Inverted Repeats Scaffolding for a Dedicated Chloroplast Genome Assembler

Victor Epain, Dominique Lavenier, Rumen Andonov

Inverted Repeats Scaffolding for a Dedicated Chloroplast Genome Assembler

Victor Epain
Univ. Rennes, IRISA/CNRS, Inria, F-35000 Rennes, France

Rumen Andonov
Univ. Rennes, IRISA/CNRS, Inria, F-35000 Rennes, France

Dominique Lavenier
Univ. Rennes, IRISA/CNRS, Inria, F-35000 Rennes, France

Abstract

This paper describes a novel assembly approach for chloroplast genomes. It contains two modular steps. In the first step, based on the hypothesis that chloroplasts genomes are over-represented compared to the nuclear genome in the plant's cell, we assemble contigs with a De Bruijn graph based approach using short reads with a high k-mer coverage. Connections between oriented contigs are also provided here.

The second step determines the order and the orientation of the contigs (scaffolding). Taking advantage of the knowledge that chloroplast genomes posses well studied circular structure, we develop a particular formulation of the scaffolding problem, called *Nested Inverted Fragments Scaffolding*. It aims to assemble highly conserved inverted repeats. We formulate it as an optimisation problem and we prove that it is NP-Complete. To solve the problem we propose and implement an integer linear programming formulation.

We evaluate our method on a set of real instances (a benchmark of 42 chloroplast genomes) and show that it obtains notable achievements with respect to the quality of the results. To further estimate the performance of our scaffolding module, we test it on huge artificially created instances. The results demonstrate an excellent behaviour of our integer formulation as even very large instances have been solved at the first Branch & Bounds node.

2012 ACM Subject Classification Applied computing

Keywords and phrases Genome assembly, Inverted repeats, Contig, Scaffolding, De Bruijn graph, Assembly graph, Elementary longest path problem, Integer programming, NP-Complete problem

Digital Object Identifier 10.4230/LIPIcs...

Acknowledgements The authors are thankful to Abdelkader Ainouche for numerous discussions and for providing real plants and chloroplasts data. They are furthermore thankful to Sven Schrinner and Gunnar Klau for the valuable discussions about the problem's complexity.

1 Introduction

Sequencing an organism consists in extracting DNA molecules contained in its cells and getting them as sequences of ATGC letters corresponding to nucleotides. Modern sequencing technologies are still unable to return one complete sequence for each DNA molecule but instead yield fragmented substrings of them called reads that need to be assembled to reconstruct the original genome. A major difficulty for the assembly is provoked by genomes repeats: a read from one region can overlap another read containing a repeat of the first one, while both reads are sequenced from distanced genomic locations. Moreover, reads are sequenced without distinction from the two complemented DNA strands. Thus, *Inverted repeats* (a DNA sequence is the reverse-complement of another one) can produce repeat-induced overlaps and can wrongly mix regions from both DNA strands.

This paper presents an approach for assembling the chloroplast genome, a plant organelle responsible for photosynthesis metabolism, which confers the ability to transform sunlight energy to chemical energy. The knowledge of chloroplast genomes allows evolutionary analyses [17], (meta)-barcoding [4], and is useful for biophotovoltaic process development [15].

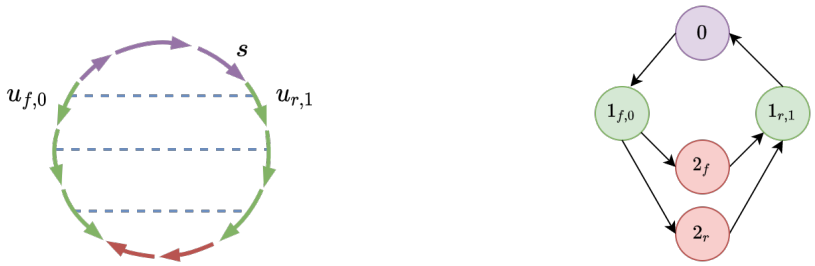
Several specialised assemblers for chloroplast genomes have been already developed in [6, 12] and have been compared in [8]. However, none of them explicitly formulates the specific subproblems that need to be solved. As a consequence, they apply essentially (meta)heuristics.

The chloroplasts possess a circular genome which is characterized by very specific but well-described repeated regions [2]. An instance of such a structure is given in Subfigure 1a. Deciphering these particular genome repeats is the first requirement for a successful chloroplast assembler. One more specific issue here is the separation of the plant reads from the chloroplast ones. Indeed, splitting them is not an easy task as there can be overlaps between both groups. As consequence, choosing repeat-induced or plant-chloroplast overlaps can lead to a misassembled genome.

Our approach can be cut in two main parts. The first part generates *contigs* that are expected to be part of the chloroplast. A *multiplicity* is attached to each contig. It provides an upper bound of the number of copies per contig. A list of potential *links* between contigs is also generated (*c.f.* Table 1 for an instance of input data: set of contigs and set of links). Finally, one contig is tagged as *seed contig* using proteome information from a close plant species. It is supposed to contain one reference gene present in a single copy.

In the second stage called *scaffolding* we finish the assembly, *i.e.* we order and orient the contigs yielded by the first step. The particularity of our approach is that it is based exclusively on the knowledge that most chloroplast genomes are circular and has two inverted repeats [2] (see for illustration Subfigure 1a). In contrast to some of the previous chloroplast genome scaffolders [1], we need no additional distance-constraints encoding the insert-size information.

This step is modular and completely independent from the first one. Any assembly graph obtained by another assembler can serve here as an input data. The data are firstly represented in a structure that we call *Multiplied Doubled Contigs Graph* (MDCG). We search in this graph to finish the assembly by reconstructing inverted repeats using the observation that they represent nested couples of inverted regions. Thanks to a mathematical specification on the inverted repeats structure, we maximally reduce the assembly graph. Thus, we model the chloroplast scaffolding as building a sequence of oriented contigs, starting from and ending with the seed contig, which maximizes the number of nested and contiguous *inverted fragments*. We call the underlying optimisation problem *Nested Inverted Fragments Scaffolding Problem* (NIFSP) and prove that it is NP-Complete. We propose an Integer Linear Programming (ILP) formulation and show that the number of variables and constraints is polynomial in respect to the vertices and the edges number. Although our formulation is somehow similar to the one proposed in [11] for RNA folding, here we deal with contigs instead of nucleotide sequences, without any knowledge of their order, which obviously increases the underlying challenges. We use the Gurobi software [10] to solve the optimization problem, resulting in a small number of larger contigs that correspond to chloroplast's genomic regions. Finally, our tool yields a scaffolding enriched assembly graph, as illustrated Subfigure 1b. We perform numerical computations and evaluate the quality of the results using Quast evaluation tool [9].



(a) Chloroplast genome's structure **(b)** Scaffolding final enriched assembly graph

Figure 1 Genomic knowledge and corresponding perfect assembly result. **(a)** Each arrow represents an oriented DNA sequence (contigs). Chloroplast's genome is circular thus it is a circular sequence of contigs, that begins at s and ends in s. Purple and red regions are unique, while green ones are *inverted repeats* (one is obtained by reversing and complementing the sequence of the other). For the contig indicated as $u_{f,0}$ on the left-side green region, the contig in front of it (on the right-side green region) has the reverse orientation (so r). Because orientations are mutually exclusive, it is necessary to chose an occurrence that differs 0 (so 1). The couple $u_{f,0}, u_{r,1}$ is defined as inverted fragments. Each inverted fragments couple is linked by blue dashed line. Inverted repeats can be modelled as sequences of nested inverted fragments. **(b)** Vertices correspond to genomic regions while edges indicate their neighbourhoods. Purple, green and red vertices correlate respectively with purple, green and red regions in Subfigure 1a. Two green vertices' label differ by their orientation 'f' and 'r' (one is the reverse complement sequence of the other). They also differ by their occurrence number (0 and 1). It simply means that their associated genomic regions are simultaneously present in the genome sequence (*i.e.* the green vertices both participate in the solution). On the opposite, red vertices just differ by their orientation: it implies that only one genomic region from one of these two vertices appears in the genome (one genomic region is the reverse complement of the other).

2 Material & Method

2.1 Material & Method Overview

Input data are reads that come from short reads sequencing technology, and can be part of plant's or/and chloroplasts' genomes. We also use chloroplast proteome data from a near plant species, and a well known gene sequence that is contained with a very high probability in one of the chloroplast's unique genomic regions, defined as the *seed gene*. An overview of the method is illustrated in Figure 2. In the next section we briefly describe how chloroplast contigs with their links are extracted.

2.2 Data Filtering & Contigs and Links Generation

Our strategy supposes that the chloroplast genome is over-represented compared to the nuclear genome of the plant. We first perform a de Bruijn graph (DBG) assembly with Minia [3] from a subset of the sequencing dataset. Typically 3 millions of reads are randomly selected (this number can be adjusted by the user depending on the chloroplast coverage). A coverage is computed for each resulting *contig*. Contigs are then aligned to the chloroplast proteome of a close species. Those that present significant similarity are tagged as *near-to-the-proteome* contigs. The one that presents the highest similarity with a predefined gene (which is known to be present in a single copy), typically the *matK* gene, is set as the *seed contig*. Its coverage becomes the reference coverage.

From these set of contigs, we then search for *links* between them. This is done by extending all the near-to-the-proteome contigs using overlapping read strategy. Contigs,

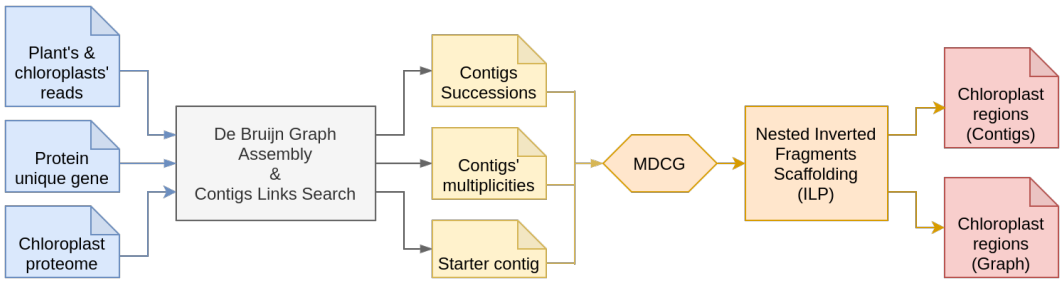


Figure 2 Method overview.The method can be cut in two main parts. First step is a driven chloroplast assembly with reads that come from both plant's genome and chloroplasts' genome. This assembly is a De Bruijn graph assembly approach. It outputs contigs' successions and an estimated multiplicity for each contig. These results are structured in a Multiplied Doubled Contigs Graph (MDCG). The following scaffolding step is done thanks to an Integer Linear Programming (ILP) optimisation. Finally, we output a fewer number of larger contigs that correspond to chloroplast's genomic regions, and a scaffolding enriched assembly graph, as illustrated Subfigure 1b.

other than seed contigs, having connection with seed contigs are fished out and added to the contig set.

We define the contigs set $\mathcal{C}$ as the set containing near-to-the-proteome tagged contigs and those which are intermediate in paths between two tagged ones. For each contig we compute a *multiplicity value* as an approximated ratio between its estimated coverage and the seed contig's coverage (the reference coverage). Thus, a contig's multiplicity may be interpreted as an upper bound of the number of times the associated contig can be met in the final assembly.

To any contig is associated a nucleotide sequence which length is provided as input data. A contig can participate in the solution either in its *forward* orientation (so the sequence does not change), or in its *reverse* orientation (the sequence is read in inverse reading and each nucleotide is complemented). However, the orientation of a contig participating in the solution is unknown *a priory*.

Let $\mathbb{N}$ denote the set of natural numbers while $\mathbb{N}^* = \mathbb{N} \setminus \{0\}$. As usual, $\mathbb{B}$ denotes the set of binaries.

Any contig $c \in \mathcal{C}$ is characterized by its identifier $c_{id} \in \mathbb{N}$ and its multiplicity $c_{mult} \in \mathbb{N}^*$ (an upper bound of the number of occurrences of c). Contigs and their multiplicity are illustrated in Subtable 1a.

In the next step we to find paths between tagged contigs in the DBG. These paths are defined as *links* and we denote the corresponding set by $\mathcal{L}$. When such a link traverses a limited number of non-tagged contigs, these contigs are selected to be in the final contigs set. Therefore, the links are *succession relations* between a couple of *oriented contigs*, *i.e.* $\mathcal{L} \subset (\mathcal{C} \times \mathbb{B})^2$. Each contig c is provided here with an orientation attribute $orc_{or} \in \mathbb{B}$ (thus c becomes orc), where the binary value 0 stands for *forward* (f), *i.e.* the original contig's sequence, while the binary value 1 stands for *reverse* (r), *i.e.* the reverse complement of the contig's sequence. Subtable 1b gives an illustration for links data.

2.3 Multiplied Doubled Contigs Graph (MDCG)

Building chloroplast genomic regions implies finding a sequence of linked oriented contigs, in the limit of multiplicity for the two orientations for each contig. Links between two multiplied (by their multiplicity) and oriented (forward or reverse) contigs can be structured in an

Table 1 An instance of input data. It contains two types of data: **(a)** Set $\mathcal{C}$ of contigs with their multiplicity (c_{mult}); **(b)** Set of links $\mathcal{L}$. An identifier (orc_{id}) and an orientation (orc_{or}) is provided here for each oriented contig orc serving as a link extremity. For the sake of saving memory, for each link $(orc, ord) \in \mathcal{L}$, only one of the existing two links is reported in the table, the one with $orc_{id} < ord_{id}$.

(a) Contigs data: $c \in \mathcal{C}$

c_{id}	c_{mult}
2	2
3	1
4	1
5	2

(b) Links data: $(orc, ord) \in \mathcal{L}$

orc_{id}	orc_{or}	ord_{id}	ord_{or}
2	1	3	1
2	1	3	0
2	0	5	1
4	0	5	0

oriented graph. Thus, we define the *Multiplied Doubled Contigs Graph*, $MDCG = (V, E)$, a directed graph where V is the set of vertices and E is the set of edges[1]. Even if in practice we do not multiply and double contigs and links data in memory, each vertex $v \in V$ is one occurrence of an oriented contig and each edge $(u, v) \in E$ corresponds to a link between occurrences of two oriented contigs. The construction rules and some MDCG properties are as follows:

- **(i)** Each contig $c \in \mathcal{C}$ is doubled according forward and reverse orientation. So $\forall v \in V$ such as $v_{id} = c_{id}$, v has an orientation attribute v_{or} equals to 0 or 1 respectively for forward or reverse orientation. We denote by $\overline{v}$ the reverse orientation of v (so that $v_{or} = 1 - \overline{v}_{or}$, and the other attributes remain the same). For each vertex $v \in V$, its reverse $\overline{v}$ is in the graph too. By definition, vertex's identifier remain the same $v_{id} = \overline{v}_{id}$, and likewise for vertex's occurrences, $v_{occ} = \overline{v}_{occ}$.
- **(ii)** Moreover, each doubled contig is multiplied by its multiplicity. Thus, if contig c is multiplied k times, it generates a set of $2k$ vertices $v_0, \ldots, v_{k-1}$, $\overline{v}_0, \ldots, \overline{v}_{k-1}$ where all these vertices have the same identifier c_{id}. Let $0 \leq v_{occ} < c_{mult}$ be an occurrence of a multiplied contig c.
- **(iii)** The set E of edges is constructed based on the links data $\mathcal{L}$ and applying the same multiplication rules as for the contigs. Furthermore, for any edge in E, its reverse is also added in E in order to validate the feature $(u, v) \in E \iff \overline{(u, v)} = (\overline{v}, \overline{u}) \in E$.

Furthermore, we add into MDCG the data that we define as *inverted fragments* and denote by InvF (its formal definition is given in section 2.5.2). Each inverted fragments is a non-oriented couple of vertices (i, j), such that they are two different occurrences of the same contig, but one is in forward orientation, while the other is in reverse orientation. These couples are candidates to be part of inverted repeats pairs.

2.4 Nested Inverted Fragments Scaffolding Problem

Thanks to the MDCG structure, building a sequence of linked oriented contigs is equivalent to finding an elementary path in MDCG. This is a necessary, but not a sufficient condition to correctly retrieve a circular genome with pairs of inverted repeats. Especially, no more that one of the vertices $v \in V$ and its reverse $\overline{v} \in V$ can be in the path. The seed contig is illustrated as a big black dot in Subfigure 1a & 3 (seed vertex $s \in V$, in arbitrarily forward

[1] The graph modeling described here has been yet applied in our previous scaffolding papers [1, 7]. However, the notation MDCG is used here firstly.

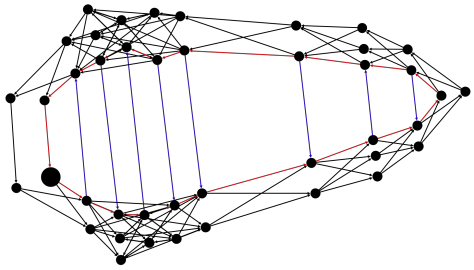


Figure 3 An instance of *Multiplied Doubled Contigs Graph*. The input contigs were multiplied by their multiplicity number, then doubled according to two DNA strands. The obtained graph possesses 42 vertices and 130 edges. Vertices candidate to participate in inverted repeats have one of their reverse oriented versions linked by a blue edge. They form a couple of vertices defined as *inverted fragments*. The solution (the assembled genome) is represented as a path in red. It passes through 8 adjacent inverted fragments that represent inverted repeats. It begins with the biggest vertex (a given starter, as illustrated in Subfigure 1a) and finishes in the same vertex since the chloroplast genomes are circular.

orientation). The circularity of the genome corresponds to a path that begins at s and ends in s.

Finding a path requires giving a position for each vertex participating in it. As illustrated in Subfigure 1a, two inverted fragments (i, j) and (k, l) in inverted repeats regions are *nested* *i.e.* if you draw in the corresponding circuit a line from i to j, and another from k to l, then the two lines do not intersect. In order to know if two inverted fragments intersect, we must compare the positions of the associated vertices.

Problem definition

Finally, the goal of the Nested Inverted Fragments Scaffolding problem (NIFSP) is to find in the corresponding MDCG an elementary path from s to s, that includes at most one of any of the vertices $v \in V$ or its reverse $\overline{v} \in V$, and that traverses as many as possible inverted fragments in the way that they are located on the path as contiguous nested pairs.

An illustration for such a path is given in Figure 3. To solve the NIFSP we describe below a linear programming formulation where vertices' position, their relative location, chosen edges and chosen inverted fragments are all integer or binary variables.

2.5 Modelling the Scaffolding Problem as a Path Search in a Graph: Integer Linear Programming Formulation

In the sequel, let N_v^- and N_v^+ denote respectively the sets of predecessors and successors of a vertex $v \in V$.

In order to define a path in a graph, one usually needs to designate its start and its terminal vertex. We use here for this purpose the given seed contig. We denote its corresponding vertex by sc and we suppose that it is in its forward orientation. We also know that it is present in the graph with only one occurrence. Furthermore, we replace sc by two new vertices s and t, both in forward orientation and with only one occurrence, where s gets all outgoing sc edges, while t gets all incoming sc edges. Note that we add neither $\overline{s}$ nor $\overline{t}$ and that $N_s^- = N_t^+ = \emptyset$, where $\emptyset$ denotes the empty set.

2.5.1 Path Definition

Equations (1)-(7) in this section follow the single-flow formulation of an $s-t$ path that is used in the field [14] and that we have adapted to the scaffolding problem specificities in our previous papers [1, 7].

Vertices s and t will be used as the source (start) and the target (terminal), respectively, of the path we are looking for. We introduce a binary variables x_e to indicate if the corresponding edge e participates or not in the path:

$$\forall e \in E: \quad x_e \in \mathbb{B}. \tag{1}$$

An unique path exits from s and enters in t, *i.e.*:

$$\sum_{v \in N_s^+} x_{sv} = \sum_{u \in N_t^-} x_{ut} = 1 \tag{2}$$

With any vertex $v \in V \setminus \{s,t\}$ is associated a variable $i_v \in [0,1]$ to encode whether v is, or isn't in the solution path. If $v \notin \{s,t\}$ is an intermediate vertex in the path, then v possesses exactly one incoming/outgoing edge. If v is not in the path, then no edge inputs/exits v, *i.e.*

$$\forall v \in V \setminus \{s,t\}: \quad i_v = \sum_{u \in N_v^-} x_{uv} = \sum_{w \in N_v^+} x_{vw} \tag{3}$$

Note that although i_v is declared as contiguous variable, it takes 0/1 value because of (3). This is a particular case of a more general theorem proven in [7].

Furthermore, for each vertex v, at most one of its orientations participates in the path:

$$\forall v \in V \setminus \{s,t\}: \quad i_v + i_{\overline{v}} \leq 1 \tag{4}$$

We introduce a variable $f_e \in \mathbb{R}$ to express the quantity of the flow circulating along the edge $e \in E$. No flow can use an edge e when $x_e = 0$, *i.e.*

$$\forall e \in E: \quad x_e \leq f_e \leq x_e \times M \tag{5}$$

where $M = |V|$. For the initial/intermediate vertex flow we set respectively:

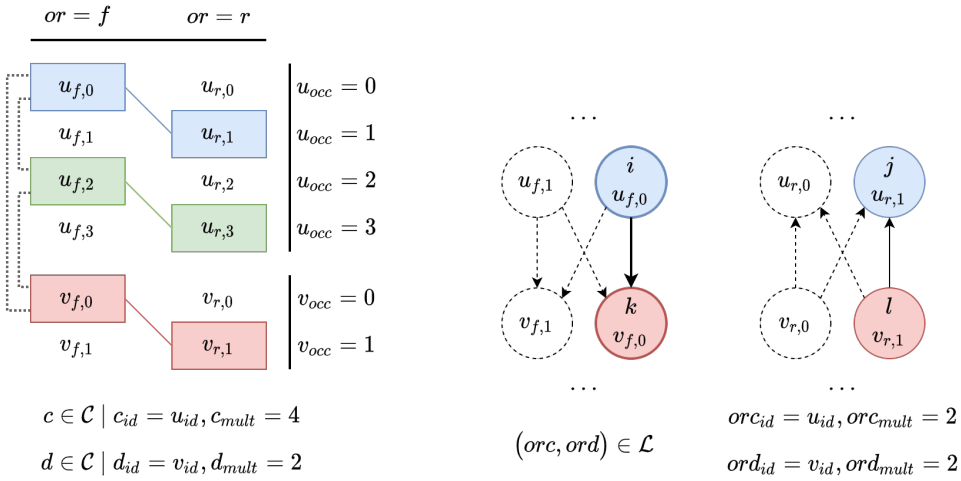
$$\sum_{v \in N_s^+} f_{sv} = 1 \tag{6}$$

$$\forall v \in V \setminus \{s,t\}: \quad \sum_{w \in N_v^+} f_{vw} - \sum_{u \in N_v^-} f_{uv} = i_v \tag{7}$$

Constraint (7) makes the flow increasing and forbids sub-tours. On the other hand, the exit flow value can be interpreted as a position/number of a vertex v on the genome in respect to the start s. Accordingly, we denote this value by pos_v, *i.e.*

$$\forall v \in V \setminus \{t\}: \quad pos_v = \sum_{w \in N_v^+} f_{vw} \tag{8}$$

Constraints (4) – (7) define an elementary path from s to t. In the sequel, we add to this model non-intersecting inverted fragments constraints.



(a) Inverted fragments and pairs

(b) Adjacent inverted fragments

Figure 4 Inverted fragments sets illustrations. **(a)** Two vertices $i, j \in V^2 \mid i_{id} = j_{id}$ are inverted fragments if they are coloured identically. They belong to the $InvF$ set. As an illustration of the `invfrag` function, note that $\texttt{invfrag}(u_{f,0}) = (u_{f,0}, u_{r,1})$, and $\texttt{invfrag}(u_{r,3}) = (u_{f,2}, u_{r,3})$. Inverted fragments linked by a dashed line represent pairs that belong to $PInvF$ set (here three). Two inverted fragments $(p, q) \in PInvF$ can be chosen in the solution only if they do not intersect. **(b)** $(i, j) \in InvF$ and $(k, l) \in InvF$ are two adjacent inverted fragments in MDCG. Remind that $k \in N_i^+ \iff j \in N_l^+$. We aim to maximise the number of such adjacent inverted fragments in the solution. We do not consider dashed edges in $AInvF$ because their extremities participate in no one of the $InvF$'s item. Note that edge $(i, k) \in E$ (in bold) is a canonical — edge $(l, j) \in E$ can be retrieved by $\texttt{adjrev}((i, k)) = (l, j)$.

2.5.2 Inverted Fragments

Let $\mathcal{R}$ be the set of contigs with multiplicity greater than 1, *i.e.* $\mathcal{R} = \{c \in \mathcal{C} \mid c_{mult} > 1\}$. An oriented couple consisting of a forward occurrence of a contig from $\mathcal{R}$, and another occurrence of it, but in reverse orientation is called *inverted fragment* (*c.f.* Subfigure 4a for illustration). Denote $InvF$ the set of inverted fragments:

$$InvF = \left\{ (i,j) \in V^2, \forall c \in \mathcal{R} \;\middle|\; \begin{array}{l} (i_{id} = j_{id} = c_{id}) \wedge (i_{or} = 1 - j_{or} = 0) \\ \wedge \left(i_{occ} = j_{occ} - 1 = 2 \times k, 0 \leq k < \left\lfloor \frac{c_{mult}}{2} \right\rfloor \right) \end{array} \right\}$$

Here the first set definition determines inverted fragments, while the second one describes the minimum number of items to build all distinct solutions.

It is also necessary to describe a set of pairs of inverted fragments:

$$PInvF = \left\{ \big((i,j),(k,l)\big) \in InvF^2 \;\middle|\; (j_{id} < k_{id}) \vee (j_{id} = k_{id} \wedge j_{occ} < k_{occ}) \right\}.$$

The condition on the identifier and the occurrences attributes assures that for each pair $(p, q) \in PInvF$, its permutation pair $(q, p) \notin PInvF$, and generates the minimum number of items for building all distinct solutions (*c.f.* Subfigure 4a for illustration). The set $PInvF$ contains unordered pairs of inverted fragments that must be checked for admissibility in respect to constraint (12).

Let $\big((i, j), (k, l)\big) \in PInvF$ and consider the positions of these inverted fragments in the genome. In order to illustrate the options for their relative locations, and without loss

of generality, we assume that $pos_i < pos_j, pos_k < pos_l$ and that the interval $[\![pos_i, pos_j]\!]$ is smaller or equal to the interval $[\![pos_k, pos_l]\!]$. Only one of the following three cases holds:

- $[\![pos_i, pos_j]\!] \cap [\![pos_k, pos_l]\!] = \emptyset$
- $[\![pos_i, pos_j]\!] \subseteq [\![pos_k, pos_l]\!]$
- $[\![pos_i, pos_j]\!] \cap [\![pos_k, pos_l]\!] \neq \emptyset$ and $[\![pos_i, pos_j]\!] \not\subseteq [\![pos_k, pos_l]\!]$

The first two cases correspond to feasible solutions to our problem, while in the last one the inverted fragments intersect. The later case is not admissible and will be forbidden as shown below.

Denote by D_{alpha} be the set of couples of vertices whose positions must be compared in order to detect intersection, and define it as follows:

$$D_{alpha} = \big\{(i,k),(i,l),(j,k),(j,l) \quad \forall\big((i,j),(k,l)\big) \in PInvF\big\}$$

$\forall (u,v) \in D_{alpha}$ we introduce a binary variable α_u^v to indicate the relative locations of these vertices. α_u^v equals 1 if $pos_u < pos_v$, otherwise 0. We formulate this definition as the below big-M constraint where $M = |V|$:

$$\forall (u,v) \in D_{alpha}: \quad -\alpha_u^v \times M \leq pos_u - pos_v \leq (1-\alpha_u^v) \times M - 1 \tag{9}$$

$$\alpha_u^v \leq pos_u + pos_v \tag{10}$$

Note that $\alpha_u^v = 1 - \alpha_v^u$, hence only one of the variables α_u^v and α_v^u needs to be maintained.

Furthermore, let us consider two couples of inverted fragments $(i,j),(k,l) \in PInvF$. It is easy to check that amongst the 24 possible permutations between their respective positions, 8 correspond to intersections that are not feasible solutions and we need to exclude them from consideration. To any of these intersections we associate a binary variable. When the pair (i,j) is chosen to be the first argument, we introduce four binary variables $inters_n(ij,kl), n = 1,\ldots,4$. For example, the case $pos_i < pos_k < pos_j < pos_l$ will be associated with a binary variable $inters_1(ij,kl)$ that equals 1 if the previous three relations are true, 0 otherwise. This definition is related to (9) by the following constraints:

$$3 \times inters_1(ij,kl) \leq \alpha_i^k + (1-\alpha_j^k) + \alpha_j^l \leq 2 + inters_1(ij,kl) \tag{11}$$

We need four additional binary variables $inters_n(kl,ij), n = 1,\ldots,4$ to model the case when the pair (k,l) is the first argument.

Moreover, we use binary variables $m(i,j),(i,j) \in InvF$, to indicate if the inverted fragment (i,j) can be *matched* (chosen to be in the solution), or not. When the pairs (i,j) and (k,l) do intersect, no more than one of them can participate in the solution. We model this condition by the below constraint:

$\forall\big((i,j),(k,l)\big) \in PInvF$

$$m(i,j) + m(k,l) \leq 2 - \left(\sum_{n=1}^{4} inters_n(ij,kl) + \sum_{n=1}^{4} inters_n(kl,ij)\right) \tag{12}$$

The vertices of a chosen inverted fragment must belong to the path, *i.e.*

$$\forall (u,v) \in InvF: \quad m(u,v) \leq i_u \; ; \; m(u,v) \leq i_v. \tag{13}$$

2.5.3 Inverted Repeats Contiguity

Inverted fragments represent subsequences in the inverted repeats regions. In this section we study how to assemble them in a contiguous way, and how to make these assembled regions

as long as possible. We achieve this goal by assembling *adjacent inverted fragments*. Those are fragments that are extremities of an edge in the considered MDCG. Since for any edge in E, its reverse is also in E, we keep only one of them (considered as *canonical edge*) in a set of edges called $AInvF$. However, only containing canonical edges implies to be be able to retrieve:

- the inverted fragments associated with any of the extremity of each $(u, v) \in AInvF$. This task is performed by the function `invfrag`: $\{v \in V \mid v_{occ} + (1 - v_{or}) < v_{mult}\} \rightarrow InvF$ (*c.f.* Subfigure 4a for illustration);
- the reverse edge for each $(u, v) \in AInvF$, task performed by the function `adjrev`: $AInvF \rightarrow E$ (*c.f.* Subfigure 4b for illustration).

We introduce a binary variable $isadj(u, v)$, $\forall (u, v) \in AInvF$. It equals 1 if the two inverted fragments associated with the canonical edge (u, v) are contiguous in the solution, 0 otherwise. The contiguity of inverted fragments implies, $\forall (u, v) \in AInvF$:

- the canonical edge (u, v) and its adjacency reverse `adjrev(u,v)` must participate in the solution:

$$isadj(u, v) \leq x_{uv} \ ; \ isadj(u, v) \leq x_{\texttt{adjrev}(u,v)} \tag{14}$$

- the associated inverted fragments do not intersect:

$$isadj(u, v) \leq m(\texttt{invfrag}(u)) \ ; \ isadj(u, v) \leq m(\texttt{invfrag}(v)) \tag{15}$$

2.5.4 Objective Function

The goal of the scaffolding problem considered here is to find an elementary $s - t$ path in the $MDCG$ that contains as many as possible contiguous non-intersecting inverted fragments. We obtain the below problem:

$$\max \sum_{p \in InvF} m(p) + \sum_{(u,v) \in AInvF} isadj(u, v)$$

subject to constraints (4) to (15)

A detailed analysis can show that the number of constraints is $O(|V|^2 + |E|)$ and the above formulation requires $O(|V|^2 + |E|)$ variables.

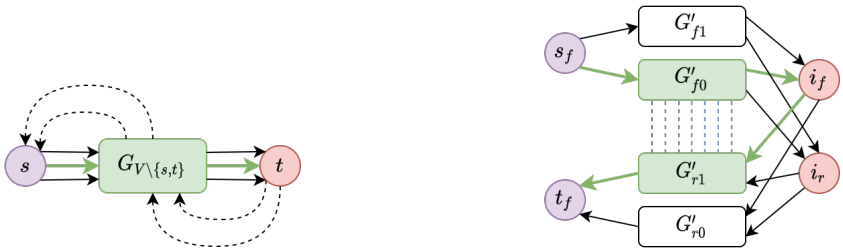
3 Complexity Analysis

▶ **Proposition 1.** *Nested Inverted Fragments Scaffolding Problem (NIFSP) is NP-Hard.*

Proof. By reduction from the longest path problem from vertex s to vertex t (LPSTP), known to be NP-Hard [13].

Consider an instance $\mathbb{I} \in$ LPSTP that is composed of a directed graph $G = (V, E)$ and two vertices $s \in V$ and $t \in V$. We shall build an instance transform function $mdgf$ such that $\mathbb{I} \in$ LPSTP $\iff mdgf(\mathbb{I}) \in$ NIFSP. As illustrated in Figure 5, function $mdgf$ transforms the graph G from 5a to the graph $G' = (V', E')$ in 5b, vertex $s \in V$ to vertices $s_f \in V'$ and $t_f \in V'$, while vertices i_f and i_r in 5b, are images of t in 5a.

All subgraphs G'_* in 5b are quasi-clones of $G_{V \setminus \{s,t\}}$ in 5a. In the graphs $G'_{f0} = (V'_{f0}, E'_{f0})$ and $G'_{f1} = (V'_{f1}, E'_{f1})$ edges are oriented in the same direction as in $G_{V \setminus \{s,t\}}$. Vertices in V'_{f0}, V'_{f1} have forward orientation attribute, and respectively 0 and 1 as occurrence attribute. Vertices in V'_{r0}, V'_{r1} have reverse orientation, while their edges E'_{r0}, E'_{r1} are oriented in the opposite direction compared to E'_{f0}, E'_{f1} edges. The sets $InvF$, $PInvF$ and $AInvF$ are



(a) Directed graph G **(b)** Directed graph G'

Figure 5 Transformation of a directed graph G for LPSTP to a directed graph G' for NIFSP. Edges in bold-green in both subfigures correspond to the solution path for LPSTP and NIFSP problems respectively. **(a)** As the longest path exits s and enters t, dashed edges do not participate in the solution. **(b)** Blue dashed line between vertices in G'_{f0} and G'_{r1} visualise the reverse inverted fragments.

constructed based on G'_* graphs' data. We assume that inverted fragments are composed of vertices from V'_{f0} and V'_{r1} uniquely, (*i.e.* $\forall (u,v) \in InvF, u \in V'_{f0}, v \in V'_{r1}$). This is depicted by blue dashed vertical lines from G'_{f0} to G'_{r1}. Vertices i_f and i_r allow the transition from G'_{f0} and G'_{f1} to G'_{r0} and G'_{r1}.

It is straightforward to see that there exists a $O(|V| + |E|)$ algorithm that computes this transform function.

As adjacent inverted fragments associate only vertices in V'_{f0} with those in V'_{r1}, the path that maximizes the number of contiguous inverted fragments exits s_f, goes through G'_{f0} to i_f (or i_r, it does not matter), and passes through G'_{r1} to t_f. Since G'_{f0} is a copy of $G_{V\setminus\{s,t\}}$, while G'_{r1} is its reverse graph, there is a bijection between V'_{f0} and V'_{r1} vertices sets. Hence maximising the number of contiguous inverted fragments is equivalent to finding the longest path $v_{f0,1} \cdots v_{f0,n}$ in G'_{f0} and its reverse $v_{r1,n} \cdots v_{r1,1}$ in G'_{r1}. These two paths have the same length, that equals the length of the longest path in G.

To conclude, as there is a linear time complexity transform function $mdgf$ such that $\mathbb{I} \in$ LPSTP $\iff mdgf(\mathbb{I}) \in$ NIFSP, NIFSP is at least NP-Hard. ◀

▶ **Proposition 2** (Solution verification is done in polynomial time)**.** *Given a path in MDCG and a set of chosen inverted fragments, there is an algorithm that verifies if these inverted fragments are nested along the path.*

Proof. Let $Path$ be the path of oriented fragments in MDCG, and let $InvF^*$ be the set of chosen inverted fragments.

Algorithm 1 verifies if the solution is admissible. Note that the stack *repeats* has the following four methods:

- `peek` Returns the vertex on the top of the stack without removing it. If the stack is empty, it returns $Null$.
- `add` Adds the vertex on the top of the stack.
- `remove` Removes the vertex on the top of the stack.
- `is_empty` Returns $True$ if and only if, the stack is empty.

Algorithm 1 is linear in the length of the path $Path$.

◀

From Propositions 1 and 2 we conclude that NIFSP is NP-Complete.

Algorithm 1 Given solution verification algorithm.

Require: $MDCG$, $Path$ and $InvF^*$
Ensure: Return $True$ if, and only if the given solution is admissible.

```
repeats ← LIFO()                                   ▷ repeats is a stack.
if Path[0] ≠ s ∨ Path[|Path| − 1] ≠ t then
    return False
for v ∈ Path do
    if invfrag(v) ∈ InvF* then
        u ← repeats.peek()
        if u = v then
            repeats.remove()
        else
            ▷ add the associated vertex of v in their inverted fragment
            repeats.add(invfrag(v)[1 − v_or])
return repeats.is_empty()
```

4 Results

The first assembly step, from reads to contigs' sequences, multiplicities and successions was implemented in Python3. The described ILP formulation for the IR scaffolding was implemented in Python3 using the PuLP package where we run Gurobi solver with academic licence. All the instances have been executed on a Linux laptop computer (32GB RAM, Intel® Core™ i7-10610U CPU @ 1.80GHz ×8).

4.1 Real Instances

4.1.1 Input Data

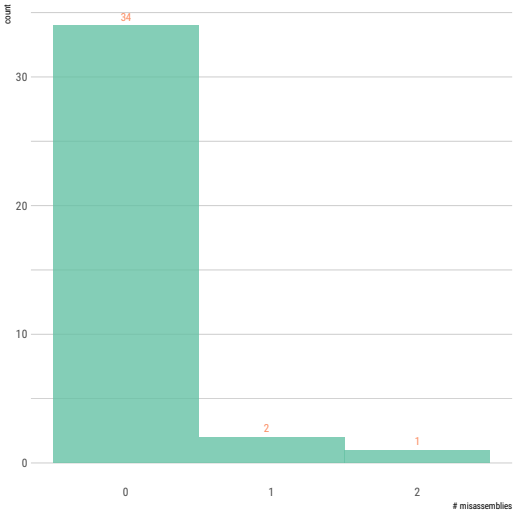
We tested the whole pipeline on reads that mix nuclear and chloroplasts genomes. All reads and reference genomes are the same as in the GetOrganelle paper [12]. A subset of 42 chloroplast genomes with inverted repeats and with a chloroplast coverage at least equal to 15 was chosen.

4.1.2 Quality of the Assemblies

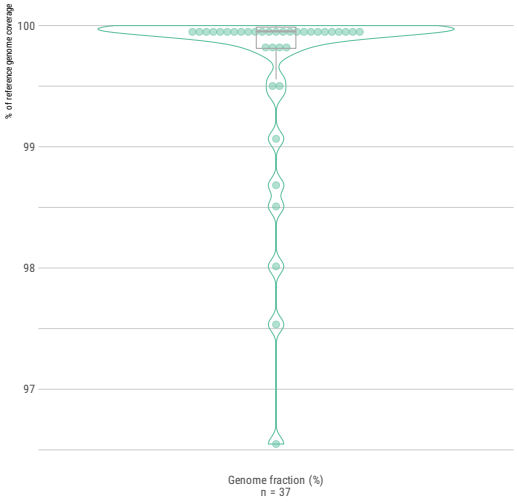
The scaffolding module solved 37 instances out of 42. Further analyses revealed that all unsolved instances failed because of missing links. As a consequence, finding a circular path in MDCG was impossible. We were able to asses the quality of the solution concerning the 37 solved instances as the genomes for these instances are known. We used the well known assembly evaluation tool Quast [9]. The results corresponding to the main measures output by Quast are presented in Figure 6. Tables A.3-**??** given in the appendix reports all details.

Below we give our observations and explanations concerning the erroneous assemblies. All other genomes have been assembled almost perfectly (with a reference genome coverage higher than 99%, *c.f.* Subfigure 6b).

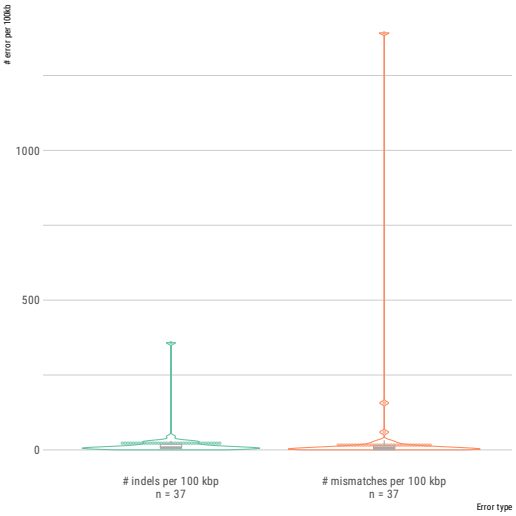
- Aoysia citriodora genome reference is covered at 98.68%. The reason is that one contig is not connected to the main assembly graph component, and another one is missing.
- Boswellia sacra solution covers 98.51% of the reference because one contig is misassembled (the one corresponding to the long unique region). Inverted regions are successfully assembled.



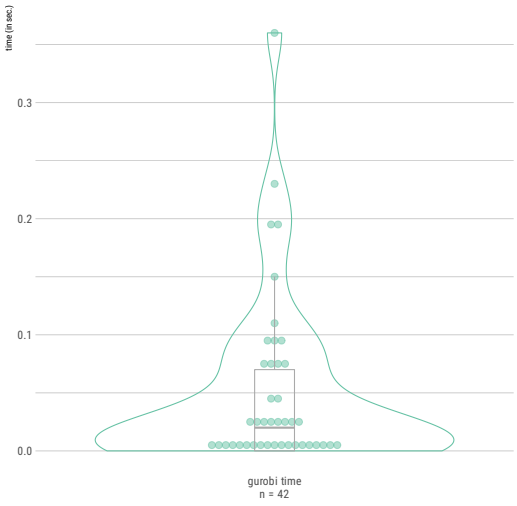
(a) Misassemblies number distribution



(b) Genome coverage percentage distribution



(c) Errors distribution



(d) GUROBI solving time distribution

Figure 6 Distribution of results' measures. Boxplots in violin plots show quartiles, where the whiskers correspond to $1.5 \times IQR$ distances. **(a)** Histogram of misassemblies number occurrences. Here, only one instance outputs contigs with 2 misassemblies, and two instances output each one with one misassembly. **(b)** Distribution of solution genomes coverage (%) by output contigs. **(c)** Distribution of local errors number (per 100kb) on contigs output. While the first distribution shows indels errors type, the second shows mismatch errors type. **(d)** Distribution of GUROBI solving time (in seconds). Each dot represents the mean of the LP relaxation and B&B running times sum.

- Dendrobium nobile solution (96.55%) has both contigs corresponding to the two unique regions misassembled, while the inverted repeats are successfully assembled.
- Haberlea rhodopensis solution covers 97.53% of the reference. Contig corresponding to IR region is shorter than it should be. It is probably due to underestimated multiplicities.
- Jasminum tortuosum solution (99.96%) has one misassembled contig corresponding to unique region. However, inverted repeats are successfully assembled.
- Pimenta dioica solution covers 98.01% the reference genome. Although inverted repeats are successfully assembled, the contig corresponding to LSC region is missing some subsequences.

4.1.3 Time of the Assemblies

For each instance, the first assembly step spends 3 minutes as mean time. Formatting its outputs to MDCG and writing the model with PuLP was less than one second. As NIFSP is proved NP-Complete, we measured the Gurobi solving time. Thus each instance has been run 10 times for the ILP solving step. Subfigure 6d shows the distribution of the LP relaxation and Branch & Bound (B&B) times sum.

4.1.4 Multimerism Revealed by Final Assembly Graph

For each feasible instance, the scaffolding module outputs 3 contigs: two of them correspond to two unique regions known as long and short single copy (LSC & SSC). The the third one corresponds to one of the inverted repeats region (IRa or IRb). In addition of these contigs, the final assembly graph that visualises the connections between the three genomic regions is provided. This assembly graph is enriched by reverse properties as illustrated in Subfigure 1b. Indeed, the graph in Subfigure 1b shows that there are two possible circular sequences of genomic regions: $0 - 1_{f,0} - 2_f - 1_{r,1}$ and $0 - 1_{f,0} - 2_r - 1_{r,1}$. These two equivalent solutions illustrate multimerism phenomenon for chloroplast genomes, studied in [5]: two conformations can simultaneously exist in the same plant's cell. In fact, one of the unique genomic region between the two inverted repeats (*e.g.* the red region in Subfigure 1a & 1b) exists in forward orientation in some genomes, and in reverse orientation in others.

4.2 Artificial Data

In order to better analyse the algorithmic limit and the behaviour of our linear model, we applied it to artificially generated data instances (contigs, links and multiplicities). Each instance has been run 5 times. Results are reported in Table 2. The average gap of the linear relaxation bound (UB) with respect to the optimal integer value (Opt), computed as $100 \times \frac{(UB-Opt)}{UB}$ (MIP gap), differs from zero for only one of the reported instances. Moreover, all the instances have been solved at the first B&B node. They confirm the quality of the integer linear formulation: the gap values stay extremely small even for the largest instance and all instances have been solved at the first B&B node.

5 Conclusion

In this paper we propose a dedicated chloroplast genome assembler. Although the first step is driven by a proteome from a close species, the second step is a scaffolding approach only based on chloroplast genomic knowledge. The first step provides contigs, an estimated multiplicity for each one and links between oriented contigs. Then the scaffolding strategy

Table 2 Benchmark on artificial data. **id**: instance's identifier; $|V|$ and $|E|$ are respectively the number of vertices and edges in MDCG; **time** (in second): average LP relaxation time (above), and B&B time (below); **opt**: optimal value for the LP relaxation (above), the integer solution (below); **%gap**: gap value of the linear relaxation bound with respect to the optimal integer value; **nodes**: number of B&B nodes exploration; **iter**: number of iterations for the LP relaxation (above) and for the B&B phase (below). $U1$, IR (so $\overline{IR}$) and $U2$ are the regions illustrated in Subfigure 1a, and respectively correspond to purple, green left/right-side, and red regions. $|U1|$, $|IR|$ and $|U2|$ report number of contigs (above) and sum of contig's multiplicity (below), for respectively the purple region, the green regions (one version) and the red region.

id	$\|V\|$	$\|E\|$	$\|U1\|$	$\|IR\|$	$\|U2\|$	time	opt	% gap	nodes	iter
0	152	248	15	20	15	.07	39	0	1	1690
			17	41	17	.66	39			3083
1	304	512	30	40	30	1.43	79	0	1	5938
			33	85	33	3.55	79			11109
2	438	704	45	60	45	2.18	119	0	1	13992
			47	121	50	14.43	119			36472
3	592	976	60	80	60	11.22	160	.62	1	21672
			65	168	62	25.3	159			42873
4	742	1228	75	100	75	22.67	200.5	.75	1	32511
			81	211	78	46.06	199			64963
5	906	1524	90	120	90	89.13	242.5	1.44	1	53394
			99	256	97	202.24	239			156103
6	1030	1678	105	140	105	87.98	282.5	1.24	1	63680
			113	289	112	129.44	279			122852
7	1188	1974	120	160	120	313.59	321.5	.78	1	96122
			130	338	125	373.97	319			161433
8	1348	2242	135	180	135	235.91	366.3798	2.01	1	152044
			147	377	149	3778.72	359			312150
9	1504	2526	150	200	150	564.12	404.3697	1.33	1	207623
			164	426	161	1012.3	399			493610

aims to built a special case of genomic regions: inverted repeats. Thus, it consists in finding an elementary circular path into MDCG that maximises the number of nested and contiguous inverted fragments.

This optimisation problem defined as NIFSP is proven to be NP-Complete. We also design an integer programming formulation and apply the associated implementation to solve real genomic and artificial data instances.

Concerning real data instances, first results evaluated by QUAST are very encouraging: the scaffolding module rarely outputs misassembled contigs and mismatches and indels errors rates is very low. Furthermore, the results on artificial data demonstrate that our formulation is very tight — the MIP gap value stays extremely small and we successfully solve large instances at the first B&B node.

Since chloroplast genomes are multimeric (regions between inverted repeats can be considered in their both orientations) this raises the question of their suitable representation. To answer this question our strategy here is to explicitly reveal this characteristic using an enriched assembly graph where each region corresponds to one node.

On the one hand, we propose here a scaffolding strategy for inverted repeats which do not suffer from any degeneration. However, a recent study suggests cases in which a gene can exist in a single occurrence instead of two whereas it belongs to inverted repeats [16]. Therefore, one repeat is split into two parts, resulting in the loss of contiguity between the inverted fragments. Thus the contiguity constraint may appear restrictive in this context, as the solver may prefer to bypass the contig containing the gene in order to make contiguous the inverted fragments before and after it. We can easily solve this issue by removing the contiguity constraint.

On the other hand, not all chloroplast genomes posses inverted repeats structure. Some of them have a pair of directed repeats, others a combination of different repeat types. Our next step will be to mathematically formulate direct repeats and unique region reconstruction. The linear programming techniques that we have described in this article can be easily adapted to consider these additional cases.

References

1 Rumen Andonov, Hristo Djidjev, Sebastien François, and Dominique Lavenier. Complete assembly of circular and chloroplast genomes based on global optimization. *Journal of Bioinformatics and Computational Biology*, 17(3):1950014, June 2019. doi:10.1142/S0219720019500148.

2 Ralph Bock and Volker Knoop, editors. *Genomics of Chloroplasts and Mitochondria*, volume 35 of *Advances in Photosynthesis and Respiration*. Springer Netherlands, Dordrecht, 2012. URL: http://link.springer.com/10.1007/978-94-007-2920-9, doi:10.1007/978-94-007-2920-9.

3 Rayan Chikhi and Guillaume Rizk. Space-Efficient and Exact de Bruijn Graph Representation Based on a Bloom Filter. In Ben Raphael and Jijun Tang, editors, *Algorithms in Bioinformatics*, Lecture Notes in Computer Science, pages 236–248, Berlin, Heidelberg, 2012. Springer. doi:10.1007/978-3-642-33122-0_19.

4 Natasha de Vere, Tim C. G. Rich, Sarah A. Trinder, and Charlotte Long. DNA Barcoding for Plants. In Jacqueline Batley, editor, *Plant Genotyping: Methods and Protocols*, Methods in Molecular Biology, pages 101–118. Springer, New York, NY, 2015. doi:10.1007/978-1-4939-1966-6_8.

5 Xing-Wang Deng, Rod A. Wing, and Wilhelm Gruissem. The chloroplast genome exists in multimeric forms. *Proceedings of the National Academy of Sciences*, 86(11):4156–4160, June 1989. Publisher: National Academy of Sciences Section: Biological Sciences: Genetics. URL: https://www.pnas.org/content/86/11/4156, doi:10.1073/pnas.86.11.4156.

6 Nicolas Dierckxsens, Patrick Mardulyn, and Guillaume Smits. NOVOPlasty: de novo assembly of organelle genomes from whole genome data. *Nucleic Acids Research*, 45(4):e18, February 2017. URL: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC5389512/, doi:10.1093/nar/gkw955.

7 Sebastien François, Rumen Andonov, Dominique Lavenier, and Hristo Djidjev. Global Optimization for Scaffolding and Completing Genome Assemblies. *Electronic Notes in Discrete Mathematics*, 64:185–194, 2018. URL: https://www.sciencedirect.com/science/article/pii/S1571065318300209, doi:https://doi.org/10.1016/j.endm.2018.01.020.

8 Jan A. Freudenthal, Simon Pfaff, Niklas Terhoeven, Arthur Korte, Markus J. Ankenbrand, and Frank Förster. A systematic comparison of chloroplast genome assembly tools. *Genome Biology*, 21:254, September 2020. URL: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC7520963/, doi:10.1186/s13059-020-02153-6.

9 Alexey Gurevich, Vladislav Saveliev, Nikolay Vyahhi, and Glenn Tesler. QUAST: quality assessment tool for genome assemblies. *Bioinformatics*, 29(8):1072–1075, April 2013. doi:10.1093/bioinformatics/btt086.

10 Gurobi Optimization, LLC. Gurobi Optimizer Reference Manual, 2022. URL: https://www.gurobi.com.

11 Dan Gusfield. The RNA-Folding Problem. In Dan Gusfield, editor, *Integer Linear Programming in Computational and Systems Biology: An Entry-Level Text and Course*, pages 105–121. Cambridge University Press, Cambridge, 2019. URL: https://www.cambridge.org/core/books/integer-linear-programming-in-computational-and-systems-biology/rnafolding-problem/A3329DDD8BACF6C203F57CCCF6E555C2, doi:10.1017/9781108377737.008.

12 Jian-Jun Jin, Wen-Bin Yu, Jun-Bo Yang, Yu Song, Claude W. dePamphilis, Ting-Shuang Yi, and De-Zhu Li. GetOrganelle: a fast and versatile toolkit for accurate de novo assembly of organelle genomes. *Genome Biology*, 21(1):241, September 2020. doi:10.1186/s13059-020-02154-5.

13 Alexander Schrijver. *Combinatorial Optimization: Polyhedra and Efficiency*. Springer Science & Business Media, February 2003. Google-Books-ID: mqGeSQ6dJycC.

14 Leonardo Taccari. Integer programming formulations for the elementary shortest path problem. *European Journal of Operational Research*, 252(1):122–130, July 2016. URL: https://www.sciencedirect.com/science/article/pii/S0377221716000084, doi:10.1016/j.ejor.2016.01.003.

15 Jenny Tschörtner, Bin Lai, and Jens O. Krömer. Biophotovoltaics: Green Power Generation From Sunlight and Water. *Frontiers in Microbiology*, 10, 2019. URL: https://www.frontiersin.org/article/10.3389/fmicb.2019.00866.

16 Monique Turmel, Christian Otis, and Claude Lemieux. Divergent copies of the large inverted repeat in the chloroplast genomes of ulvophycean green algae. *Scientific Reports*, 7(1):994, April 2017. Number: 1 Publisher: Nature Publishing Group. URL: https://www.nature.com/articles/s41598-017-01144-1, doi:10.1038/s41598-017-01144-1.

17 Zheng Xiao-Ming, Wang Junrui, Feng Li, Liu Sha, Pang Hongbo, Qi Lan, Li Jing, Sun Yan, Qiao Weihua, Zhang Lifang, Cheng Yunlian, and Yang Qingwen. Inferring the evolutionary mechanism of the chloroplast genome size by comparing whole-chloroplast genome sequences in seed plants. *Scientific Reports*, 7(1):1555, May 2017. Number: 1 Publisher: Nature Publishing Group. URL: https://www.nature.com/articles/s41598-017-01518-5, doi:10.1038/s41598-017-01518-5.

A Detailed Results on Real Data

Table A.3 Scaffolding results on chloroplast benchmark with reference genomes. **Instance**: species name; **size**: genome length in number of nucleotides (in base-pairs); $|V|$ and $|E|$ are respectively the number of vertices and edges; **time**: CPU solver time (in seconds); **%gnm**: percentage of genome length covered by the solution computed by our approach; **NGA50**: NG50 corrected of assembly errors; **#mis**: number of misassemblies in the proposed solution; **|mis|**: misassemblies total length (in bp); **mism**: number of alignment mismatches per 100 kbp; **indels**: number of alignment indels per 100 kbp.

instance	size	$\|V\|$	$\|E\|$	time	%gnm	NGA50	#mis	|mis|	mism	indels
Allium sativum	153118	12	20	< .01	99.99	82090.0	0	0	0.0	1.96
Aloysia citriodora	154699	58	90	.07	98.68	85159.0	0	0	12.45	27.51
Althaea officinalis	159987	10	18	< .01	99.96	87918.0	0	0	0.63	0.0
Amborella trichopoda	162686	16	24	< .01	99.92	90428.0	0	0	156.86	28.91
Artemisia annua	150952	38	54	.01	99.99	82772.0	0	0	0.0	5.3
Azolla filiculoides	147665	40	64	< .01	-	-	-	-	-	-
Boswellia sacra	159228	38	46	< .01	98.51	40853.0	1	84629	0.64	3.83
Citrus limon	160101	136	208	.2	99.45	86976.0	0	0	32.03	30.15
Coffea arabica	155188	122	194	.36	99.07	84008.0	0	0	59.19	44.88
Dendrobium nobile	152018	42	50	< .01	96.55	33709.0	2	105240	1392.16	358.1
Digitalis lanata	153108	46	74	.05	99.77	83262.0	0	0	6.55	5.24
Dioscorea villosa	153974	64	100	.09	99.95	83877.0	0	0	0.0	5.2
Echinacea atrorubens	151912	24	22	< .01	-	-	-	-	-	-
Echinacea purpurea	151913	154	240	.19	99.95	83480.0	0	0	7.24	15.81
Echinacea sanguinea	151926	34	44	.02	99.99	83674.0	0	0	11.19	7.24
Echinacea speciosa	151860	26	42	.02	99.99	83531.0	0	0	3.29	8.56
Eleutherococcus senticosus	156863	22	34	.01	99.99	86842.0	0	0	8.29	5.74

(the table continues on the next page)

Table A.3, continued

instance	size	$\|V\|$	$\|E\|$	time	%gnm	NGA50	#mis	$\|$mis$\|$	mism	indels
Eriobotrya japonica	159156	34	58	.02	100.0	87263.0	0	0	3.14	5.65
Fragaria virginiana	155577	48	72	.04	99.87	85577.0	0	0	1.93	9.01
Ginkgo biloba	156945	24	38	$<$ .01	-	-	-	-	-	-
Gnetum gnemon	115022	20	32	.01	99.93	66153.0	0	0	5.22	5.22
Haberlea rhodopensis	153099	76	120	.15	97.53	83933.0	0	0	14.06	27.46
Hydrastis canadensis	160000	14	26	.01	100.0	87147.0	0	0	0.0	2.5
Illicium anisatum	142723	18	26	$<$ .01	100.0	100791.0	0	0	3.5	7.01
Illicium floridanum	143571	24	32	$<$ .01	99.95	101328.0	0	0	27.18	17.42
Illicium henryi	143240	28	26	$<$ .01	-	-	-	-	-	-
Jasminum tortuosum	162080	74	122	.08	99.96	89117.0	1	89992	22.22	29.63
Laurus nobilis	152750	40	56	.02	99.9	93540.0	0	0	0.0	5.24
Lupinus albus	151921	44	128	.11	99.99	82282.0	0	0	9.22	11.85
Magnolia biondii	160002	32	56	.03	99.99	88127.0	0	0	0.0	2.5
Magnolia denudata	160089	14	26	.01	99.99	88149.0	0	0	0.0	0.62
Magnolia officinalis	160136	52	76	.03	99.99	88169.0	0	0	3.75	4.37
Mitragyna speciosa	155600	68	104	.1	99.56	85060.0	0	0	13.56	25.18
Nicotiana tabacum	155943	8	16	$<$ .01	99.99	86706.0	0	0	0.0	0.0
Physcomitrella patens	122890	36	40	$<$ .01	-	-	-	-	-	-
Pimenta dioica	158984	126	178	.23	98.01	84223.0	0	0	6.42	19.25
Piper auritum	159909	50	86	.1	99.95	87529.0	0	0	15.02	15.64
Piper nigrum	161523	50	82	.07	99.86	89114.0	0	0	20.46	19.84

(the table continues on the next page)

Table A.3, continued

instance	size	$\|V\|$	$\|E\|$	time	%gnm	NGA50	#mis	|mis|	mism	indels
Prunus dulcis	157723	30	46	.02	99.81	85866.0	0	0	2.54	7.62
Scutellaria lateriflora	152283	42	74	.07	99.99	84432.0	0	0	7.22	9.85
Theobroma cacao	160619	36	52	.02	99.97	89335.0	0	0	7.47	11.83
Zea mays	140447	14	34	.01	99.98	82202.0	0	0	6.41	8.55