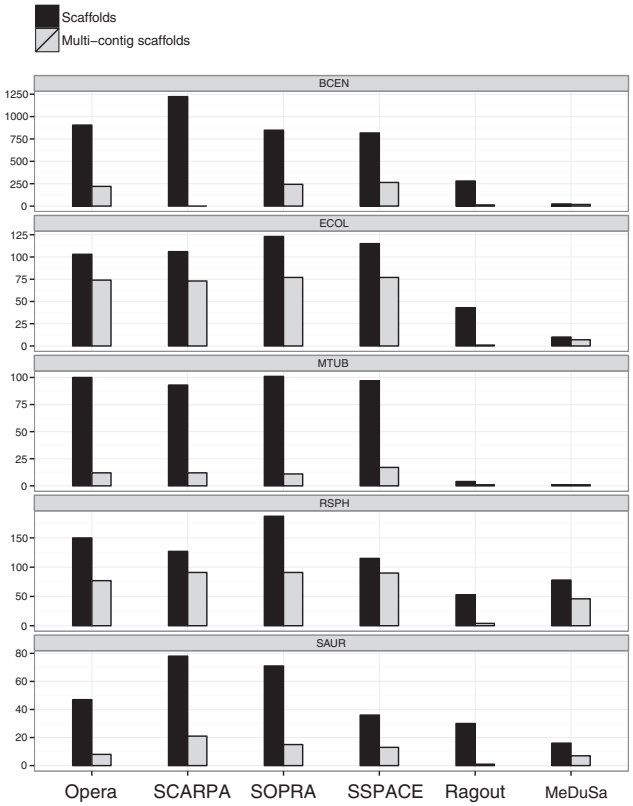

Fig. 4. Comparison between the performances of MeDuSa and those from other selected scaffolders in terms of number of generated scaffolds and multi-contig scaffolds. Observe that no scaffold was generated on the BCEN and SAUR dataset by SCARPA

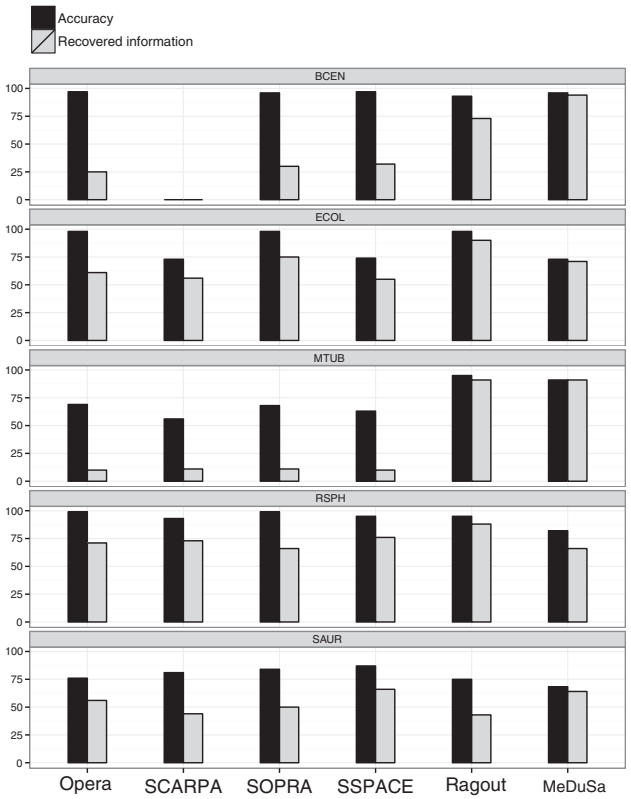

Fig. 5. Accuracy and recovered information comparison among the benchmarked tools and MeDuSa. Observe that no scaffold was generated on the BCEN and SAUR dataset by SCARPA

These results suggest that our method is sufficiently robust to noise created by redundant information. Also, the small number of false positives included in the final solutions is confirmed by the extreme stability of the accuracy level, shown in Figure 1. These considerations are true whether closed or draft genomes are used as the comparison set. With the only exception of the MTUB dataset, the use of closed genomes gives more information and the completeness of the solution is higher. On the other hand, the accuracy in this case is slightly lower. This can be explained by at least two lines of evidence. From a biological viewpoint, complete genomes may embed structural variations (e.g. duplicated and/or inverted regions) that, due to *de novo* assembly issues, might not be observed in their fragmented draft counterparts. These biological features, in turn, may hinder the scaffolds reconstruction and possibly lead to wrong joins.

Moreover, from an informational viewpoint, including complete genomes in the comparison dataset may lead to an increased number of predicted joins and, consequently, to a higher false-positive rate. The choice of the comparison genomes is crucial for evaluating the performance of MeDuSa since the choice of strains phylogenetically related to the target genome is less likely to produce a poor quality graph, eventually leading to better scaffolds and a low number of true positives. However, relatively recent evolutionary events, such as lineage-specific genomic rearrangements, could affect the results of MeDuSa with false-positive adjacencies even when the genomes from strains of the same species of the target are chosen as reference. In such cases, we recommend to use a weighting scheme based on sequence similarity that has been shown to perform better in specific cases (see Supplementary File S1).

3.4 Benchmarking

The performance of MeDuSa was compared with those of five other programs, namely SOPRA V. 1.4.6 (Dayarian *et al.*, 2010), SCARPA V. 0.241 (Donmez and Brudno, 2013), Opera V. 1.4 (Gao *et al.*, 2011), SSPACE V. 3.0 (Boetzer *et al.*, 2011), Ragout V. 1.0 (Kolmogorov *et al.*, 2014). The first four of these scaffolders are paired ends-based and the choice to use these specific ones was based on both their performances and their usability as assessed by Hunt *et al.* (2014). The choice to use the recently developed Ragout

relies on the fact that it implements an overall strategy that resembles that of MeDuSa, although requiring more input information (phylogenetic tree of the analysed genomes). Options and parameters (e.g. the choice of reads mapper) for each of the paired ends-based methods were selected among those leading to the best performances on genome-scale data as reported in Hunt *et al.* (2014) (see Supplementary File S1). Each paired ends-based software was used both on trimmed (using DynamicTrimming from the SolexaQA package (Cox *et al.*, 2010) and Phred 30 as the quality threshold) and untrimmed reads datasets. Indeed reads trimming is usually performed after a sequencing run to remove poor quality bases although, in some cases, it may lead to a loss of information during scaffolding. We here report the values for the option—trimmed or untrimmed—leading to the best results. With the exception of insert length (that was set to its appropriate value for each dataset), all the other parameters used are reported in Supplementary File S1. As for Ragout, the reconstruction of the reference phylogenetic tree was performed using Oma (Roth *et al.*, 2008) with default parameters. Importantly, all the results obtained during this benchmarking were double-checked with those obtained using the scripts provided by Hunt *et al.* (2014) in their survey. No major differences were observed concerning the performances of the mate pairs-based scaffolders on the selected genome datasets.

As indicated by the results of these tests (reported in Fig. 4), the number of scaffolds produced by our algorithm is lower than that produced by all the other four paired end-based scaffolders in all the performed tests. Notably, Ragout and MeDuSa produce similar results on each dataset, with the latter leading to a lower number of scaffolds in the BCEN, ECOL and SAUR datasets and both of them leading to a single scaffold with the MTUB dataset. What is particularly interesting is also the high percentage of multi-contig scaffolds over the total number of scaffolds reconstructed by MeDuSa (75%, on average), a crucial aspect since minimization of the number of scaffolds is clearly the final goal of any scaffolding method. As expected, the analysis of the N50 metrics revealed that MeDuSa outperforms all the other paired ends-based scaffolders and produces results that are, in most cases, similar to Ragout (see Supplementary File S1). Additionally, in Figure 5, we report accuracy and recovered information for the software tested herein and for MeDuSa. This comparison revealed that our algorithm produces results that overlap (and, in some cases, outperform) those from other currently available programs, even in terms of reliability of the proposed solution.

In conclusion, both the high percentage of true joins recovered and the low percentage of errors observed make MeDuSa very competitive with the other scaffolders in general, including those exploiting a similar strategy (i.e. Ragout). It is to be noticed, however, that MeDuSa requires far less information in respect to the aforementioned methods and this greatly increases its usability. Also, MeDuSa performs very well in respect to all the other benchmarked software in terms of required running time. Indeed, all the paired end-based tools generally have long running time due to their reprocessing and read mapping stages and, on our datasets, most of them were unable to complete the scaffolding in <2 h. Despite the fact that Ragout processes input files quite quickly (23, 4, 16 and 90 min for the MTUB, RSPH, BCEN and ECOL datasets, respectively), it requires two operations that can be quite time consuming when dealing with a high number of genomes, i.e. computation of orthologous groups of sequences and phylogenetic tree reconstruction. The same datasets were scaffolded by MeDuSa in <10 min, on average.

4 Conclusion and perspectives

Draft genome scaffolding is a key step in the finishing stages of microbial genomic pipelines. In this article, we presented MeDuSa, a novel graph theory-based algorithm for scaffolding draft genomes by ordering and orientating their contigs. Unlike traditional software, it does not rely either on paired-end information of sequencing reads or on a phylogenetic distance of the microorganisms used in the analysis. This sensibly increases the usability of our software and, at the same time, reduces the computational time required for genome scaffolding.

Using real microbial and eukaryotic datasets, we show that the algorithm implemented in MeDuSa is capable of significantly reducing the fragmentation of draft genomes and, in most cases, of producing less and longer scaffolds in comparison to commonly used scaffolders, while maintaining comparable accuracy and correctness of the predicted joins.

Funding

B.D. is a recipient of a grant from the European Research Council under the European Community's Seventh Framework Programme (FP7/2007-2013) / ERC grant agreement [247073]10 SISYPHE. The work described in this publication was financially supported by two PNRA (Piano Nazionale per la Ricerca in Antartide) grants (PNRA 2013/B4.02 and PNRA 2013/AZ1.04). P.C. was supported in part by the Italian Ministry of Education, University, and Research under PRIN 2012C4E3KT national research project.

Conflict of Interest: none declared.

References

Barton,M.D. and Barton,H.A. (2012) Scaffolder - software for manual genome scaffolding. *Source Code Biol. Med.*, **7**, 4.

Boetzer,M. *et al.* (2011) Scaffolding pre-assembled contigs using sspace. *Bioinformatics*, **27**, 578–579.

Cox,M.P. *et al.* (2010) Solexaqa: at-a-glance quality assessment of illumina second-generation sequencing data. *BMC Bioinformatics*, **11**, 485.

Darling,A.E. *et al.* (2010) Progressivemauve: multiple genome alignment with gene gain, loss and rearrangement. *PLoS One*, **5**, 1754–1760.

Dayarian,A. *et al.* (2010) Sopra: Scaffolding algorithm for paired reads via statistical optimization. *BMC Bioinformatics*, **11**, 345.

Donmez,N. and Brudno,M. (2013) Scarpa: scaffolding reads with practical algorithms. *Bioinformatics*, **29**, 428–434.

Fondi,M. *et al.* (2014) Draft genomes of three antarctic psychrobacter strains producing antimicrobial compounds against burkholderia cepacia complex, opportunistic human pathogens. *Mar. Genomics*, **13**, 37–38.

Galardini,M. *et al.* (2011) Contiguator: a bacterial genomes finishing tool for structural insights on draft genomes. *Source Code Biol. Med.*, **6**, 11.

Gao,S. *et al.* (2011) Opera: reconstructing optimal genomic scaffolds with high-throughput paired-end sequences. *J. Comput. Biol.*, **18**, 1681–1691.

Gritsenko,A.A. *et al.* (2012) Grass: a generic algorithm for scaffolding next-generation sequencing assemblies. *Bioinformatics*, **28**, 1429–1437.

Hunt,M. *et al.* (2014) A comprehensive evaluation of assembly scaffolding tools. *Genome Biol.*, **15**, R42.

Husemann,P. and Stoye,J. (2010) Phylogenetic comparative assembly. *Algorithms Mol. Biol.*, **5**, 3.

Kim,J. *et al.* (2013) Reference-assisted chromosome assembly. *Proc. Natl. Acad. Sci. USA*, **110**, 1785–1790.

Kolmogorov,M. *et al.* (2014) Ragout-a reference-assisted assembly tool for bacterial genomes. *Bioinformatics*, **30**, i302–i309.

Kurtz,S. *et al.* (2004) Versatile and open software for comparing large genomes. *Genome Biol.*, **5**, R12.

Langmead,B. *et al.* (2009) Ultrafast and memory-efficient alignment of short DNA sequences to the human genome. *Genome Biol.*, **10**, R25.

Li,H. and Durbin,R. (2009) Fast and accurate short read alignment with burrows-wheeler transform. *Bioinformatics*, **25**, 1754–1760.

Moran,S. *et al*. (1990) Approximation algorithms for covering a graph by vertex-disjoint paths of maximum total weight. *Networks*, **20**, 55–64.

Rajaraman,A. *et al*. (2013) Fpsac: fast phylogenetic scaffolding of ancient contigs. *Bioinformatics*, **29**, 2987–2994.

Reddy,T.B. *et al*. (2014) The genomes online database (gold) v.5: a metadata management system based on a four level (meta) genome project classification. *Nucleic Acids Res*, **43**, D1099–D1106.

Roth,A.C. *et al*. (2008) Algorithm of OMA for large-scale orthology inference. *BMC Bioinformatics*, **9**, 518.

Salmela,L. *et al*. (2011) Fast scaffolding with small independent mixed integer programs. *Bioinformatics*, **27**, 3259–3265.

Salzberg,S.L. *et al*. (2012) Gage: a critical evaluation of genome assemblies and assembly algorithms. *Genome Res.*, **22**, 557–567.

Silva,G.G. *et al*. (2013) Combining *de novo* and reference-guided assembly with scaffold_builder. *Source Code Biol. Med.*, **8**, 23.

Simpson,J.T. *et al*. (2009) Abyss: a parallel assembler for short read sequence data. *Genome Res.*, **19**, 1117–1123.

van Hijum,S.A. *et al*. (2005) Projector 2: contig mapping for efficient gap-closure of prokaryotic genome sequence assemblies. *Nucleic Acids Res.*, **33**(Web Server issue), W560–W566.