Global exact optimisations for chloroplast genome multimeric forms scaffolding

Victor Epain, Rumen Andonov

Global exact optimisations for chloroplast genome multimeric forms scaffolding

Victor Epain[1*] and Rumen Andonov[1*]

[1*]GenScale, Centre Inria de l'Université de Rennes, IRISA, 263 Avenue Général Leclerc, Rennes, 35700, France.

*Corresponding author(s). E-mail(s): victor.epain@inria.fr; rumen.andonov@irisa.fr;

Abstract

Background Scaffolding is an intermediate stage of fragment assembly. It consists in orienting and ordering the contigs obtained by the assembly of the sequencing reads. In the general case, the problem has been largely studied with the use of distances data between the contigs. Here we focus on a dedicated scaffolding for the chloroplast genomes. As these genomes are small, circular and with few repeats, numerous approaches have been proposed to assemble them. However, their specificities have not been sufficiently exploited.

Results We give a new formulation for the scaffolding in the case of chloroplast genomes as a discrete optimisation problem, that we prove to be $\mathcal{NP}$-Complete. It does not require distance information. It is focused on a genomic regions view, with the priority on scaffolding the repeats first. In this way, we encode the multimeric forms issue in order to retrieve several genome forms that can exist in the same chloroplast cell. In addition, we provide an integer linear program to obtain exact solutions that we implement in `Python3` package `khloraascaf`. We test it on synthetic data to investigate its performance behaviour and its robustness against several chosen difficulties.

Conclusions While the scaffolding problem is traditionally defined with distances data, we show it is possible to avoid them in the case of the well-studied circular chloroplast genomes. The presented results show that the regions view seems to be sufficient to scaffold the repeats.

Keywords: Genome assembly, Inverted repeats, Integer linear programming, NP-Complete

1 Background

DNA molecule is a support of living mechanism information found in all the living organisms. Although the interactions between other molecules and the DNA molecule are very complex and sometimes require to investigate the 3D molecules' conformation, most studies consider the DNA molecule only as a sequence of nucleotides. This sequence can be seen as a word on the alphabet $\Sigma^+ = \{A, T, G, C\}$. Genetic, genomic and epigenetic analysis lead to combinatorial problems that need to be solved by computing approaches and thus require to pass from the DNA molecule to a word representation in a computer. This process is described as sequencing the molecule using a sequencing technology.

The scaffolding in the process of fragment assembly

Current technologies are not yet able to read an entire DNA molecule. They output a huge amount of small overlapping erroneous DNA sequences,

named *reads*. Moreover, due to the double-strand structure of the DNA, the reads come from either one strand or its complement in reverse order, and the sequencing technologies cannot assure that two reads were sequenced from the same strand. Thus, each read must be considered in two orientations: the one given by the technologies (defined as the *forward* orientation), and its reverse-complement (defined as the *reverse* orientation) (e.g. $ATGCCA$ and $TGGCAT$ are each other's reverse-complement).

From the reads, genome assembly methods aim to find the longest true DNA sequences. Note that it is sufficient to only find one strand out of two, as the other is obtained by the reverse-complement transformation of the first one. Assembly methods are often split into three major stages: (i) *assembly* of reads based on their overlaps to obtain longer sequences (*contigs*); (ii) *scaffolding*, that aims to obtain an order of oriented contigs, potentially separated by nucleotide distances (*scaffolds*); (iii) *gap-filling*, that aims to complete the assembly by filling the gap between the contigs in the scaffolds. Here we focus on the scaffolding step uniquely.

The vast majority of proposed scaffolding formulations are based on distances data between the reads (paired-end or mate-pair data) that are adapted for the contigs to obtain scaffolds. Read distances can either be counted to represent a confidence linking information between two contigs (Chateau and Giroudeau, 2015; Mandric and Zelikovsky, 2015), either be considered precisely for contig nucleotide positioning (Andonov et al, 2019), or combined both approaches as in (Huson et al, 2002; Wang et al, 2002; Salmela et al, 2011).

Chloroplast genome architecture and multimeric forms

In this paper, we address the scaffolding problem for the particular class of chloroplast genomes. Chloroplasts are organisms living in endosymbiosis with the plants' cells and are responsible for the photosynthesis process. Over the evolution time, the chloroplast genome has reduced in length and loosed in terms of complexity (Xiao-Ming et al, 2017). As a result, chloroplast genomes possess few repeats that are usually equal in nucleotide sequences. One the most studied forms of chloroplast genome is the circular DNA molecule whose sequence may include a pair of highly similar (or equal) nucleotide subsequences, sometimes separated by single-copy ones (*long single-copy, LSC*, and *short single-copy, SSC*) (Bock and Knoop, 2012; Palmer, 1985). There are two types of repeats: (i) the *direct repeats (DR)*, where the sequences are highly similar; (ii) the *inverted repeats (IR)*, where one sequence is the reverse-complement of the other. Figure 1 illustrates usual chloroplast genome architectures.

Furthermore, each chloroplast has multiple copies of its genome, and the molecular forms of the copies differ (*multimeric genome forms*, Bendich (2004)). This phenomenon can be induced by flip-flop inversion: one subsequence is reverse-complemented (*reversed*) during the DNA replication. This inversion is provoked by the existence of facing inverted repeats on either side of the reversed subsequence.

Chloroplast scaffolding approaches

We raise the following two questions: (i) Is it possible to scaffold the class of chloroplast genomes without any distances? (ii) How to include the multimerism into the scaffolding problem formulation?

Although some other dedicated chloroplast genome assemblers and scaffolders have been developed, we consider that the specificities of the chloroplast genome have not been sufficiently exploited. Some of them are pipelines of generic methods applied on cleaned input dataset (Ankenbrand et al, 2018), or based on locally approaches as seed-and-extend algorithms (Coissac et al, 2016; Dierckxsens et al, 2017). `GetOrganelle` (Jin et al, 2020) the multiplicity of each contig is statistically computing to be the closest as possible to their mapping coverage by the reads.

Concerning the handling of multimeric forms, `GetOrganelle` returns several solutions and explore in post-process the corresponding architecture. In Andonov et al (2019) the flip-flop inversion breakpoints are detected in a post-scaffolding-process, and the optimality of the new solutions is proven in the case of the use of distances.

To formalise further the scaffolding by integrating the multimeric forms at its core, we postulate on the particularities of the chloroplast genome: (i) repeats are pairs of regions; (ii) the two regions of a repeat have the same (reversed) nucleotide sequence; (iii) multimeric forms can be

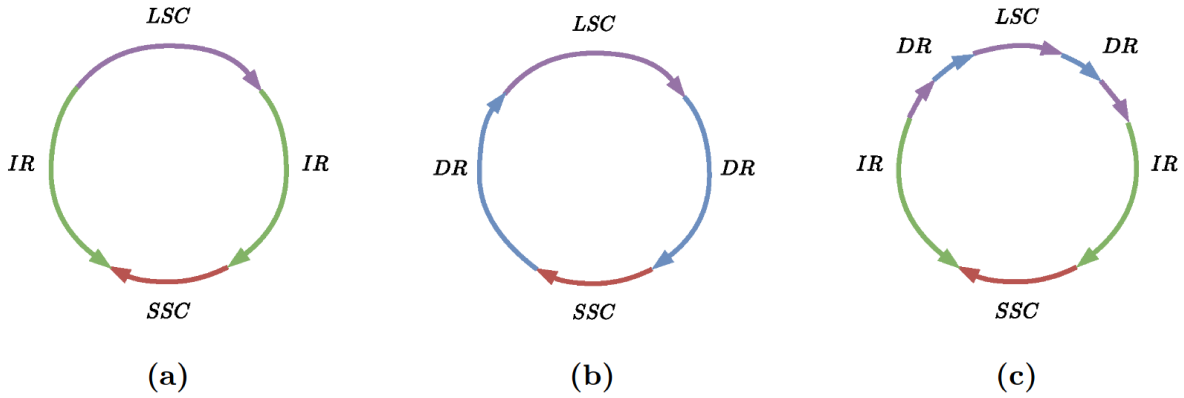


Figure 1 The most studied chloroplast genome form is circular and very often quadripartite. For each of the figures **(a)**, **(b)** and **(c)**, coloured arrows represent nucleotide sequences. LSC and SSC respectively stand for Long Single Copy and Short Single Copy (purple and red). They correspond to regions (subsequences) that are not repeated in the genome. On the opposite, IR and DR respectively stand for Inverted Repeat and Direct Repeat (green and blue). **(a)** This architecture is the most common one and is defined as a quadripartite architecture (Bock and Knoop, 2012). The two green IR arrows face each other and illustrates that one is the reverse-complement sequence of the other; **(b)** the two blue DR arrows are in the same direction and illustrates that one is the same sequence as the other; **(c)** the two types of repeat can simultaneously exist in the chloroplast genome, and direct repeats are shorter than inverted repeats (Palmer, 1985).

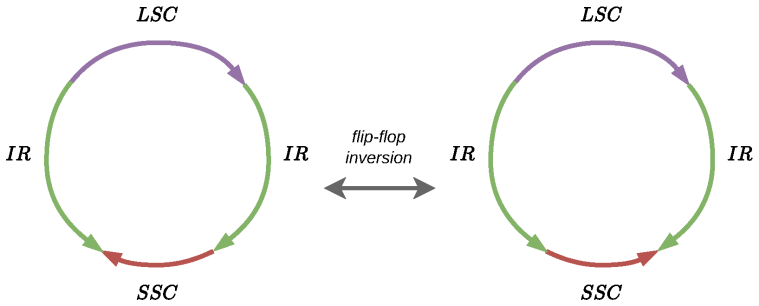


Figure 2 During the DNA replication of the chloroplast genome, one of the region between the two inverted repeats can be reversed (c.f. the red region SSC). This transformation defined as flip-flop inversion provokes the existence of several forms of the genome in the same chloroplast (Palmer, 1983; Deng et al, 1989; Wang and Lanfear, 2019), called *multimeric forms*.

seen as permutations in a sequence of oriented contigs.

Based on these three assumptions, we propose a new scaffolding problem formulation for the chloroplast genome without the use of any distances. Focusing on retrieving the repeats firstly, our approach is region-driven to reveal the multimeric forms thanks to a region graph. We prove this problem to be $\mathcal{NP}$-Complete and we solve it exactly profiting from the small size of the chloroplast instances.

Organisation of the paper

In Section 2 we describe the input data for our approach and give notations and definitions. We formulate the scaffolding problem for chloroplast genomes in Section 3. Section 4 is dedicated to the graph structure for which we apply several Integer Linear Programs (ILP) in Section 5. How we handle the multiple genome forms is described in Section 6. Section 7 explains how we combine the solutions given by the ILPs. The $\mathcal{NP}$-completeness is proven in Section 8. We give some numerical results in Section 9 and we conclude in Sections 10 and 11.

2 Input data and notation

2.1 Scaffolding input data

Let $\mathbb{N}$ denote the set of natural numbers while $\mathbb{N}^* = \mathbb{N} \setminus \{0\}$.

2.1.1 Set of contigs $\mathcal{C}$

Contigs are words in the nucleotides DNA alphabet Σ^+. Each contig c is associated with its *identifier* $c_{id} \in \mathbb{N}$. It can only occur in the genome up to an associated number of times called *multiplicity* $c_{mult} \in \mathbb{N}^*$. Any of its *occurrence* can appear in one of two possible reverse-complementary and mutually exclusive *orientations*: *forward* and *reverse* (aliased by 'f' and 'r', and respectively associated with the binaries 0 and 1). Each contig is provided with a probability that at least one of its occurrences is in the genome (*existence-probability*) $c_{prob} \in [0;1]$. Finally, one contig in this set is defined as the *starter* (s) and must exist in only one occurrence in the genome. Table 1 gives an example of contig set.

2.1.2 Set of links $\mathcal{L}$

Each link is an ordered pair of oriented contigs. The nature of the double-strands DNA requires that $\forall (c,d) \in \mathcal{L}, (\overline{d}, \overline{c}) \in \mathcal{L}$, where $\overline{c}$ and $\overline{d}$ respectively denote the oriented contigs c and d in their reverse orientation (note that $\overline{\overline{c}} = c$). The links between two oriented contigs c and d are valid for all occurrences of c and d respecting the same orientations. Table 1 gives an example of link set: 'f' and 'r' respectively stand for the forward and the reverse orientation, and resp. equal to 0 and 1.

Table 1 Example of input data. Left: set of contigs $\mathcal{C}$. Right: set of links $\mathcal{L}$. For the sake of space, for no one of the links in the table, its reverse is given, although it belongs to $\mathcal{L}$.

c_{id}	c_{mult}	c_{prob}
0	1	0.70
1	2	0.83
2	2	0.17
3	1	0.43

c_{id}	c_{or}	d_{id}	d_{or}
0	f	2	r
0	r	2	f
1	r	2	f
1	f	3	f
1	f	3	r

2.2 Notations

Based on the biological knowledge, we address the dedicated chloroplast scaffolding problem as a region-driven scaffolding, such that specific regions fit into a circular structure. We identify three types of regions to scaffold: the inverted repeats (IR), the direct repeats (DR) and the single copy regions (SC).

Definition 1 (Sequence of oriented contigs)**.** *Consider a sequence* $SOC = c_0 c_1 \cdots c_n$.
— $\forall i \in [\![0; n[\![, (c_i, c_{i+1}) \in \mathcal{L}$
— $\forall c \in \mathcal{C}$, $|\{c' \in SOC \mid c'_{id} = c_{id}\}| \leq c_{mult}$

Definition 2 (Region)**.** *A region* $r = c_0 c_1 \cdots c_n$ *is a sequence of oriented contigs. Region* r *is oriented. Let* $\overline{r} = \overline{c_n} \cdots \overline{c_1 c_0}$ *be the reverse region of* r*. It is composed of the oriented contigs of* r*, considered in their reverse orientation, and given in the reversed order. According to the reverse symmetry in the links,* $\overline{r}$ *also respects Definition 1.*

Definition 3 (Direct repeat – DR)**.** *A DR is a couple of regions* (dr_i, dr_j) *where* $dr_i = dr_j$.

Definition 4 (Inverted repeat – IR)**.** *An IR is a couple of regions* (ir_k, ir_l) *where* $ir_l = \overline{ir_k}$.

Definition 5 (Repeated region)**.** *A repeat is the generic term to denote either DR or IR.*

Definition 6 (Single-copy region – SC)**.** *An SC is a region that is neither a DR nor an IR.*

Definition 7 (Region weight)**.** *The weight* r_{prob} *of a region* r *is defined as:*

$$r_{prob} = \sum_{c \in r} c_{prob}$$

Definition 8 (Sequence of oriented regions)**.** *Consider a sequence* $SOR = r_0 r_1 \cdots r_n$.
— $\forall i \in [\![0; n[\![, (r_i[|r_i| - 1], r_{i+1}[0]) \in \mathcal{L}$
— $\forall c \in \mathcal{C}$, $|\bigcup_{r \in SOR} \{c' \in r \mid c'_{id} = c_{id}\}| \leq c_{mult}$

3 Chloroplast scaffolding problem formulations

Definition 9 (Chloroplast scaffolding problem $\mathcal{CHSP}$)**.** *Consider a set of contigs with their multiplicities and their existence-probabilities, a starting contig and a link set. The aim is to obtain a circular sequence of oriented regions composed of a maximum number of the longest repeats, joined by the maximum-weighted single-copy regions.*

$\mathcal{CHSP}$ involves three subproblems, each one associated with a particular type of region: (i) $\mathcal{DRP}$ for the direct repeat; (ii) $\mathcal{IRP}$ for the inverted repeat and (iii) $\mathcal{SCP}$ for the single-copy. We tackle $\mathcal{CHSP}$ in a hierarchical succession of $\mathcal{DRP}$, $\mathcal{IRP}$ and $\mathcal{SCP}$. Any intermediate problem must preserve the regions found by its predecessors.

Definition 10 (Hierarchical problem succession)**.** *The form of each solution of $\mathcal{CHSP}$ satisfies one of the two problem successions: $\mathcal{DRP}$–$\mathcal{IRP}$–$\mathcal{SCP}$ (h_1) and $\mathcal{IRP}$–$\mathcal{DRP}$–$\mathcal{SCP}$ (h_2).*

The hierarchical approach prioritises the scaffolding of the repeats previously to the SC regions. Indeed, scaffolding the repeats is the most difficult problem as it can lead to misassemblies (wrongly chosen links). Hence, a contig should only be used as many times as possible if its occurrences enable the scaffolding of repeats that most closely represent the architecture of the chloroplast genome, as illustrated in Figure 1. We assume that the existence-probabilities concern events that are less relevant comparing to the genomes architecture and because they are weights on the contigs and not on the links.

The next question is how to prioritise $\mathcal{DRP}$ and $\mathcal{IRP}$? We propose resolving the order by comparing the scores defined in Section 5.5: if $\mathcal{DRP}$ score is better than this of $\mathcal{IRP}$, then the retained succession will be $\mathcal{DRP}$–$\mathcal{IRP}$–$\mathcal{SCP}$, otherwise it will be $\mathcal{IRP}$–$\mathcal{DRP}$–$\mathcal{SCP}$. In the equality case, we discriminate at a further step of the hierarchical successions. The process is detailed in Section 7.

Definition 11 (Chloroplast direct repeat scaffolding problem $\mathcal{DRP}$)**.** *Consider a set of contigs, their multiplicities, a starting contig and a link set. Find a circular sequence of oriented regions SOR, such that:*

- *it contains a maximum number of the longest DR, joined by regions of any kind;*
- *for any couple of DR (i, j) and (k, l) found in SOR such that their respective positions σ respect $\sigma(i) < \sigma(j)$, $\sigma(k) < \sigma(l)$, and $\sigma(i) < \sigma(k)$, then:*
  - $[\![\sigma(i); \sigma(j)]\!] \cap [\![\sigma(k); \sigma(l)]\!] = \varnothing$;
  - *or* $[\![\sigma(k); \sigma(j)]\!] \subset [\![\sigma(i); \sigma(l)]\!]$.

Definition 12 (Chloroplast inverted repeat scaffolding problem $\mathcal{IRP}$)**.** *Consider a set of contigs, their multiplicities, a starting contig and a link set. Find a circular sequence of oriented regions SOR, such that:*

- *it contains a maximum number of the longest IR, joined by regions of any kind;*
- *for any couple of IR (i, j) and (k, l) found in SOR such that that their respective positions σ respect $\sigma(i) < \sigma(j)$, $\sigma(k) < \sigma(l)$ and $\sigma(i) < \sigma(k)$, then:*
  - $[\![\sigma(i); \sigma(j)]\!] \cap [\![\sigma(k); \sigma(l)]\!] = \varnothing$;
  - *or* $[\![\sigma(k); \sigma(l)]\!] \subset [\![\sigma(i); \sigma(j)]\!]$.

Although Figure 6 illustrates the authorised and forbidden positions for the latter defined repeated fragments, it can be generalisable for the $\mathcal{DRP}$ and $\mathcal{IRP}$ regions' position cases.

Definition 13 (Chloroplast single-copy scaffolding problem $\mathcal{SCP}$)**.** *Consider a set of contigs, their multiplicities, their existence-probabilities, a starting contig and a link set. Find a circular sequence of oriented regions such that all regions that are not repeats maximise their weight.*

Note that in Definition 13, if they are no repeats, the problem is reduced to find the maximum-weighted circuit of oriented contigs.

Definition 14 (Chloroplast scaffolding problem succession)**.** *The problems from Definitions 11 to 13 can also take as input a set of fixed regions that must be preserved in the resulting sequence of oriented regions.*

Finally, each hierarchical problem succession produces a circular sequence of oriented regions. From the obtained sequence it is possible to extract a set of ordered pairs of oriented regions. It enables to build several circular sequences of oriented regions of the same length. Each of them representing one possible chloroplast genome

form. This all-equivalent-form process is described in Section 6.

4 Graph and repeated fragment set definitions

In order to efficiently handle the multiplicities of the contigs, hence of the links, we need to build adapted data structures. On the one hand, finding a sequence of oriented contigs, when the links correspond to ordered pairs of oriented contigs, justifies the use of a directed graph to represent the oriented contigs and their links. Section 4.1 defines such a directed graph structure. On the other hand, scaffolding the repeats requires choosing pairs of contigs occurring several times in the oriented contig sequence. Section 4.2 defines the sets of such repeat contig candidate.

4.1 Graph structure

Here we describe a directed graph suitable for further algorithms and mathematical formulation of the scaffolding problems from Definitions 11 to 13.

Definition 15 (Multiplied Doubled Contig Graph – $MDCG$)**.** *Given a set of contigs $\mathcal{C}$, their multiplicities and the link set $\mathcal{L}$, the multiplied doubled contig graph $MDCG = (V, E)$ is defined such that:*

- *V is the set of doubled multiplied contig. Each vertex $v \in V$ is associated with an occurrence of an oriented contig c, and each contig is associated with $2c_{mult}$ vertices. For each vertex $v \in V$ is provided a contig identifier $v_{id} = c_{id}$, a contig fixed orientation $v_{or} \in \{0; 1\}$ and an occurrence $v_{occ} \in [\![0; c_{mult}[\![$*
- *E is the set of multiplied links. For each link $(c, d) \in \mathcal{L}$, there are $c_{mult}d_{mult}$ edges.*

Figure 3 illustrates the $MDCG$ representing the example data given in Table 1.

4.2 Repeated fragment sets

According to Definitions 3 and 4, the repeats are couples of the same oriented contig sequence (modulo reverse-complement for IR). Multiplicities greater than one permit to use several occurrences of the concerned contigs. Denote by $\mathcal{R}$ the set of such contigs, i.e. $\mathcal{R} = \{c \in \mathcal{C} \mid c_{mult} > 1\}$. Definition 15 enables to describe what are the pairs of vertices contained in the two regions of the repeat couples.

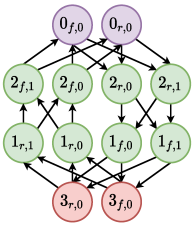


Figure 3 Each vertex is associated with an occurrence of an oriented contig, and each contig is represented by an even number of vertices. For example, vertex labelled $2_{r,1}$ means that it comes from contig 2, in its reverse orientation, and in its second occurrence. The colours are the same as the ones in Figure 1 (a) to relate the input data with the IR architecture.

Definition 16. *A repeated fragment is an unordered pair of vertices such that one of the corresponding oriented contig belongs to the first region of the repeat couple, while the other belongs to the second region. The vertices have the same contig identifier v_{id}, but they differ by their occurrence v_{occ}.*

In Figure 3, $(2_{f,0}, 2_{r,1})$ is a repeated fragment for the IRs. In the following, Definition 16 is specified for each repeat type.

Definition 17. *A direct fragment $(u, v) \in V^2$ is a repeated fragment, such that $u_{or} = v_{or}$. Let $DirF$ be the set of direct fragments:*

$$DirF = \bigcup_{c \in \mathcal{R}} \left\{ \begin{array}{c} (i,j) \in V^2 \ s.t. \\ i_{id} = j_{id} = c_{id} \\ \wedge\, i_{or} = j_{or} \\ \wedge\, i_{occ} = j_{occ} - 1 = 2k \\ 0 \leq k < \left\lfloor \frac{c_{mult}}{2} \right\rfloor \end{array} \right\}$$

Definition 18. *An inverted fragment $(u, v) \in V^2$ is a repeated fragment, such that $u_{or} = 1 - v_{or}$.*

Let $InvF$ be the set of inverted fragments:

$$InvF = \bigcup_{c \in \mathcal{R}} \left\{ \begin{array}{l} (i,j) \in V^2 \ s.t. \\ \quad i_{id} = j_{id} = c_{id} \\ \quad \wedge\, i_{or} = 1 - j_{or} = 0 \\ \quad \wedge\, i_{occ} = j_{occ} - 1 = 2k \\ \qquad 0 \leq k < \left\lfloor \frac{c_{mult}}{2} \right\rfloor \end{array} \right\}$$

Figure 4 (a) illustrates $DirF$ and $InvF$ sets. In addition, we add two functions to retrieve the repeated fragments from the vertices in $\Theta(1)$: `dirfrag`: $V \to DirF$ and `invfrag`: $V \to InvF$ (abstracted with the `repfrag` function, c.f. Section A for their definitions). These functions are applicable only when appropriate.

Furthermore, a repeat is a couple of regions (Definition 5), themselves defined as sequences (Definition 2) requiring to model adjacency between repeated fragments:

Definition 19. *An adjacent repeated fragment is an edge $(u,v) \in E$ such that u and v participate in two distinct repeated fragments.*

Definition 19 is extended for each type of repeat:

Definition 20. *An adjacent direct fragment is an edge between two direct fragments. Let $ADirF$ be the set of adjacent direct fragments:*

$$ADirF = \left\{ \begin{array}{l} (u,v) \in E \ s.t. \\ \quad u_{id} \neq v_{id} \\ \quad \wedge\, u_{occ} = 2k \\ \qquad 0 \leq k < \left\lfloor \frac{u_{mult}}{2} \right\rfloor \\ \quad \wedge\, v_{occ} = 2k' \\ \qquad 0 \leq k' < \left\lfloor \frac{v_{mult}}{2} \right\rfloor \end{array} \right\}$$
$$\bigcup \left\{ \begin{array}{l} (u,v) \in E \ s.t. \\ \quad u_{id} = v_{id} \\ \quad \wedge\, (u_{or} = 0 \vee v_{or} = 0) \\ \quad \wedge\, u_{occ} = 2k \\ \quad \wedge\, v_{occ} = 2k' \\ \qquad 0 \leq k < k' < \left\lfloor \frac{u_{mult}}{2} \right\rfloor \end{array} \right\}$$

Definition 21. *An adjacent inverted fragment is an edge between two inverted fragments. Let $AInvF$ be the set of adjacent inverted fragments:*

$$AInvF = \left\{ \begin{array}{l} (u,v) \in E \ s.t. \\ \quad u_{id} < v_{id} \\ \quad \wedge\, u_{occ} = 2k + u_{or} \\ \qquad 0 \leq k < \left\lfloor \frac{u_{mult}}{2} \right\rfloor \\ \quad \wedge\, v_{occ} = 2k' + v_{or} \\ \qquad 0 \leq k' < \left\lfloor \frac{v_{mult}}{2} \right\rfloor \end{array} \right\}$$
$$\bigcup \left\{ \begin{array}{l} (u,v) \in E \ s.t. \\ \quad u_{id} = v_{id} \\ \quad \wedge\, (u_{or} = 0 \vee v_{or} = 0) \\ \quad \wedge\, u_{occ} - u_{or} = 2k \\ \quad \wedge\, v_{occ} - v_{or} = 2k' \\ \qquad 0 \leq k < k' < \left\lfloor \frac{u_{mult}}{2} \right\rfloor \end{array} \right\}$$

Edges in $ADirF$ and $AInvF$ play the role of *canonical edges* between two adjacent repeated fragments, see Figure 5. In addition, we add two functions to retrieve the adjacent repeated fragments from the edges in $\Theta(1)$: `diradj`: $E \to E$ and `invadj`: $E \to E$ (abstracted with the `repadj` function, c.f. Section A for their definitions). These functions are applicable only when appropriate.

Finally, Definitions 11 and 12 respectively suppose to compare pairs of direct/inverted fragments, that leads to the following:

Definition 22 (Set of pairs of direct fragments)**.**

$$PDirF = \left\{ \begin{array}{l} ((i,j),(k,l)) \in DirF^2 \ s.t. \\ \quad j_{id} < k_{id} \\ \quad \vee \\ \quad j_{id} = k_{id} \wedge j_{occ} < k_{occ} \end{array} \right\}$$

Definition 23 (Set of pairs of inverted fragments)**.**

$$PInvF = \left\{ \begin{array}{l} \big((i,j),(k,l)\big) \in InvF^2 \ s.t. \\ \quad j_{id} < k_{id} \\ \quad \vee \\ \quad j_{id} = k_{id} \wedge j_{occ} < k_{occ} \end{array} \right\}$$

Figure 4 (b) illustrates $PDirF$ and $PInvF$ sets.

The constraints defining $DirF$, $InvF$, $ADirF$, $AInvF$, $PDirF$ and $PInvF$ are explained in details in Section B, where proofs about the minimality of these sets are also given.

5 Integer linear programming (ILP) formulation

Modelling $\mathcal{DRP}$, $\mathcal{IRP}$ and $\mathcal{SCP}$ from Definitions 11 to 13 requires finding a valid circuit in $MDCG$.

Definition 24 (Valid circuit in $MDCG$)**.** *Given a graph $MDCG = (V, E)$, a circuit cp in $MDCG$ is said valid when:*

- *it begins from and ends to the starting vertex s where s_{id} is the starting contig identifier, $s_{or} = f$ and $s_{occ} = 0$;*
- *$\forall\, v \in cp, \overline{v} \notin cp$, where $v_{id} = \overline{v}_{id}$, $v_{or} = 1 - \overline{v}_{or}$ and $v_{occ} = \overline{v}_{occ}$;*
- *two consecutive vertices u and v in the circuit correspond to an edge $(u, v) \in E$.*

First we describe common constraint blocks in Sections 5.1 to 5.3 for ILPs formulations, and then we give the $\mathcal{DRP}$, $\mathcal{IRP}$ and $\mathcal{SCP}$ scaffolding problems ILP in Section 5.5.

Let $\mathsf{M} = \sum_{c \in \mathcal{C}} c_{mult}$ be a constant, N_v^- and N_v^+ respectively be the sets of predecessors and successors of the vertex $v \in V$.

5.1 Circuit constraints

The following set of constraints defines a valid circuit of oriented contig in $MDCG$, and is defined with a flow formulation as in (Andonov et al, 2019) instead of using Miller-Tucker-Zemlin constraints to avoid cycles (Miller et al, 1960).

Binary variables

- x_e to encode whether the edge $e \in E$ is chosen in the path or not.

Continuous variables

- $i_v \in [0; 1]$ to encode whether the vertex $v \in V \setminus \{s; \overline{s}\}$ is intermediate in the path or not. Although it is a continuous variable, it will act as a binary one as proven in (François et al, 2018).
- $f_e \in \mathbb{R}^+$ to encode the position of the edge $e \in E$ in the path (set to 0 if not).

CCircuit ***constraints***

$$x_e \leq f_e \leq \mathsf{M} x_e \qquad \forall e \in E \qquad \text{(C1)}$$

$$\sum_{v \in N_s^+} x_{sv} = \sum_{v \in N_s^-} x_{vs} = 1 \qquad \text{(C2)}$$

$$\sum_{v \in N_s^+} f_{sv} = 1 \qquad \text{(C3)}$$

$$x_{v\overline{s}} = 0 \qquad \forall v \in N_{\overline{s}}^- \qquad \text{(C4)}$$

$$x_{\overline{s}v} = 0 \qquad \forall v \in N_{\overline{s}}^+ \qquad \text{(C5)}$$

$\forall v \in V' :$

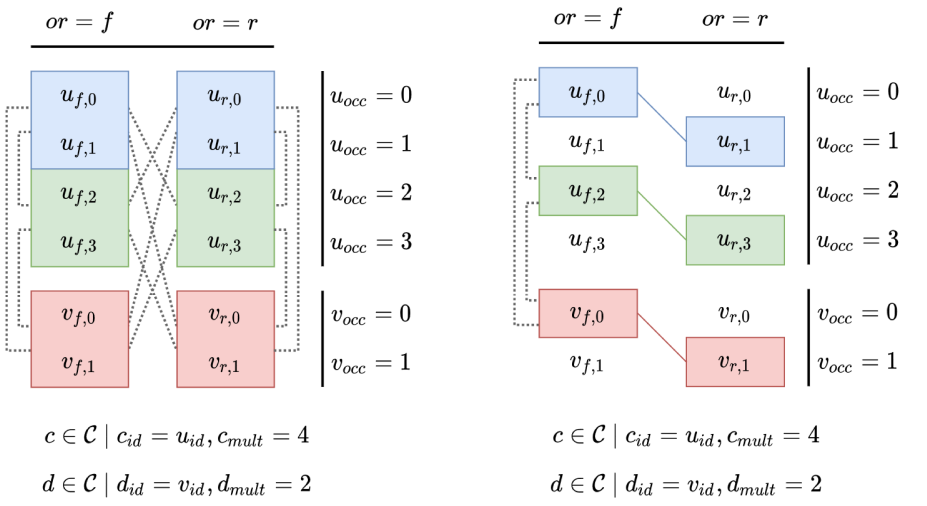
$$i_v + i_{\overline{v}} \leq 1 \qquad \text{(C6)}$$

$$\sum_{u \in N_v^-} x_{uv} \leq i_v \leq \sum_{w \in N_v^+} x_{vw} \qquad \text{(C7)}$$

$$\sum_{w \in N_v^+} f_{vw} - \sum_{u \in N_v^-} f_{uv} = i_v \qquad \text{(C8)}$$

$$x_e \in \{0; 1\} \quad \forall e \in E$$
$$i_v \in [0; 1] \quad \forall v \in V'$$
$$f_e \in \mathbb{R}^+ \quad \forall e \in E$$
$$\text{where } V' = V \setminus \{s; \overline{s}\}$$

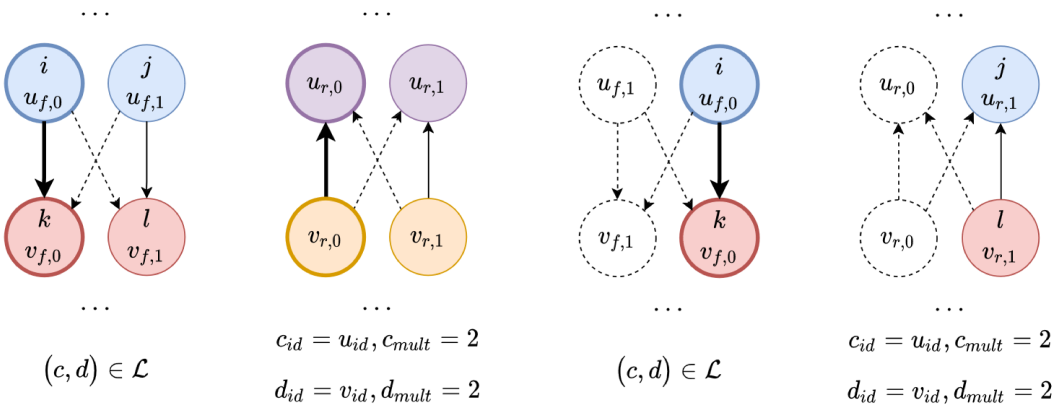
Constraint C1 prevents the flow to be equal to zero if the corresponding edge is not participating. Constraints C2 to C5 force the circuit to begin from and end to the starter in forward orientation. Constraint C6 forces to choose at most one orientation of a vertex, Constraints C7 and C8 define the contiguity of the path and prevent to loop by increasing monotony of the flow.



(a) Direct fragments and pairs of direct fragments

(b) Inverted fragments and pairs of inverted fragments

Figure 4 Repeated fragment sets illustration for the case of the above contigs c and d. Two vertices with the same contig identifier are respectively direct/inverted fragments if they are in the same coloured box, and so they belong to the $DirF/InvF$ set. Direct/inverted fragments linked by a dashed line represent pairs that belong to the $PDirF/PInvF$ set. **(a)** $|DirF| = 6$, and e.g. $(u_{f,0}, u_{f,1}) \in DirF$ so $\texttt{dirfrag}(u_{f,0}) = \texttt{dirfrag}(u_{f,1}) = (u_{f,0}, u_{f,1})$. $|PDirF| = 12$, and e.g. $((u_{f,0}, u_{f,1}), (u_{r,2}, u_{r,3})) \in PDirF$. **(b)** $|InvF| = 3$, and e.g. $(u_{f,2}, u_{r,3}) \in InvF$ so $\texttt{invfrag}(u_{f,2}) = \texttt{invfrag}(u_{r,3}) = (u_{f,2}, u_{r,3})$. $|PInvF| = 3$, and e.g. $((u_{f,2}, u_{r,3}), (v_{f,0}, v_{r,1})) \in PInvF$.



(a) Adjacent direct fragments

(b) Adjacent inverted fragments

Figure 5 Adjacent repeated fragment sets examples. Two vertices of the same colour visualise a repeated fragment. Bold edges (canonical) are the ones that belong to the adjacent repeated fragments sets. The normal edges can be retrieved with the bold ones and `diradj`/`invadj` functions. Dashed edges do not participate in $ADirF/AInvF$. Remind that $(u, v) \in E \iff (\overline{v}, \overline{u}) \in E$. **(a)** edge $(j, l) \in E$ can be retrieved by $\texttt{diradj}(i, k) = (j, l)$. **(b)** edge $(l, j) \in E$ can be retrieved by $\texttt{invadj}(i, k) = (l, j)$.

5.2 Repeated regions constraints

The following constraints are general to define ILPs for $\mathcal{DRP}$ and $\mathcal{IRP}$. Definitions 11 and 12 define the repeated regions according to the positions of the oriented contig in them. It follows that some order of the vertices in the pairs of repeated fragments are allowed, and some others are

forbidden. We decide to write the constraints for the forbidden cases because they are fewer than the allowed ones. To model the forbidden orders between 4 vertices, we can introduce a binary relation: this is what we address with the binary variables α_{ij}.

Specifically for $\mathcal{IRP}$, the constraints modelling the forbidden orders echo those described in the RNA folding problem (Gusfield, 2019), except that the positions of the nucleotides are known for the RNA at the opposite of our case where the contigs' positions are not: hence the need of α_{ij} variables.

According to Definitions 11 and 12, and given $PDirF$ and $PInvF$ (Definitions 22 and 23), denote by $ForbidDR$ and $ForbidIR$ the sets of forbidden quartet vertices for respectively the DR and IR, i.e.:

$$ForbidDR = \left\{ \begin{matrix} (a,c,d,b) \\ (c,a,b,d) \end{matrix} \middle| \begin{matrix} a \neq b, a \in p, b \in p \\ c \neq d, c \in q, d \in q \\ \forall (p,q) \in PDirF \end{matrix} \right\}$$

$$ForbidIR = \left\{ \begin{matrix} (a,c,b,d) \\ (c,a,d,b) \end{matrix} \middle| \begin{matrix} a \neq b, a \in p, b \in p \\ c \neq d, c \in q, d \in q \\ \forall (p,q) \in PInvF \end{matrix} \right\}$$

Figure 6 illustrates the authorised and forbidden positions for $\mathcal{DRP}$ and $\mathcal{IRP}$.

To know if we are in the forbidden cases described in these two sets, we propose to compare the vertices two-by-two. Denote by $AlphaDR$ and $AlphaIR$ the sets containing the couples of vertices to be compared to determine the forbidden cases respectively associated with $ForbidDR$ and $ForbidIR$ sets, such that:

$$AlphaDR = \left\{ \begin{matrix} (i,j),(k,l),(i,k), \\ (j,l),(i,l),(j,k) \\ \text{s.t. } ((i,j),(k,l)) \in PDirF \end{matrix} \right\}$$

$$AlphaIR = \left\{ \begin{matrix} (i,k),(i,l),(j,k),(j,l) \\ \text{s.t. } ((i,j),(k,l)) \in PInvF \end{matrix} \right\}$$

In the following, the sets of repeated fragments and these for the forbidden orders are abstracted to generalise $\mathcal{DRP}$ and $\mathcal{IRP}$. Table 2 gives the correspondence of the sets depending on the problem to solve.

Table 2 ILP sets and functions corresponding table.

ILP	RepF	PRepF	ARepF	`repfrag`	`repadj`
$\mathcal{DRP}$	$DirF$	$PDirF$	$ADirF$	`dirfrag`	`diradj`
$\mathcal{IRP}$	$InvF$	$PInvF$	$AInvF$	`invfrag`	`invadj`

ILP	Forbid	Alpha
$\mathcal{DRP}$	$ForbidDR$	$AlphaDR$
$\mathcal{IRP}$	$ForbidIR$	$AlphaIR$

Binary variables

- m_{ij} to encode whether the repeated fragment $(i,j) \in RepF$ is participating in the path or not i.e. if the (i,j) is a part of a repeat.
- $isadj_e$ to encode whether the edge $e \in ARepF$ connecting two repeated fragments is participating in the path or not.
- $forbid_{ijkl}$ to encode whether the forbidden vertices order $(i,j,k,l) \in Forbid$ is satisfied or not.
- α_{uv} to encode whether the vertex u is before the vertex v in the path or not. In fact, the set of α_{uv} variables is reduced to the strict need of $forbid_{ijkl}$ variables.

Continuous variables

- $i_v \in [0;1]$ to encode whether the vertex $v \in V \setminus \{s;\bar{s}\}$ is intermediate in the path or not. While it is a continuous variable, it will act as a binary one according to (François et al, 2018).
- $f_e \in \mathbb{R}^+$ to encode the position of the edge $e \in E$ in the path (set to 0 if not).

CRepeat ***constraints***

Add the set of constraints *CCircuit*

$$\forall (i,j) \in RepF :$$
$$m_{ij} \leq i_i \quad \text{(C9)}$$
$$m_{ij} \leq i_j \quad \text{(C10)}$$

$$\forall (u,v) \in Alpha :$$
$$pos(v) - pos(u) \leq \mathsf{M}\alpha_{uv} \quad \text{(C11)}$$
$$pos(u) - pos(v) \leq \mathsf{M}(1 - \alpha_{uv}) \quad \text{(C12)}$$
$$pos(u) + pos(v) \geq \alpha_{uv} \quad \text{(C13)}$$

$$\forall (i,j,k,l) \in Forbid :$$

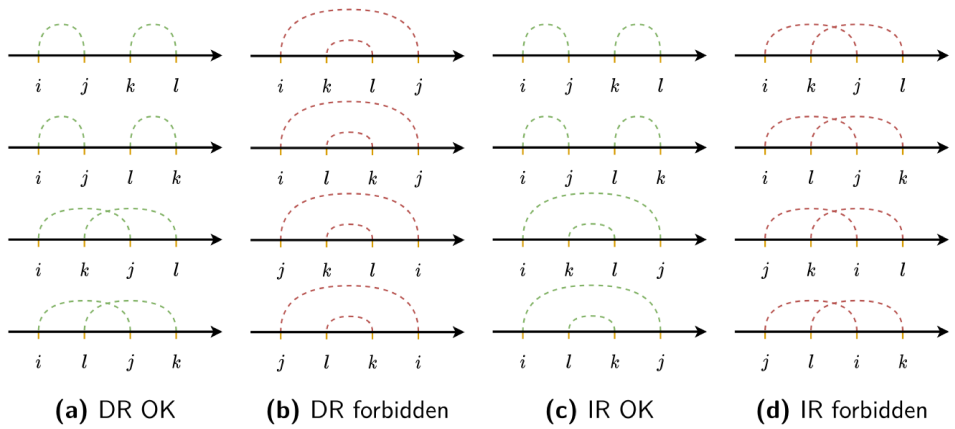


Figure 6 Non-exhaustive illustrations for authorised and forbidden order cases for two repeated fragments $((i,j),(k,l)) \in PRepF$. **(a)** and **(b)** Authorised and forbidden orders for $PDirF$; **(c)** and **(d)** Authorised and forbidden orders for $PInvF$.

$$3forbid_{ijkl} \leq \alpha_{ij} + \alpha_{jk} + \alpha_{kl} \quad \text{(C14)}$$
$$2 + forbid_{ijkl} \geq \alpha_{ij} + \alpha_{jk} + \alpha_{kl} \quad \text{(C15)}$$

$$\forall (p,q) \in PRepF :$$
$$m_p + m_q \leq 2 - \sum_{\substack{(a,b,c,d) \\ \in Forbid \\ \text{s.t. } (p,q)}} forbid_{abcd} \quad \text{(C16)}$$

$$\forall (u,v) \in ARepF :$$
$$isadj_{uv} \leq x_{uv} \quad \text{(C17)}$$
$$isadj_{uv} \leq x_{\texttt{repadj}(u,v)} \quad \text{(C18)}$$
$$isadj_{uv} \leq m_{\texttt{repfrag}(u)} \quad \text{(C19)}$$
$$isadj_{uv} \leq m_{\texttt{repfrag}(v)} \quad \text{(C20)}$$

$$m_p \in \{0;1\} \quad \forall p \in RepF$$
$$isadj_e \in \{0;1\} \quad \forall e \in ARepF$$
$$forbid_{ijkl} \in \{0;1\} \quad \forall (i,j,k,l) \in Forbid$$
$$\alpha_{uv} \in \{0;1\} \quad \forall (u,v) \in Alpha$$

Where $\forall v \in V, pos(v) = \sum_{w \in N_v^+} f_w$. Note that $\alpha_{uv} = 1 - \alpha_{vu}$, thus even if $(v,u) \notin Alpha$, for a sake of clarity we write α_{vu} instead of $1 - \alpha_{uv}$.

Constraints C9 and C10: a repeated fragment participate in a repeat if its vertices are participating in the circuit. Constraints C11 to C13 define the α_{uv} variables. Constraints C14 to C16 define the forbidden orders for a pair of repeated fragments. Constraints C17 to C20 define the adjacency for two adjacent repeated fragments.

5.3 Fixing regions constraints

When repeats were already scaffolded, the regions have to be fixed for the next problems to solve. Let $ADirF^*$, $DirF^*$, $AInvF^*$ and $InvF^*$ respectively be the sets of (adjacent) direct and (adjacent) inverted fragments composing the direct and inverted repeats that have been scaffolded.

CFixRegions ***constraints***

$$\forall (u,v) \in DirF^* \cup InvF^* :$$
$$i_u = 1 \quad \text{(C21)}$$
$$i_v = 1 \quad \text{(C22)}$$

$$\forall (u,v) \in ADirF^* \cup AInvF^* :$$
$$x_{uv} = 1 \quad \text{(C23)}$$

$$x_{\texttt{diradj}(u,v)} = 1 \quad \forall (u,v) \in ADirF^* \quad \text{(C24)}$$
$$x_{\texttt{invadj}(u,v)} = 1 \quad \forall (u,v) \in AInvF^* \quad \text{(C25)}$$

5.4 Speed-up constraints

Constraints C26 and C27 prevent the solver to loop on strictly equivalent solutions e.g. solutions that differ according to a permutation of the occurrences. Denote by $ConsOcc$ the set of occurrence-consecutive vertices, such that:

$$ConsOcc = \left\{ (u,v) \in V^2 \;\middle|\; \begin{array}{l} u_{id} = v_{id} \\ \wedge\, u_{or} = v_{or} = 0 \\ \wedge\, u_{occ} = v_{occ} - 1 \end{array} \right\}$$

Also, denote by $ConsRepF$ the set of consecutive repeated fragments, such that:

$$ConsRepF = \left\{ \begin{array}{l} ((i,j),(k,l)) \in RepF^2 \text{ s.t.} \\ \quad i_{id} = j_{id} = k_{id} = l_{id} \\ \quad \wedge\, i_{or} = k_{or} \wedge j_{or} = l_{or} \\ \quad \wedge\, i_{occ} = k_{occ} - 2 \\ \quad \wedge\, j_{occ} = l_{occ} - 2 \end{array} \right\}$$

$$i_v + i_{\overline{v}} \leq i_u + i_{\overline{u}} \quad \forall (u,v) \in ConsOcc \qquad \text{(C26)}$$
$$m_q \leq m_p \quad \forall (p,q) \in ConsRepF \qquad \text{(C27)}$$

5.5 Scaffolding problems ILP

Finally, it is possible to define the ILP formulations for the $\mathcal{DRP}$, $\mathcal{IRP}$ and $\mathcal{SCP}$ scaffolding problems as a combination of the constraints described before. For $\mathcal{DRP}$ and $\mathcal{IRP}$ the ILP formulations are the same, and it is sufficient to choose the sets $RepF$, $PRepF$, $ARepF$, $Alpha$, $Forbid$ and $ConsRepF$ according to the repeats the problems scaffold.

$\mathcal{DRP}$ and $\mathcal{IRP}$ models

$$\begin{array}{lll} \max & \displaystyle\sum_{p \in PRepF} m_p + \sum_{e \in ARepF} isadj_e & \\ \text{s.t.} & CCircuit & \\ & CRepeat & \\ & \text{(C26)} & \\ & \text{(C27)} & \text{if no repeats to fix} \\ & CFixRegions & \text{otherwise} \end{array}$$

The first term in the objective corresponds to searching for the maximum number of repeats and the second one guaranties that they are the longest.

$\mathcal{SCP}$ model

$$\begin{array}{lll} \max & \displaystyle\sum_{v \in V \setminus \{s;\overline{s}\}} \mathsf{v_{prob}} i_v & \\ \text{s.t.} & CCircuit & \\ & \text{(C26)} & \\ & CFixRegions & \text{if repeats to fix} \end{array}$$

While the two repeat problems $\mathcal{DRP}$ and $\mathcal{IRP}$ find a feasible circuit to connect the repeats, here we search for the maximum-weighted existence probabilities paths connecting them.

Both for $\mathcal{DRP}$ and $\mathcal{IRP}$, the number of variables and constraints are in $O(|V|^2 + |E|)$. The number of variables and constraints for $\mathcal{SCP}$ are in $O(|V| + |E|)$.

6 Multiple genome forms

From a solution found by an ILP, we extract the corresponding genome architecture. Two items are sufficient to describe it:

— a linearised circular sequence SOR of oriented regions i.e. a sequence of couples (r_{id}, r_{or}) where $r_{id} \in \mathbb{N}$ and $r_{or} \in \{0;1\}$;
— each region considered in its forward orientation is associated with a sequence of oriented contigs.

The key idea of such an extraction algorithm is to start from the starting vertex s and walk over the chosen edges. During the walk, for each vertex we need to identify the type of its region, and then add the vertex to the corresponding region (to the corresponding region identifier r_{id}). Algorithm 1 aims to determine the region type for a given vertex.

In the following, the given time complexities are under the assumption that the belonging test "*is $o \in S$?*" for an object o in a set S is in $\Theta(1)$.

Algorithm 1 Get the region type for a given vertex

Require: A vertex v, the chosen direct fragments $DirF^*$ and the chosen inverted fragments $InvF^*$.
Ensure: Returns the region type of the vertex.

```
1: function GET_REGION_CODE(v)
2:     if dirfrag(v) ∈ DirF* then
3:         return DR
4:     if invfrag(v) ∈ InvF* then
5:         return IR
6:     return SC
```

GET_REGION_CODE function is in $\Theta(1)$. To initialise the region extraction, we will begin by exploring the solution path from the first vertex that belongs to the single-copy region containing the starting vertex. This first vertex is defined as

the *initial vertex* and can be the starting vertex itself. Finding the initial vertex is the purpose of the INITIAL_VERTEX function.

Algorithm 2 Get the initial vertex of the single-copy region containing the starting vertex

Require: The starting vertex s, the set of fixed variable values $\mathsf{x}^*_e, e \in E$, the chosen direct fragments $DirF^*$ and the chosen inverted fragments $InvF^*$.

Ensure: Returns the first vertex of the single-copy region containing the starting vertex.

```
1:  function INITIAL_VERTEX( )
2:      v ← s
3:      u ← u | x*_us = 1
4:      reg_code ← GET_REGION_CODE(u)
5:      while u ≠ s ∧ reg_code = SC do
6:          v ← u
7:          u ← u | x*_uv = 1
8:          reg_code ← GET_REGION_CODE(u)
9:      if region_code = SC then ▷ Special case of a unique single-copy region
10:         return s
11:     return v
```

The INITIAL_VERTEX function is in $O(|SC_s|)$, where $|SC_s|$ is the number of contigs composing the single-copy region that contains the staring vertex.

It is now possible to write Algorithm 4 that extracts the regions. The REPEAT_IS_CONTIGUOUS function is in $O(1)$, and so the PATH_TO_REGIONS function is in $O(|V| + |E|)$.

The sequence of oriented regions SOR represents one chloroplast genome form. Remember that especially with the LSC-IR-SSC-IR architecture (see Figure 1 (a)), the SSC can be reversed during the DNA replication phase. In the following, our goal is to retrieve other forms from the one obtained by the hierarchical problem succession. Towards this goal, we introduce a specific assembly graph: the *region graph*.

Definition 25 (Region graph $RegGraph$)**.** *Given the sequence of oriented regions SOR, denote by $RegGraph = (Vreg, Ereg)$ a directed multigraph called the region graph, such that:*

Algorithm 3 Is the repeat contiguous?

Require: Two vertices $u, v \in V$, the set of fixed variable values $\mathsf{x}^*_e, e \in E$, the previous and current region codes $prev\_region\_code$ and $region\_code$.

Ensure: Returns true if the repeat is contiguous, else false.

```
1: function REPEAT_IS_CONTIGUOUS(u, v, prev_region_code, region_code)
2:     if prev_region_code ≠ region_code then
3:         return False
4:     if region_code = DR then
5:         return x*_diradj(u,v) = 1
6:     if region_code = IR then
7:         return x*_invadj(u,v) = 1
8:     return False
```

— *$Vreg$ is the set of oriented region. Each $r \in Vreg$ has an identifier attribute $r_{id} \in \mathbb{N}$ and has an orientation $r_{or} \in \{0; 1\}$*

— *$Ereg$ is the set of links between two oriented regions. Two consecutive oriented regions r_i, r_{i+1} in SOR (the last and the first regions too, as the sequence is circular) create two edges in the graph: (r_i, r_{i+1}) and $(\overline{r_{i+1}}, \overline{r_i})$.*

Figure 7 illustrates the region graph for the data given in Table 1.

Based on the above graph, we can find different sequences of oriented regions. Each region, independently of its orientation, in each sequence, must participate the same number of times.

Definition 26 (Eulerian circuit in $RegGraph$)**.** *A circuit in $RegGraph = (Vreg, Ereg)$ is defined as eulerian when:*

— *it begins from and ends with the region containing the starter in forward orientation (region 0_f)*

— *it passes through exactly one of the two versions of each edge ($(r_i, r_{i+1}) \in Ereg$ or $(\overline{r_{i+1}}, \overline{r_i}) \in Ereg$)*

Proposition 1 (An eulerian circuit is a valid oriented contigs sequence for $RegGraph$)**.** *Given an eulerian circuit in $RegGraph = (Vreg, Ereg)$ it respects Definition 1.*

Proof. Let $RegGraph = (Vreg, Ereg)$ be a region graph and let $euc = r_0 r_1 \cdots r_{n-1}$ be an eulerian

Algorithm 4 Path to regions

Require: Graph $MDCG$, the starting vertex s, the set of fixed variable values $\mathsf{x}^*_\mathsf{e}, e \in E$, the chosen direct fragments $DirF^*$ and the chosen inverted fragments $InvF^*$.

Ensure: Returns the oriented region sequence SOR, and the oriented contig sequence for each forward oriented region COR.

1: **function** PATH_TO_REGIONS( )
2: $prev\_region\_code \leftarrow \mathsf{SC}$
3: $SOR \leftarrow [0_f]$ ▷ *Oriented region sequence, initialised by the single-copy in forward orientation that contains the starting vertex*
4: $COR \leftarrow [\,[\,]\,]$ ▷ *Oriented contig sequence for each forward region, initialised for the first region*
5: $region\_index \leftarrow 0$ ▷ *First region index*
6: $\mathsf{init\_v} \leftarrow$ INITIAL_VERTEX()
7: $u \leftarrow \mathsf{init\_v}; v \leftarrow \mathsf{init\_v}$
8: $rep\_queue \leftarrow$ HASHTABLE() ▷ *For each repeat index is associated a queue of vertices (FIFO for DR, LIFO for IR)*
9: $repf\_rep \leftarrow$ HASHTABLE() ▷ *Each repeated fragment is associated with its repeat index*
10: **repeat**
11: $region\_code \leftarrow$ GET_REGION_CODE(v)
12: **if** $region\_code = \mathsf{SC}$ **then**
13: **if** $prev\_region\_code \neq region\_code$ **then** ▷ *New single-copy*
14: $region\_index \leftarrow |COR|$
15: SOR.APPEND($region\_index_f$)
16: COR.APPEND([ ])
17: $COR[region\_index]$.APPEND(v_{id}, v_{or})
18: **else**
19: $repf \leftarrow \mathtt{dirfrag}(v)$ if $region\_code = \mathsf{DR}$ else $\mathtt{invfrag}(v)$
20: **if** $repf \notin repf\_rep$ **then** ▷ *Fist region of the repeat*
21: **if** ¬REPEAT_IS_CONTIGUOUS(u, v, $prev\_region\_code$, $region\_code$) **then**
22: $region\_index \leftarrow |COR|$
23: SOR.APPEND($region\_index_f$)
24: COR.APPEND([ ])
25: $rep\_queue[region\_index] \leftarrow$ FIFO() if $region\_code = \mathsf{DR}$ else LIFO()
26: $COR[region\_index]$.APPEND(v_{id}, v_{or})
27: rep_queue.PUT($\mathtt{dirfrag}(v)$ if $region\_code = \mathsf{DR}$ else $\mathtt{invfrag}(v)$)
28: $repf\_rep[repf] \leftarrow region\_index$
29: **else** ▷ *Second region of the repeat*
30: **if** ¬REPEAT_IS_CONTIGUOUS(u, v, $prev\_region\_code$, $region\_code$) **then**
31: $region\_index \leftarrow repf\_rep[repf]$
32: SOR.APPEND($region\_index_f$ if $region\_code = \mathsf{DR}$ else $region\_index_r$)
33: **assert** $v = rep\_queue[region\_index]$.GET()
34: $prev\_region\_code \leftarrow region\_code$
35: $u \leftarrow v$
36: $v \leftarrow v \mid \mathsf{x}^*_{\mathsf{uv}} = 1$
37: **until** $v \neq \mathsf{init\_v}$
38: **return** SOR, COR

circuit in *RegGraph*. For each (r_i, r_{i+1}) two consecutive oriented regions in *euc*, by construction there is an edge $(r_i, r_{i+1}) \in Ereg$. According to Definition 25, r_i and r_{i+1} are also consecutive in the oriented region sequence that has originally built *RegGraph*. Thus, according to Algorithm 4, there is an edge $(u, v) \in E$ in the $MDCG$ graph such that, for c the last oriented contig in r_i and d the first oriented contig in r_{i+1}, $c_{id} = u_{id}, c_{or} = u_{or}$ and $d_{id} = v_{id}, d_{or} = v_{or}$. Note that the number of times a contig occurs (independently of its orientations) is upper bounded by its multiplicity by construction of $MDCG$. □

We can easily verify that the number of eulerian circuits is bounded by $O(2^n)$, where n is the number of inverted repeats.

Now, given a region graph *RegGraph*, finding all the eulerian circuits is equivalent to retrieve all the possible chloroplast genome forms. Each eulerian circuit traverses exactly the same regions, but not necessary in the same orientations. Figure 7 gives the resulting region graph obtained from the input data given in Table 1.

Although an eulerian circuit can contradict the repeated region interval constraints given in Definitions 11 and 12, we accept it. As an example, consider the oriented region sequence $0 \to 1 \to 2 \to 3 \to 1 \to \overline{3}$ that respects the definitions (region 1 is a DR, region 3 is an IR). The eulerian circuit can produce another oriented region sequence $0 \to 1 \to 2 \to 3 \to \overline{1} \to \overline{3}$ that contradicts Definition 12 (note that region 1 becomes an IR).

As described in the introduction and Figure 2, we interpret the eulerian circuits obtained from one region graph as the multimeric forms for one circular chloroplast genome.

7 Hierarchical problem succession

Finally, here we describe how we combine the $\mathcal{DRP}$, $\mathcal{IRP}$ and $\mathcal{SCP}$ scaffolding problems. As described in Definition 10, two problem combinations are opposed. The combinations are kept depending on the value of the problems' objective functions.

Definition 27 (Hierarchical problem succession solutions)**.** *Let denote by h_1, h_2 respectively*

Figure 7 The input data given in Table 1 producing the graph visualised in Figure 3 will give a region graph that shows an LSC-IR-SSC-IR architecture as the one given in Figure 1 (a). Two oriented region sequences (genome forms) are obtained by finding the eulerian circuits: $0_f \to 1_f \to 2_f \to 1_r$ and $0_f \to 1_f \to 2_r \to 1_r$. They correspond to two multimeric forms that are commonly found in the chloroplast cells, and described in Figure 2. **Left (artistic view):** Each arrow represents a region. Each link connects two arrows' extremity. Entering the tail/head and exiting the head/tail of an arrow resp. corresponds to choosing the region in its forward/reverse orientation. **Right (directed multigraph view):** The same information is visualised. Each vertex is a region with a fixed orientation, and each edge connects two oriented regions.

the two hierarchical problem successions $\mathcal{DRP}$–$\mathcal{IRP}$–$\mathcal{SCP}$ and $\mathcal{IRP}$–$\mathcal{DRP}$–$\mathcal{SCP}$. For each $h \in \{h_1; h_2\}$, let denote by $Z_h^* \in \mathbb{R}^3$ the vector containing the values of the objective functions for each problem in the order of the problem succession corresponding to h.

By S we denote the set of kept problem successions, such that:

$$S = \left\{ s \;\middle|\; \forall k \in [\![0; 2]\!], Z_s^*[k] = \max_{h \in \{h_1; h_2\}} Z_h^*[k] \right\}$$

Note that $0 \leq |S| \leq 2$, and Definition 27 is stable for any problem with an objective value equal to zero. For example, $Z_{h_1}^*[1] = 0$ means that there is no inverted repeat. For an easier interpretation of the architecture, we adopt a *problem code combination*, summarised in Table 3.

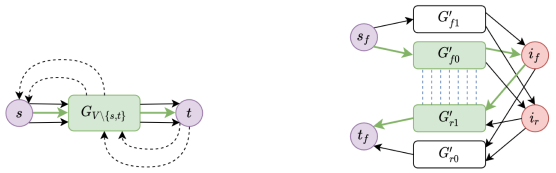
Table 3 Problem code combinations

Z_h^*			h_1	h_2
[0]	[1]	[2]	$\mathcal{DRP}$–$\mathcal{IRP}$–$\mathcal{SCP}$	$\mathcal{IRP}$–$\mathcal{DRP}$–$\mathcal{SCP}$
0	0	–	`sc`	`sc`
> 0	0	–	`dr–sc`	`ir–sc`
> 0	> 0	–	`dr–ir–sc`	`ir–dr–sc`

8 $\mathcal{NP}$-completeness

To prove the $\mathcal{NP}$-completeness of one of the two hierarchical problem successions, it is sufficient to focus on only one scaffolding problem for the repeat. Here, we will focus on $\mathcal{IRP}$.

Proposition 2. *$\mathcal{IRP}$ is $\mathcal{NP}$-Hard.*



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

Figure 8 Transformation of a directed graph G for $LPSTP$ to a directed graph G' for $\mathcal{IRP}$. Edges in bold-green in both sub-figures correspond to the solution path for $LPSTP$ and $\mathcal{IRP}$ 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 inverted fragments.

Proof. By reduction from the longest path problem from vertex s to vertex t ($LPSTP$), known to be $\mathcal{NP}$-Hard (Schrijver, 2003).

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 $\mathbb{I} \in LPSTP \iff mdgf(\mathbb{I}) \in \mathcal{IRP}$. $\mathbb{I}$ contains a directed graph $G = (V, E)$, two vertices $s \in V$ and $t \in V$. Function $mdgf$ transforms graph G to a graph $G' = (V', E')$, and $s \in V$ vertex to $s_f \in V'$ and $t_f \in V'$ vertices. As illustrated in Figure 8, function $mdgf$ transforms graph G in (a) to a graph $G' = (V', E')$ in (b), vertex $s \in V$ in (a) to vertices $s_f \in V'$ and $t_f \in V'$ in (b), while vertices i_f and i_r in (b), are images of t in (a).

All sub-graphs $G'_{\_}$ in Figure 8 (b) can be seen as copies of $G_{V\setminus\{s,t\}}$ in Figure 8 (a). 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 built 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 visualised by blue dashed vertical lines from G'_{f0} to G'_{r1} in Figure 8. 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 an algorithm in $O(|V| + |E|)$ that computes this transform function.

As adjacent inverted fragments associate only vertices in V_{f0} with those in V_{r1}, the path that maximises 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 find both 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 which equals the length of the longest path in G.

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

□

Proposition 3 (Solution verification is done in polynomial time)**.** *Given a graph $MDCG = (V, E)$, a starting vertex s, fixed values of $\mathsf{x}^*_e, e \in E$ variables, the chosen direct fragments $DirF^*$ and the chosen inverted fragments $InvF^*$. There is a polynomial time algorithm that checks if these values satisfy the conditions for feasible solution.*

Proof. We can slightly modify Algorithm 4 to do the job. It is sufficient to:

- Verify if the algorithm does not crash (especially during the assertion Line 33)
- At the end, verify if all the queues are empty i.e.

$$\forall\, queue \in rep\_queue$$
$$\textbf{assert}\ queue.\text{IS\_EMPTY}()$$

Finally, Algorithm 4 is linear according to the number of vertices and the number of repeated fragments if we assume the belonging test on

sets is $\Theta(1)$. It remains linear with the above modifications. □

From Propositions 2 and 3 we conclude that $\mathcal{IRP}$ is $\mathcal{NP}$-Complete.

9 Numerical results

We develop `khloraascaf`[1], a python package that computes the scaffolding of chloroplast contigs. It can either use `Gurobi` solver or `CBC`. All the following runs have been executed on a Linux laptop computer (32GB RAM, Intel® Core™ i7–10610U CPU @ 1.80GHz ×8). Each time, `Gurobi` solver is selected.

9.1 Complexity validation on artificial data

`khloraascaf` is also accessible as an API, that permits in this section to study the combinatorial behaviour of $\mathcal{IRP}$.

We demonstrate in Section 8 that $\mathcal{IRP}$ is in the general case $\mathcal{NP}$-Complete. Furthermore, the multimeric forms of the chloroplast genome is very often caused by the presence of an inverted repeat, that reverses the region(s) between it.

Thus, we artificially build a contig set with the associated attributes, and a link set, such that the genome architecture behind corresponds to the following circular sequence of oriented regions: $SC1 - IR - SC2 - \overline{IR}$. In the following, we run what corresponds to $\mathcal{IRP}$ computation in `khloraascaf` on two types of growing generated data: perfect and noisy artificial ones. To emphasise the effect of the inverted repeats in the complexity of $\mathcal{IRP}$, we fix the length of the single-copy regions i.e. $|SC1| = |SC2| = |SC| = 20$, and we incrementally raise the length of the inverted repeat $|IR| = 20k$ for $k \in [\![1; 10]\!]$.

9.1.1 Perfect artificial data

The data generated for this section correspond to the smallest set of contigs, links, and the smallest multiplicities to make sure that $\mathcal{IRP}$ is feasible. The multiplied doubled contig graph associated with these perfect data has exactly the same topology as the one illustrated in Figure 3. For instance, testing perfect artificial data acts as a control for further tests. Table 4 gives some `Gurobi` metrics[2].

Observe that the gap is equal to 0% and the problem is solved either during the presolving or the linear relaxation. Indeed, for the class of graph that contains only the perfect artificial data, there exist a polynomial algorithm. However, the relaxation time seems to fit an exponential distribution as well as for the B&B time, as shown in Figure 9, even though the distributions should be treated with caution because of the limited number of points.

9.1.2 Noisy artificial data

Here we test the behaviour of the solver when we add noise to the perfect artificial data. In that case, for each generated contig, the multiplicity has 25% chance to be overestimated by one (that increases $InvF$, $PInvF$ and possibly $AInvF$ sets). Similarly, for each contig, there is 25% of chance to create a new link to another randomly chosen contig. This can generate more loops, and can increase the $AInvF$ set[3].

As expected, now the gap is not ensured to be null, and some instances are not solved at the presolving or at the linear relaxation steps.

These numerical results corroborate the $\mathcal{NP}$-Complete demonstration for a more general class of graphs.

9.2 Synthetic chloroplast input data

In this section, we aim to validate experimentally the relevance of our scaffolding problem definition by running `khloraascaf` on synthetic data.

9.2.1 Input data generation

Here we briefly describe our protocol for input data generation[4]. 200 chloroplast genomes (selected in CpGDB[5]) were downloaded from the NCBI[6]. For each of them, a set of reads

[1] https://khloraa-scaffolding.readthedocs.io/en/latest

[2] To reproduce the results, please refer to https://khloraascaf-results.readthedocs.io/en/latest/benchmark_4/.

[3] To reproduce the results, please refer to https://khloraascaf-results.readthedocs.io/en/latest/benchmark_5/.

[4] For more details, please refer to https://khloraascaf-results.readthedocs.io/en/latest/benchmark_3/

[5] CpGDB: A Comprehensive Database of Chloroplast Genomes http://www.gndu.ac.in/CpGDB/index.aspx

[6] NCBI: The National Center for Biotechnology Information advances science and health by providing access to biomedical and genomic information https://www.ncbi.nlm.nih.gov/

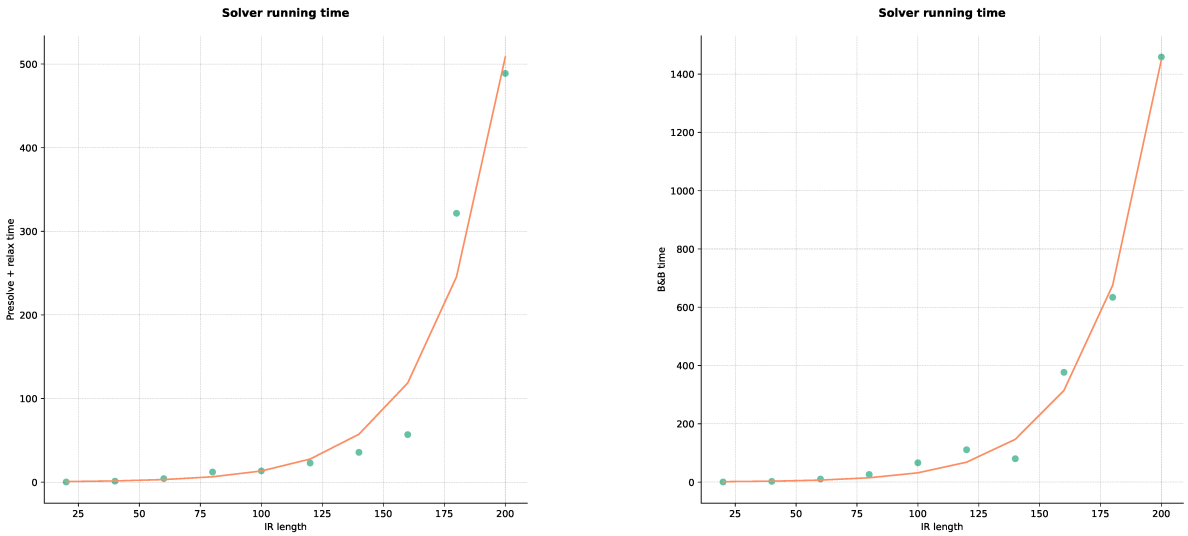


Figure 9 Solver running time distributions for perfect artificial data. Points are measured times, the red curves correspond to the best ae^{bx} functions applied on IR length axis.

Table 4 `Gurobi` solver metrics on perfect artificial growing data. |V|, |E|, |SC|, |IR| and $|\mathcal{L}|$ respectively stand for the number of vertices, edges, contigs in each single-copy region, contigs in each region of the inverted repeat, and links; **Time**: the presolve time plus the relaxation time (above) and the B&B time (below); **Opt.**: the linear relaxation bound UB (above) and the integer optimal value Opt (below); **% Gap**: the MIP gap equals $100 \times \frac{(UB-Opt)}{UB}$; **Nodes**: number of explored B&B nodes; **Iter.**: number of iterations for the LP relaxation (above) and for the B&B phase (below).

|V|	|E|	|SC|	|IR|	$\|\mathcal{L}\|$	**Time**	**Opt.**	**% Gap**	**Nodes**	**Iter.**
160	244	20	20	122	0.25	39.00	0.00	1	1242
					0.48	39.00			2944
240	404	20	40	162	1.23	79.00	0.00	1	4690
					2.54	79.00			10 946
320	564	20	60	202	4.27	119.00	0.00	1	8945
					10.45	119.00			23 864
400	724	20	80	242	12.15	159.00	0.00	1	15 196
					26.29	159.00			39 562
480	884	20	100	282	13.46	199.00	0.00	1	23 109
					66.28	199.00			68 740
560	1044	20	120	322	22.91	239.00	0.00	1	44 048
					110.98	239.00			97 646
640	1204	20	140	362	35.68	279.00	0.00	1	46 071
					80.26	279.00			125 084
720	1364	20	160	402	56.89	319.00	0.00	1	75 371
					376.74	319.00			256 831
800	1524	20	180	442	321.52	359.00	0.00	1	71 971
					634.01	359.00			196 905
880	1684	20	200	482	488.74	399.00	0.00	1	88 157
					1458.66	399.00			236 406

was generated. The contigs were generated with `Minia` (Chikhi and Rizk, 2012), a de Bruijn graph assembly approach. The links correspond to k-mer paths in the resulting compacted de Bruijn graph (cDBG) that connect two contigs. Finally, 31 instances were selected for which various difficulties have been detected e.g. extra-links in the link set or combination of repeats.

The starter is the contig for which the matK gene, usually found in a single-copy region, maps on.

To obtain the multiplicity of a given contig c, we sum the length of the alignments of the reads mapping on c (we denote by $MapR_c$ the set containing the reads that map on c). This sum is defined as the coverage c_{cov} of the contig c by the

Table 5 `Gurobi` solver metrics on noisy artificial growing data. The column descriptions are the same as the one in Table 4. Except that because of the noise on multiplicities, the sum of contig multiplicity for each region can change: this is the value below the number of contig in columns |SC1|, |SC2| and |IR|. Similarly, because of the noise on the number of links, the below value for $\mathcal{L}$ corresponds to the number of noisy links.

|V|	|E|	|SC1|	|SC2|	|IR|	$\lvert\mathcal{L}\rvert$	Time	Opt.	% Gap	Nodes	Iter.
186	372	20	20	20	152	1.01	39.00	0.00	1	2862
		26	24	43	30	5.16	39.00			6888
280	688	20	20	40	212	6.89	79.00	0.00	1	6727
		28	23	89	50	12.26	79.00			19 930
366	946	20	20	60	262	42.52	123.50	3.64	1	16 821
		25	26	132	60	79.28	119.00			50 246
452	1208	20	20	80	320	90.06	161.50	1.55	1	27 822
		22	23	181	78	295.22	159.00			129 174
556	1366	20	20	100	322	196.04	199.00	0.00	1	42 292
		23	24	231	40	244.41	199.00			92 009
662	1804	20	20	120	412	1007.59	242.50	1.44	1	84 639
		29	26	276	90	1434.16	239.00			210 798
736	1946	20	20	140	454	1108.09	283.00	1.41	1	228 198
		24	26	318	92	3540.79	279.00			691 619
822	2212	20	20	160	514	1118.76	323.00	1.24	1	86 592
		26	27	358	112	2449.55	319.00			287 146
902	2362	20	20	180	542	1591.18	363.00	1.10	1	91 936
		26	27	398	100	2576.60	359.00			269 958
996	2656	20	20	200	602	2294.85	404.00	1.24	1	116 315
		26	26	446	120	3747.02	399.00			351 501

reads:

$$c_{cov} = \sum_{read \in MapR_c} \left| align_c^{read} \right|$$

Where $align_c^{read}$ is the sequence representing the alignment of the read $read$ on the contig c. Its length equals the number of nucleotides of $read$ that match on c (identity or substitution). Then the multiplicity c_{mult} of c is obtained by normalising its coverage c_{cov} by this of the starter s, s_{cov}: $c_{mult} = \max(1, \lceil c_{cov}/s_{cov} - 0.1 \rceil)$. As the multiplicity is an upper-bound of the usage of contig, we prefer to round up the normalisation only if the decimal part is greater than 0.1.

The existence-probability c_{prob} for a contig c is computed by counting the number of nucleotides of c that are covered by at least a gene of protein from a chloroplast near-species, normalised by the length of c.

9.2.2 The evaluation's metrics

For each synthetic instance, we know the sequence of the oriented contigs and the sequence of the oriented regions we search for. In the sequel we test our scaffolding approach and evaluate the obtained region graph. For each instance, for each optimisation problem combination, we provide the following metrics:

— the total number of eulerian circuits in the region graph (genome forms);
— how many of them coincide with the sequence of oriented contigs we search for;
— how many of them coincide with the sequence of oriented regions we search for.

Evaluating the sequence of the oriented contigs is stringent. On the other hand, although a result can be evaluated as a false one, we can still retrieve the sequence of the chloroplast genome by applying an alternative sequence. As a consequence, for each instance we use `Quast` (Gurevich et al, 2013) to evaluate the sequences associated to each genome form. As the genome reference is known, `Quast` tries to find the minimum number of differences (relocation, inversion, indels) between the reference and the sequence we provide. Three metrics are chosen to evaluate the best genome form for each problem succession:

— the genome fraction of the reference;
— the number of misassemblies;
— the number of local misassemblies.

For more detailed descriptions of `Quast` metrics, you can refer to Section C.1.

It would be expected that `Quast` metrics illustrate wrong assembly for the instances for which the sequence of the contigs, and a fortiori this of the regions, are not retrieved. Analogously, the instance that truthfully retrieves the sequences would have good `Quast` metric. However, these assertions may be contradicted because of the contig and the link sequences generation.

In Section 9.2.3, we provide the two metric sets when `khloraascaf` is applied to the original synthetic data, while in Section 9.2.4 the metrics are reported for a subset of modified synthetic data.

9.2.3 Initial version

Table 6 provides all the metrics defined above for the 31 instances. The instances are solved very quickly (solver times < 4.5 sec., Table D2). `khloraascaf` successfully founds the sequence of the oriented contigs in 20 of them and retrieves the sequence of the oriented regions in 28 of them[7]. Three categories of failures are identified.

Wrong starting contig and multiplicities estimations

This is the category for which our approach is dependent and sensible. In the presented version, we have used a given starting contig, and a wrong one can lead to reduce all the multiplicities. This is the case for Begonia_pulchrifolia and Lamprocapnos_spectabilis for which the starters are contigs that normally participate into an IR, that contradicts our assumption that the starter participates in an SC.

Independently of a right starting contig, the multiplicity computation is sensible from the noise on the contig coverage by the reads. Agathis_dammara and Pelargonium_nanum both suffer from only one contig under-estimated multiplicity.

$\mathcal{IRP}$'s objective: a maximum of the longest repeats

The sequence of oriented regions for the Cucumix_hystrix's reference contains the following subsequences: $IR - IR' \cdots \overline{IR'} - SC - \overline{IR}$. As $\mathcal{IRP}$ is defined as finding a maximum number of the longest repeats, the "find the longest" part merges $IR - IR'$ and consequently not retrieves SC, normally inserted between $\overline{IR'}$ and $\overline{IR}$.

Model's robustness

While the sequences of oriented contigs for the repeats are retrieved, the ones of the single-copy regions suffer from extra-links combined with low, sometimes null, existence-probabilities. On the one hand, in case of null existence-probabilities, the circuits in Lathyrus_pubescens and Triosteum_pinnatifidum do not pass through contigs that must participate in the SCs. On the other hand, Podocarpus_totara possesses two objective-equivalent sub-paths in an SC.

Surprisingly, our tool `khloraascaf` reversed some subparts of single-copies. This is due to extra-links, but they are specifically caused by the existence of very short IRs hidden in them (remind that the links correspond to paths between pairs of contigs in the cDBG). This behaviour is observed in Carpodetus_serratus, Jasminum_tortuosum and Lophocereus_schottii.

Nucleotide sequences misassemblies

To avoid confusion, we always use nucl. seq. to denote "nucleotide sequence" to contrast with sequences of contigs / regions. In the following we analyse the instances presenting an unexpected behaviour for the `Quast` metrics, regarding if the sequences are found. Supplementary Table D1 permits verifying if the contigs that participate in the sequence have been correctly assembled. Except for Lathyrus_pubescens where one local misassembly is due to the missing contig in the sequence, all the (local) missassemblies in Commiphora_foliacea, Eucommia_ulmoides, Juniperus_scopulorum, Musa_ornata, Sciadopitys_verticillata, Taxus_baccata, Welwitschia_mirabilis provided by `Quast` are found in the nucl. seq. of links.

9.2.4 Modified version

In this section, we present a manually changed synthetic data version to succeed the scaffolding. The goal is to precisely evaluate the robustness of `khloraascaf`. We detail the modifications bellow[8]:

Agathis_dammara The multiplicity of contig 1 is raised to be equal to 3.

[7] For more details on the result for the initial version of the synthetic data, please refer to https://khloraascaf-results.readthedocs.io/en/latest/benchmark_3/v1_order_analysis/ and https://khloraascaf-results.readthedocs.io/en/latest/benchmark_3/v1_quast_analysis/.

[8] For more details, please refer to https://khloraascaf-results.readthedocs.io/en/latest/benchmark_3/synthetic_data_v2/.

Table 6 Sequence and `Quast` metrics for the initial synthetic data version. For each instance in the column **Instance**: **ILPs** provides the optimal (at most two) hierarchical problem successions; **Total** reports the number of eulerian circuits in the region graph (genome forms); SOC is the number of oriented contig sequences that equal to the reference oriented contig sequence; SOR is the number of oriented region sequences in bijection with the reference oriented region sequence; **%gnm** is the genome fraction of the best sequence produced by one of the genome forms; **#mis** are the number of misassemblies (left) and of local misassemblies (right).

			Successful		Quast		
Instance	**ILPs**	**Total**	SOC	SOR	**%gnm**	**#mis**	
Abies_alba	`dr-sc`	1	0	0	99.88	3	0
	`ir-sc`	2	1	2	100.00	0	0
Acorus_americanus	`ir-sc`	2	1	2	99.99	0	0
Agathis_dammara	`dr-ir-sc`	10	0	2	80.51	1	2
	`ir-dr-sc`	6	0	0	80.94	2	2
Azima_tetracantha	`ir-sc`	2	1	2	99.99	0	0
Begonia_pulchrifolia	—	—	—	—	—	—	—
Carpodetus_serratus	`ir-sc`	2	0	2	100.00	2	0
Circaeaster_agrestis	`ir-sc`	2	1	2	99.99	0	0
Clematis_repens	`ir-sc`	2	1	2	100.00	0	0
Commiphora_foliacea	`ir-sc`	4	1	4	98.82	0	5
Cucumis_hystrix	`ir-sc`	2	0	0	100.00	0	0
Eucommia_ulmoides	`ir-sc`	2	1	2	99.99	0	2
Jasminum_tortuosum	`ir-sc`	2	0	2	99.99	2	0
Juniperus_scopulorum	`sc`	1	1	1	99.72	0	1
Lamprocapnos_spectabilis	`dr-sc`	1	0	0	35.86	3	0
Lathyrus_pubescens	`dr-sc`	1	0	0	98.47	2	2
	`ir-sc`	2	0	2	98.50	0	2
Lophocereus_schottii	`sc`	1	0	1	99.88	1	0
Musa_ornata	`ir-sc`	2	1	2	99.97	0	2
Oenothera_glazioviana	`ir-sc`	2	1	2	100.00	0	0
Pelargonium_nanum	`ir-sc`	2	0	2	96.44	4	0
Podocarpus_totara	`sc`	1	0	1	99.71	4	0
Porphyra_purpura	`dr-sc`	1	1	1	100.00	0	0
Sagittaria_trifolia	`ir-sc`	2	1	2	100.00	0	0
Sciadopitys_verticillata	`dr-sc`	1	0	0	98.99	6	3
	`ir-sc`	2	1	2	99.00	1	4
Sciaphila_densiflora	`sc`	1	1	1	100.00	0	0
Selaginella_kraussiana	`dr-sc`	1	1	1	100.00	0	0
Selaginella_vardei	`dr-sc`	1	1	1	100.00	0	0
Taxus_baccata	`sc`	1	1	1	99.77	0	2
Triosteum_pinnatifidum	`ir-dr-sc`	2	0	2	99.28	0	2
Uvaria_macrophylla	`ir-sc`	2	1	2	99.90	0	0
Welwitschia_mirabilis	`ir-sc`	2	1	2	100.00	0	1
Wolffia_australiana	`ir-sc`	4	1	4	99.99	0	0

Begonia_pulchrifolia Contig 4 becomes the starter. So according to the multiplicity computation described in Section 9.2.1, the multiplicities of contigs 1 to 5 become 1, while this one of contig 0 raises to 2.

Carpodetus_serratus Link $(10_r, 11_f)$ is deleted.

Jasminum_tortuosum Link $(6_f, 4_f)$ is added.

Lamprocapnos_spectabilis Contig 8 becomes the starter. So the multiplicities of contigs 2, 4, 6, 10 and 11 increase by one, while the ones of contigs 0, 1 and 3 increase by two.

Lathyrus_pubescens The existence probability of contig 12 raises to 0.01.

Lophocereus_schottii Link $(10_f, 3_f)$ is deleted.

Pelargonium_nanum The multiplicity of contig 2 raises from 3 to 4, without respecting the computation of the multiplicity described in Section 9.2.1.

Podocarpus_totara Link $(6_f, 11_r)$ is deleted.

Triosteum_pinnatifidum The existence probability of contig 7 raises to 0.01.

Table 7 provides all the metrics obtained by running `khloraascaf`. The instances are also solved very quickly (solver times < 3 sec., Table D3). It truthfully finds all the sequences for the modified synthetic data except for Agathis_dammara instance. Although all the repeats (both the direct and the inverted ones) have been retrieved, one of the single-copy region has not been found. It can be explained by extra-links that create alternative paths with the same optimal value for $\mathcal{SCP}$. Note that the oriented region sequence has been still retrieved.

All the (local) misassemblies provided by `Quast` for Carpodetus_serratus, Lathyrus_pubescens,

Pelargonium_nanum and Triosteum_pinnatifidum just concern the nucl. seq. of links that is not a `khloraascaf` issue[9].

10 Conclusion

While the scaffolding problem is traditionally defined with distances data between the reads, in this paper we show it is possible to avoid them in the case of the well-studied circular chloroplast genomes. Based on their specificities, we provide a new scaffolding formulation focused on revealing multimeric forms.

Under the assumption that chloroplast genomes possess few repeats, we formalise their architectures as combinations of direct and inverted repeats, joined by single-copy regions, where the repeats are couples of equal (or reversed) nucleotide sequences. We tackle the chloroplast genome scaffolding as an optimisation problem that yields three suboptimisation ones. We split the inherent multi-objective problem into one optimisation problem per region type. As a consequence, it is necessary to choose the order of subproblem resolutions as a function of the results of previously solved problems. This is what has been addressed through the hierarchical combination strategy. For each subproblem, we give an Integer Linear Programming (ILP) formulation.

As the dedicated chloroplast scaffolding definition is a region-scaffolding-driven, the region graph is a natural data structure to reveal the multimeric genome forms that can exist in a same cell. Indeed, particularly due to an IR flip-flop mechanism, regions between the IRs can be reversed during the genome replication process.

Moreover, we prove the Chloroplast Scaffolding Problem ($\mathcal{CHSP}$) to be $\mathcal{NP}$-Complete in the general case, even though numerical results on perfect artificial data suggest there is a class of $MDCG$ graphs where the problem is in $\mathcal{P}$. Expectingly, noisy artificial data benchmark confirms the theoretical complexity.

We have implemented our approach and the ILP formulations in a `Python` package, `khloraascaf`, that we test on synthetic chloroplast contigs and links data. When the input data permit finding the solution, `khloraascaf` successfully retrieves the genome forms. Even if the problem is $\mathcal{NP}$-Complete, the small size of the input data enables to quickly solve optimally $\mathcal{CHSP}$.

Our results show that the scaffolding-repeat problem formulations $\mathcal{DRP}$ and $\mathcal{IRP}$ seem to be sufficient to scaffold the repeats. This tends to validate our assumptions on the small number of repeats, and especially on the sufficiency of defining the repeats as pairs of equal (reversed) nucleotide sequences.

11 Discussion and perspectives

While our scaffolding problem formulation seems to be sufficient to retrieve the repeats, it is not suitable for single-copy regions. If we applied the maximum-weighted circuit problem to scaffold the single-copy regions after having scaffolded the repeats, that was just to keep things in the context of global optimisation. On the one hand, having weights on links may have been more appropriate than just considering weights on the contigs: in some sense, that is the purpose of distances. On the other hand, `khloraascaf` could initially scaffold the repeats and then propose several solutions to link them, e.g. scored by the existence-probabilities.

Concerning the tests on synthetic data, we should use a more traditional assembly graph input: in fact, as revealed by comparing the `khloraascaf` results with the reference genomes, the used link generation suffers from local, or worse – global, misassemblies. As next step, we plan to inject `khloraascaf` into a state-of-the-art chloroplast genome pipeline, like `GetOrganelle`, and substitute what can be identified as the scaffolding part by our method. Hence, we should be able to relevantly compare `khloraascaf` approach with the state-of-the-art.

`khloraascaf` is sensible to the contig multiplicity computation. For now, a contig multiplicity is obtained by normalising its coverage by this of the starter. A better strategy may be to choose the

[9] For more details on the results for the cleared version of the synthetic data, please refer to https://khloraascaf-results.readthedocs.io/en/latest/benchmark_3/v2_order_analysis/ and https://khloraascaf-results.readthedocs.io/en/latest/benchmark_3/v2_quast_analysis/.

Table 7 Sequence and `Quast` metrics for the modified synthetic data version. The caption is the same as in Table 6.

			Successful		Quast	
Instance	**ILPs**	**Total**	SOC	SOR	**%gnm**	**#mis**
Agathis_dammara	`dr-ir-sc`	10	0	2	99.97	3 2
	`ir-dr-sc`	6	0	0	99.99	2 2
Begonia_pulchrifolia	`ir-sc`	2	1	2	99.99	0 0
Carpodetus_serratus	`ir-sc`	2	1	2	100.00	0 1
Jasminum_tortuosum	`ir-sc`	2	1	2	99.99	0 0
Lamprocapnos_spectabilis	`ir-sc`	2	1	2	99.63	0 3
Lathyrus_pubescens	`dr-sc`	1	0	0	99.01	2 4
	`ir-sc`	2	1	2	99.73	0 1
Lophocereus_schottii	`sc`	1	1	1	100.00	0 0
Pelargonium_nanum	`ir-sc`	2	1	2	97.66	1 2
Podocarpus_totara	`sc`	1	1	1	99.75	0 0
Triosteum_pinnatifidum	`ir-dr-sc`	2	1	2	99.53	0 2

smallest coverage for the normalisation as we expect the multiplicities to be upper-bounds of the contigs use.

Another issue concerns the choice of the starter: while it depends on the result of the mapping of matK gene map on the contigs, $\mathcal{DRP}$, $\mathcal{IRP}$ and $\mathcal{SCP}$ problems may be adapted for a set of candidate for the starter.

To generalise $\mathcal{CHSP}$ on non-equally (reversed) pair of regions for the repeats, two combining ideas are proposed: on the one hand, we can add pairs of contigs to the repeated fragment sets from the user-input. On the other hand, $\mathcal{CHSP}$ should be able to handle the case when a single-copy region is in only one of the regions of a repeat: for now, the contiguity constraint and the objective exclude this case. The contiguity constraint can be adapted to accept only one contiguous region out of the two.

From a user-case perspective, the region graph data structure can be used to determine what genome forms are present in the read set, and in which proportion. Indeed, as the region graph explicitly describes the junctions between the regions, especially between the inverted repeats, one may extract the nucleotide sequences of these junctions to answer the existence of the forms in the read set.

Supplementary information. Supplementary definitions and proofs can be found in Section A and Section B. Metrics for the numerical results can be found in Section C.1, and supplementary results are in Section D.

Availability of data and materials. The `Python3` package `khloraascaf` can be installed with PyPI at https://pypi.org/project/khloraascaf/ and the numerical results can be reproduced thanks to the instructions given in https://khloraascaf-results.readthedocs.io/en/latest/benchmark_3/.

Declarations

Competing interests. The authors declare that they have no competing interest.

Authors' contributions. Both authors conceived the method, designed the ILP models and the algorithms, gave the proof of $\mathcal{NP}$-completeness, wrote the paper. V.E. implemented the `Python3` `khloraascaf` package and tested it on all the data.

Acknowledgments. The authors are grateful to Dominique Lavenier who provided the data for the synthetic data. Many thanks to Jacques Nicolas for his suggestion about formalism aspects. They are thankful to Abdelkader Ainouche for numerous discussions on chloroplast genomics. Sven Schrinner and Gunnar Klau are aknowledged for the valuable discussions about the problem's complexity.

Funding. Not applicable.

Appendix A Repeated fragment set functions

`dirfrag`: $V \to DirF$

$$v \mapsto \begin{cases} (v, v') & \text{if } v_{occ} \mid 2 \\ \text{where } v'_{occ} = v_{occ} + 1 & \\ (v', v) & \text{else } v_{occ} \nmid 2 \\ \text{where } v'_{occ} = v_{occ} - 1 & \end{cases}$$

$\texttt{invfrag}: V \to InvF$

$$v \mapsto \begin{cases} (v, v') & \text{if } v_{or} = 0 \\ \text{where } v'_{or} = 1 \wedge v'_{occ} = v_{occ} + 1 & \\ (v', v) & \text{else } v_{or} = 1 \\ \text{where } v'_{or} = 0 \wedge v'_{occ} = v_{occ} - 1 & \end{cases}$$

$\texttt{diradj}: E \to E$

$$\begin{aligned} (u, v) &\mapsto (u', v') \\ u'_{id} &= u_{id} \\ u'_{or} &= u_{or} \\ u'_{occ} &= \begin{cases} u_{occ} + 1 \text{ if } u_{occ} \mid 2 \\ u_{occ} - 1 \text{ otherwise} \end{cases} \\ v'_{id} &= v_{id} \\ v'_{or} &= v_{or} \\ v'_{occ} &= \begin{cases} v_{occ} + 1 \text{ if } v_{occ} \mid 2 \\ v_{occ} - 1 \text{ otherwise} \end{cases} \end{aligned}$$

$\texttt{invadj}: E \to E$

$$\begin{aligned} (u, v) &\mapsto (v', u') \\ v'_{id} &= v_{id} \\ v'_{or} &= 1 - v_{or} \\ v'_{occ} &= 2\left\lfloor \frac{v_{occ}}{2} \right\rfloor + (1 - v_{or}) \\ u'_{id} &= u_{id} \\ u'_{or} &= 1 - u_{or} \\ u'_{occ} &= 2\left\lfloor \frac{u_{occ}}{2} \right\rfloor + (1 - u_{or}) \end{aligned}$$

Appendix B Reduction of the repeated fragment sets

In this section, we give prove the minimality and the sufficiency of the repeated fragment sets $(RepF, PRepF, ARepF)$ definitions.

B.1 Repeated fragment set reductions

Here we consider the reduction operations for $DirF$ and $InvF$, and conclude on their minimality. Only two vertices $(i, j) \in V^2$ with the same identifier i.e. $i_{id} = j_{id}$ can form a repeated fragment. The relative orientations between i and j is determined according to the type of the repeat the fragment is associated to. The occurrences of the two vertices in a repeated fragment must differ.

Two combinatorial reductions are necessary — *commutative* and *pairing* reductions:

i. commutative reduction means that $\forall (i, j) \in RepF, (j, i) \notin RepF$;
ii. pairing reduction means that it is not necessary to pair all the occurrences cross all the occurrences, and we can group by consecutive occurrences without any intersection.

Reminder of the $DirF$ definition:

$$DirF = \bigcup_{c \in \mathcal{R}} \left\{ \begin{array}{c} (i, j) \in V^2 \text{ s.t.} \\ i_{id} = j_{id} = c_{id} \\ \wedge\, i_{or} = j_{or} \\ \wedge\, i_{occ} = j_{occ} - 1 = 2k \\ 0 \leq k < \left\lfloor \frac{c_{mult}}{2} \right\rfloor \end{array} \right\}$$

Proposition 4 (Commutative reduction for $DirF$)**.**

$$\forall (i, j) \in DirF, (j, i) \notin DirF$$

Proof. Let $(i, j) \in DirF$. $j_{occ} = i_{occ} + 1$ so $(j, i) \notin DirF$ □

Proposition 5 (Pairing reduction for $DirF$)**.** *The direct fragment set contains a sufficient number of direct fragments to form all the solutions.*

Proof. The idea is to find the minimum occurrence distance between two direct fragments such that they can both be chosen to form DRs. Let $(i, j) \in DirF$ be a direct fragment and suppose that it is chosen to participate into a DR. Let $(k, l) \in V^2 \mid k_{id} = l_{id} = i_{id} = j_{id} \wedge k_{or} = l_{or} = i_{or} = j_{or}$ be a candidate for the next direct fragment:

— the occurrences of k and l must differ from the ones of i and j, as a vertex can occupy at most one position, so $j_{occ} < k_{occ}$;
— according to Proposition 4, $k_{occ} < l_{occ}$ and since occurrences are integers, $k_{occ} = l_{occ} - 1$.

So $i_{occ} = j_{occ} - 1 = k_{occ} - 2 = l_{occ} - 3$. □

Proposition 6. *Direct fragment set $DirF$ is a minimal set for $\mathcal{DRP}$.*

Proof. Firstly, we must keep the direct fragment for both the forward ($i_{or} = j_{or} = 0$) and the reverse orientations ($i_{or} = j_{or} = 1$) as the scaffolding problem aims to order and orientate the contigs, so by definition we cannot determine which orientation will participate in the order. Secondly, making step bigger than $2k$ prevents to use all the occurrences of a contig, so it contradicts the contig multiplicity definition. □

Reminder of the $InvF$ definition:

$$InvF = \bigcup_{c \in \mathcal{R}} \left\{ \begin{array}{l} (i,j) \in V^2 \text{ s.t.} \\ \quad i_{id} = j_{id} = c_{id} \\ \quad \wedge\, i_{or} = 1 - j_{or} = 0 \\ \quad \wedge\, i_{occ} = j_{occ} - 1 = 2k \\ \quad 0 \leq k < \left\lfloor \frac{c_{mult}}{2} \right\rfloor \end{array} \right\}$$

Proposition 7 (Commutative reduction for $InvF$)**.**

$$\forall (i,j) \in InvF, (j,i) \notin InvF$$

Proof. Let $(i,j) \in InvF$. $j_{occ} = i_{occ} + 1$ so $(j,i) \notin InvF$ □

Proposition 8 (Pairing reduction for $InvF$)**.** *The inverted fragment set contains a sufficient number of inverted fragments to form all the solutions.*

Proof. The idea is to find the minimum occurrence distance between two inverted fragments such that they can both be chosen to form IRs. Let $(i,j) \in InvF$ be an inverted fragment and suppose that it is chosen to participate into an IR. Let $(k,l) \in V^2 \mid k_{id} = l_{id} = i_{id} = j_{id} \wedge k_{or} \neq l_{or}$ be a candidate for the next direct fragment:

— to respect the property given by Proposition 7, $k_{or} = 1 - l_{or} = 0$

— the occurrences of k and l must differ from the ones of i and j when their orientations are matching, as a vertex can occupy at most one position. Furthermore, for a given occurrence and a given identifier, the two orientations are mutually exclusive, so $j_{occ} < k_{occ}$;

— according to Proposition 7, $k_{occ} < l_{occ}$ and since occurrences are integers, $k_{occ} = l_{occ} - 1$.

So $i_{occ} = j_{occ} - 1 = k_{occ} - 2 = l_{occ} - 3$. □

Proposition 9. *Inverted fragment set $InvF$ is a minimal set for $\mathcal{IRP}$.*

Proof. Step bigger than $2k$ prevents to use all the occurrences of a contig, so it contradicts the contig multiplicity definition. □

B.2 Pairs of repeated fragment set reductions

Here we consider the reduction operations for $PDirF$ and $PInvF$, and conclude on their minimality. Just the *commutative reduction* is necessary.

Reminder of pairs of direct fragments set definition:

$$PDirF = \left\{ \begin{array}{l} \big((i,j),(k,l)\big) \in DirF^2 \text{ s.t.} \\ \quad j_{id} < k_{id} \\ \quad \vee \\ \quad j_{id} = k_{id} \wedge j_{occ} < k_{occ} \end{array} \right\}$$

Proposition 10 (Commutative reduction for $PDirF$)**.**

$$\forall ((i,j),(k,l)) \in PDirF, ((k,l),(i,j)) \notin PDirF$$

Proof. Let $((i,j),(k,l)) \in PDirF$:

— if $j_{id} < k_{id}$ thus $l_{id} > i_{id}$ so $((k,l),(i,j)) \notin PDirF$

— else $j_{id} = k_{id} \wedge j_{occ} < k_{occ}$ thus $l_{occ} > i_{occ}$ so $((k,l),(i,j)) \notin PDirF$

□

Proposition 11. *Pair of direct fragment set $PDirF$ is a minimal set for $\mathcal{DRP}$.*

Proof. By *reductio ad absurdum*: if $PDirF$ can be minimised to a $PDirF'$ set, then there exists a pair of direct fragments $(p,q) \in PDirF \mid (p,q) \notin PDirF'$.

Let $p = (i,j)$ and $q = (k,l)$, and let $q_2 = (m,n) \mid (p,q_2) \in PDirF' \wedge (q,q_2) \in PDirF'$. i,j,k,l,m,n vertices can simultaneously participate in the solution.

Let build a counter-example: suppose that we have the order $i\,k\,l\,j\,m\,n$. This implies the direct fragments p and q fall into a forbidden case for $\mathcal{DRP}$, but this is not detected by Constraints C14 and C15 because $(p,q) \notin PDirF'$. According to Constraint C16, the direct fragments p and q_2 can both match, and direct fragments q and q_2 too. So

$m_p = m_q = m_{q_2} = 1$: absurd because p and q are nested so m_p and m_q should be equal to 0. □

Reminder of pairs of inverted fragments set definition:

$$PInvF = \left\{ \begin{array}{l} \big((i,j),(k,l)\big) \in InvF^2 \text{ s.t.} \\ \quad j_{id} < k_{id} \\ \quad \vee \\ \quad j_{id} = k_{id} \wedge j_{occ} < k_{occ} \end{array} \right\}$$

Proposition 12 (Commutative reduction for $PInvF$)**.**

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

Proof. Let $((i,j),(k,l)) \in PInvF$:
— if $j_{id} < k_{id}$ thus $l_{id} > i_{id}$ so $((k,l),(i,j)) \notin PInvF$
— else $j_{id} = k_{id} \wedge j_{occ} < k_{occ}$ thus $l_{occ} > i_{occ}$ so $((k,l),(i,j)) \notin PInvF$

□

Proposition 13. *Pair of inverted fragment set $PInvF$ is a minimal set for $\mathcal{IRP}$.*

Proof. By *reductio ad absurdum*: if $PInvF$ can be minimised to a $PInvF'$ set, then there exists a pair of inverted fragments $(p,q) \in PInvF \mid (p,q) \notin PInvF'$.

Let be $p = (i,j)$ and $q = (k,l)$, and let $q_2 = (m,n) \mid (p,q_2) \in PInvF' \wedge (q,q_2) \in PInvF'$. i,j,k,l,m,n vertices can simultaneously participate in the solution.

Let build a counter-example: suppose that we have the order $i\,k\,j\,l\,m\,n$. This implies the inverted fragments p and q fall into a forbidden case for $\mathcal{DRP}$, but this is not detected by Constraints C14 and C15 because $(p,q) \notin PInvF'$. According to Constraint C16, the inverted fragments p and q_2 can both match, and inverted fragments q and q_2 too. So $m_p = m_q = m_{q_2} = 1$: absurd because p and q intersect so m_p and m_q should be equal to 0. □

B.3 Adjacent repeated fragment set reductions

Here we consider the reduction operations for $ADirF$ and $AInvF$, and conclude on their minimality.

Reminder of adjacent direct fragments set definition:

$$ADirF = \left\{ \begin{array}{l} (u,v) \in E \text{ s.t.} \\ \quad u_{id} \neq v_{id} \\ \quad \wedge\, u_{occ} = 2k \\ \qquad 0 \leq k < \left\lfloor \frac{u_{mult}}{2} \right\rfloor \\ \quad \wedge\, v_{occ} = 2k' \\ \qquad 0 \leq k' < \left\lfloor \frac{v_{mult}}{2} \right\rfloor \end{array} \right\}$$
$$\bigcup \left\{ \begin{array}{l} (u,v) \in E \text{ s.t.} \\ \quad u_{id} = v_{id} \\ \quad \wedge\, (u_{or} = 0 \vee v_{or} = 0) \\ \quad \wedge\, u_{occ} = 2k \\ \quad \wedge\, v_{occ} = 2k' \\ \qquad 0 \leq k < k' < \left\lfloor \frac{u_{mult}}{2} \right\rfloor \end{array} \right\}$$

This set is the minimum exhaustive set of canonical edges which represent adjacent direct fragments. Two combinatorial reductions are necessary — *canonical reduction for different identifiers* (Propositions 14 and 15) and *same identifiers occurrences* reductions (Propositions 16 and 17).

Proposition 14 (Canonical reductions for $ADirF$ for different identifiers)**.** *$ADirF$ set contains only two of the eight edges that exist between two adjacent direct fragments when the vertex' identifiers are different i.e.*
$\forall(i,j) \in DirF, \forall(k,l) \in DirF \mid i_{id} \neq k_{id}$*:*

$$\begin{array}{l} (i,k) \in E \\ \iff (i,l) \in E \\ \iff (j,k) \in E \\ \iff (j,l) \in E \end{array} \iff \begin{array}{l} (i,k) \in ADirF \\ \wedge(\bar{k},\bar{i}) \in ADirF \\ \wedge(i,l) \notin ADirF \\ \wedge(\bar{l},\bar{i}) \notin ADirF \\ \wedge(j,k) \notin ADirF \\ \wedge(\bar{k},\bar{j}) \notin ADirF \\ \wedge(j,l) \notin ADirF \\ \wedge(\bar{l},\bar{j}) \notin ADirF \end{array}$$

$$\begin{array}{l} (k,i) \in E \\ \iff (k,j) \in E \\ \iff (l,i) \in E \\ \iff (l,j) \in E \end{array} \iff \begin{array}{l} (k,i) \in ADirF \\ \wedge(\bar{i},\bar{k}) \in ADirF \\ \wedge(k,j) \notin ADirF \\ \wedge(\bar{j},\bar{k}) \notin ADirF \\ \wedge(l,i) \notin ADirF \\ \wedge(\bar{i},\bar{l}) \notin ADirF \\ \wedge(l,j) \notin ADirF \\ \wedge(\bar{j},\bar{l}) \notin ADirF \end{array}$$

Proof. The reciprocal are trivial as $ADirF \subset E$.

All the edges using only i, $\overline{i}$, k or $\overline{k}$ belong to $ADirF$ as all the vertex' occurrences are even. While all the edges using j, $\overline{j}$, l or $\overline{l}$ cannot belong to $ADirF$ as one of the vertex' occurrence is odd. □

Proposition 15 (Minimum $ADirF$ set for different identifiers)**.** *When the vertex' identifiers are different, it is not possible to reduce the number of canonical edges based on occurrences comparisons.*

Proof. In order to reduce $ADirF$ when the vertex' identifiers are different, we have to be more restrictive on the occurrences constraints. To reduce the loops on k and k', we have to find an order between the occurrences of u and v.

However, for each order relation between the occurrences ($u_{occ} \leq v_{occ}$ and $u_{occ} \geq v_{occ}$), we can find two associated counter-examples. For the sake of clarity, we will find a counter-example for the case $u_{occ} \leq v_{occ}$. The reasoning is analogous for the second case.

Let a, b and c be three contigs in $\mathcal{C}$ such that all their identifiers are different, $a_{mult} = 2$, $b_{mult} = c_{mult} = 1$, and $(a_f, b_f) \in \mathcal{L}$, $(b_f, a_f) \in \mathcal{L}$ and $(a_f, c_f) \in \mathcal{L}$.

Let $fa_1 = (i, j)$, $fa_2 = (i', j')$ be the two direct fragments associated to a and $fb = (k, l)$, $fc = (m, n)$ the two respectively associated to b and c. The idea behind the proof is to know if the adjacent direct fragments $fa_1\, fb\, fa_2\, fc$ can participate into a solution path $Path$. In such a case, $Path$ is sub-defined by edges $i \to k \to i' \to m$ (and $j \to l \to j' \to n$). Remark that k can permute with l, and m with n.

According to $ADirF$ set definition, $(i, k) \in ADirF$, $(k, i') \in ADirF$ and $(i', m) \in ADirF$. If we constrain the occurrences by the relation $u_{occ} \leq v_{occ}$, it implies that:

$$\begin{aligned} &i_{occ} \leq k_{occ} (= l_{occ} - 1) \\ &l_{occ} - 1 \leq i'_{occ} - 1 \\ &i'_{occ} \leq m_{occ} \end{aligned}$$

As $fa_1 \neq fa_2$, $i_{occ} \neq i'_{occ} - 1$:

if $i_{occ} < i'_{occ} - 1$ then when we focus on the first edges path (in the "best" case n permutes with m) $k_{occ} (= 0) \leq i'_{occ} (= 2) \leq n_{occ} (= 1)$, so it is absurd;

else $i_{occ} > i'_{occ} - 1$ then when we focus on the first edges path, you can permute i' and i and you fall into the same absurd implication as above.

□

Proposition 16 (Occurences reduction for equal identifiers in $ADirF$)**.** *There exists an order between the occurrences of the vertices of an edge between two direct fragments that does not reduce the number of feasible and distinct solutions when the vertex' identifiers are equal.*

Proof. Let $u_1, u_2 \cdots u_n$, $n = 2\left\lfloor \frac{u_{mult}}{2} \right\rfloor$ be same identifiers vertices participating in a solution path for $\mathcal{IRP}$, in this order (i.e. $\forall k \in [\![1; n[\![$, u_k is before u_{k+1} in the solution path).

It can be proven[10] that: regardless the equation set of $k-1$ order relation $R_{k,k+1} \in \{<, >\}$ between two vertices u_{kid} and u_{k+1id} (i.e. $u_{kid} R_{k,k+1} u_{k+1id}, \forall k \in [\![1; n[\![$), there exists a permutation between all distinct occurrences of u such that all the relations $R_{k,k+1}$ are satisfied. Thus, let defined the following equation set (used in $ADirF$):

$$R_{k,k+1} := \begin{cases} < & \text{if } u_{kor} = 0 \vee u_{k+1or} = 0 \\ > & \text{else } u_{kor} = u_{k+1or} = 1 \end{cases}$$

Have we got enough edges to build a path with $\frac{n}{2}$ adjacent direct fragments? On the one hand, the maximum number of edges we would need is: $2\left(\left\lfloor \frac{u_{mult}}{2} \right\rfloor - 1\right)$ (the 2 is caused by direct fragments). On the other hand, $ADirF$ provides:

$$2 \sum_{k'=1}^{\left\lfloor \frac{u_{mult}}{2} \right\rfloor - 1} 1 = 2\left(\left\lfloor \frac{u_{mult}}{2} \right\rfloor - 1\right)$$

edges, (the 2 is caused by `diradj` function). □

Proposition 17 (Canonical reductions for $ADirF$ for equal identifiers)**.** *$ADirF$ set contains only one of the eight edges that exist between two adjacent direct fragments when the vertex' identifiers are equal i.e.*

[10] With Lucas Robidou we are currently writing a paper and an associated algorithm

$\forall (i,j) \in DirF, \forall (k,l) \in DirF \mid i_{id} = k_{id} \wedge i_{occ} < k_{occ}$:

$$\begin{array}{l} (i,k) \in E \\ \iff (i,l) \in E \\ \iff (j,k) \in E \\ \iff (j,l) \in E \end{array} \iff \begin{array}{l} (i,k) \in ADirF \\ \wedge (\bar{k},\bar{i}) \notin ADirF \\ \wedge (i,l) \notin ADirF \\ \wedge (\bar{l},\bar{i}) \notin ADirF \\ \wedge (j,k) \notin ADirF \\ \wedge (\bar{k},\bar{j}) \notin ADirF \\ \wedge (j,l) \notin ADirF \\ \wedge (\bar{l},\bar{j}) \notin ADirF \end{array}$$

$$\begin{array}{l} (k,i) \in E \\ \iff (k,j) \in E \\ \iff (l,i) \in E \\ \iff (l,j) \in E \end{array} \iff \begin{array}{l} (k,i) \notin ADirF \\ \wedge (\bar{i},\bar{k}) \in ADirF \\ \wedge (k,j) \notin ADirF \\ \wedge (\bar{j},\bar{k}) \notin ADirF \\ \wedge (l,i) \notin ADirF \\ \wedge (\bar{i},\bar{l}) \notin ADirF \\ \wedge (l,j) \notin ADirF \\ \wedge (\bar{j},\bar{l}) \notin ADirF \end{array}$$

Proof. The reciprocals are trivial as $ADirF \subset E$.

Only $(i,k) \in ADirF$ and $(\bar{i},\bar{k}) \in ADirF$ as the occurrences of i, $\bar{i}$, k or $\bar{k}$ are even, and $i_{occ} = \bar{i}_{occ} < k_{occ} = \bar{k}_{occ}$.

For the other cases, either they contradict the even-occurrences constraint or the constraint on the occurrences order. □

Reminder of adjacent inverted fragments set definition:

$$AInvF = \left\{ \begin{array}{l} (u,v) \in E \text{ s.t.} \\ \quad u_{id} < v_{id} \\ \quad \wedge\, u_{occ} = 2k + u_{or} \\ \qquad 0 \leq k < \left\lfloor \frac{u_{mult}}{2} \right\rfloor \\ \quad \wedge\, v_{occ} = 2k' + v_{or} \\ \qquad 0 \leq k' < \left\lfloor \frac{v_{mult}}{2} \right\rfloor \end{array} \right\}$$

$$\bigcup \left\{ \begin{array}{l} (u,v) \in E \text{ s.t.} \\ \quad u_{id} = v_{id} \\ \quad \wedge\, (u_{or} = 0 \vee v_{or} = 0) \\ \quad \wedge\, u_{occ} - u_{or} = 2k \\ \quad \wedge\, v_{occ} - v_{or} = 2k' \\ \qquad 0 \leq k < k' < \left\lfloor \frac{u_{mult}}{2} \right\rfloor \end{array} \right\}$$

This set is the minimum exhaustive set of canonical edges which represent adjacent inverted fragments. Two combinatorial reductions are necessary — *canonical reduction for different identifiers* (Propositions 18 and 19) and *same identifiers occurrences* reductions (Propositions 20 and 21).

Proposition 18 (Canonical reductions for $AInvF$ for different identifiers)**.** *$AInvF$ set contains only one of the four edges that exist between two adjacent inverted fragments when the vertex' identifiers are different i.e.*

$\forall (i,j) \in InvF, \forall (k,l) \in InvF \mid i_{id} < k_{id}$:

$$(i,k) \in E \iff \begin{array}{l} (i,k) \in AInvF \\ \wedge (\bar{k},\bar{i}) \notin AInvF \\ \wedge (l,j) \notin AInvF \\ \wedge (\bar{j},\bar{l}) \notin AInvF \end{array}$$

$$(k,i) \in E \iff \begin{array}{l} (k,i) \notin AInvF \\ \wedge (\bar{i},\bar{k}) \notin AInvF \\ \wedge (j,l) \in AInvF \\ \wedge (\bar{l},\bar{j}) \in AInvF \end{array}$$

$$(i,l) \in E \iff \begin{array}{l} (i,l) \in AInvF \\ \wedge (k,j) \notin AInvF \\ \wedge (\bar{l},\bar{i}) \notin AInvF \\ \wedge (\bar{j},\bar{k}) \notin AInvF \end{array}$$

$$(j,k) \in E \iff \begin{array}{l} (j,k) \in AInvF \\ \wedge (\bar{k},\bar{j}) \notin AInvF \\ \wedge (l,i) \notin AInvF \\ \wedge (\bar{i},\bar{l}) \notin AInvF \end{array}$$

Proof. The reciprocals are trivial as $AInvF \subset E$.

Let $(i,j) \in InvF$, $(k,l) \in InvF$ be two adjacent inverted fragments such that $i_{id} < k_{id}$. We will focus on demonstrating the case $(i,k) \in E$. The others follow the same logics.

- $(i,k) \in AInvF$ as $\exists n \in [\![0; \lfloor \frac{i_{mult}}{2} \rfloor [\![$ such that $i_{occ} = 2n + i_{or} = 2n$, and $\exists n' \in [\![0; \lfloor \frac{k_{mult}}{2} \rfloor [\![$ such that $k_{occ} = 2n' + k_{or} = 2n' + 1$;
- $(\bar{k},\bar{i}) \notin AInvF$ as $\bar{k}_{id} > \bar{i}_{id}$;
- $(l,j) \notin AInvF$ as $l_{id} > j_{id}$;
- $(\bar{j},\bar{l}) \notin AInvF$ as $\bar{j}_{or} = 0$ and $\bar{j}_{occ} \nmid 2$ so $\nexists n \in [\![0; \lfloor \frac{j_{mult}}{2} \rfloor [\![$ such that $\bar{j}_{occ} = 2n$.

□

Proposition 19 (Minimum $AInvF$ set for different identifiers)**.** *When the vertex' identifiers are different, it is not possible to reduce the number of canonical edges based on occurrences comparisons.*

Proof. In order to reduce $AInvF$ when the vertex' identifiers are different, we have to be more

restrictive on the occurrences constraints. To reduce the loops on k and k', we have to find an order between the occurrences of u and v.

However, for each order relation between the occurrences ($u_{occ} \leq v_{occ}$ and $u_{occ} \geq v_{occ}$), we can find two associated counter-examples. For the sake of clarity, we will find a counter-example for the case $u_{occ} \leq v_{occ}$. The reasoning is analogous for the second case.

Let c and d be two contigs in $\mathcal{C}$ such that $c_{id} < d_{id}$, $c_{mult} = 4$ and $d_{mult} = 2$, and $(c_f, d_f) \in \mathcal{L}$ and $(d_f, c_f) \in \mathcal{L}$. Let $p_1 = (i, j)$, $p_2 = (i', j')$ be the two inverted fragments associated to c and $q = (k, l)$ the one associated to d. The idea behind the proof is to know if the adjacent inverted fragments $p_1\, q\, p_2$ can participate into a solution path $Path$. In such a case, $Path$ is sub-defined by edges $i \to k \to i'$ and $j' \to l \to j$.

According to $AInvF$ set definition, $(i, k) \in AInvF$ and $(j', l) \in AInvF$. If we constrain the occurrences by the relation $u_{occ} \leq v_{occ}$, thus on the one hand, $i_{occ} \leq k_{occ} (= l_{occ} - 1)$, and on the other hand $l_{occ} - 1 \geq j'_{occ} - 1$. As $p_1 \neq p_2$, $i_{occ} \neq j'_{occ} - 1$:

if $i_{occ} < j'_{occ} - 1$ then $k_{occ}(= 0) > i_{occ}(= 0)$ so it is absurd;
else $i_{occ} > j'_{occ} - 1$ then $i_{occ}(= 2) \leq k_{occ}(= 0)$ so it is absurd.

□

Proposition 20 (Occurences reduction for equal identifiers in $AInvF$)**.** *There exists an order between the occurrences of the vertices of an edge between two inverted fragments that does not reduce the number of feasible and distinct solutions when the vertex' identifiers are equal.*

Proof. Let $u_1, u_2 \cdots u_n$, $n = 2\lfloor \frac{u_{mult}}{2} \rfloor$ be same identifiers vertices participating in a solution path for $\mathcal{IRP}$, in this order (i.e. $\forall k \in [\![1; n[\![$, u_k is before u_{k+1} in the solution path).

It can be proven[11] that: regardless the equation set of $k-1$ order relation $R_{k,k+1} \in \{<, >\}$ between two vertices u_{kid} and u_{k+1id} (i.e. $u_{kid} R_{k,k+1} u_{k+1id}, \forall k \in [\![1; n[\![$), there exists a permutation between all distinct occurrences of u such that all the relations $R_{k,k+1}$ are satisfied.

[11] With Lucas Robidou we are currently writing a paper and an associated algorithm

Thus, let define the following equation set (used in $AInvF$):

$$R_{k,k+1} := \begin{cases} < & \text{if } u_{kor} = 0 \vee u_{k+1or} = 0 \\ > & \text{else } u_{kor} = u_{k+1or} = 1 \end{cases}$$

Have we got enough edges to build a path with $\frac{n}{2}$ adjacent inverted fragments? On the one hand, the maximum number of edges we would need is: $2\big(\lfloor \frac{u_{mult}}{2} \rfloor - 1\big)$ (the 2 is caused by inverted fragments). On the other hand, $AInvF$ provides:

$$2 \sum_{k'=1}^{\lfloor \frac{u_{mult}}{2} \rfloor - 1} 1 = 2\Big(\lfloor \frac{u_{mult}}{2} \rfloor - 1\Big)$$

edges, (the 2 is caused by `invadj` function). □

Proposition 21 (Canonical reductions for $AInvF$ for equal identifiers)**.** *$ADirF$ set contains only one of the four edges that exist between two adjacent inverted fragments when the vertex' identifiers are different i.e.*

$\forall (i, j) \in InvF, \forall (k, l) \in InvF \mid i_{id} = k_{id} \wedge i_{occ} < k_{occ}$*:*

$$(i,k) \in E \iff \begin{array}{l} (i,k) \in AInvF \\ \wedge (\bar{k}, \bar{i}) \notin AInvF \\ \wedge (l, j) \notin AInvF \\ \wedge (\bar{j}, \bar{l}) \notin AInvF \end{array}$$

$$(i,l) \in E \iff \begin{array}{l} (i,l) \in AInvF \\ \wedge (k, j) \notin AInvF \\ \wedge (\bar{l}, \bar{i}) \notin AInvF \\ \wedge (\bar{j}, \bar{k}) \notin AInvF \end{array}$$

$$(j,k) \in E \iff \begin{array}{l} (j,k) \in AInvF \\ \wedge (\bar{k}, \bar{j}) \notin AInvF \\ \wedge (l, i) \notin AInvF \\ \wedge (\bar{i}, \bar{l}) \notin AInvF \end{array}$$

Nota bene *$(k, i) \in E$ case is not here because (i, k) does not change the solution as their sequences are equal. This non-redundancy advantage is allowed by occurrence reduction in Proposition 20.*

Proof. The reciprocals are trivial as $AInvF \subset E$.

Let $(i, j) \in InvF$, $(k, l) \in InvF$ be two adjacent inverted fragments such that $i_{id} = k_{id}$ and $i_{occ} < k_{occ}$. We will focus on demonstrating the case $(i, k) \in E$. The others follow the same logics.

— $(i,k) \in AInvF$ as $\exists n \in [\![0; \lfloor \frac{i_{mult}}{2} \rfloor [\![$ such that $i_{occ} = 2n + i_{or} = 2n$, and $\exists n' \in [\![0; \lfloor \frac{k_{mult}}{2} \rfloor [\![$ such that $k_{occ} = 2n' + k_{or} = 2n' + 1$;
— $(\overline{k}, \overline{i}) \notin AInvF$ as $\overline{k}_{occ} > \overline{i}_{occ}$;
— $(l,j) \notin AInvF$ as $l_{occ} > j_{occ}$;
— $(\overline{j}, \overline{l}) \notin AInvF$ as $\overline{j}_{or} = 0$ and $\overline{j}_{occ} \nmid 2$ so $\nexists n \in [\![0; \lfloor \frac{j_{mult}}{2} \rfloor [\![$ such that $\overline{j}_{occ} = 2n$.

□

Appendix C Metrics

C.1 `Quast` metrics

`Quast` metric descriptions can be found at https://quast.sourceforge.net/docs/manual.html#sec3.1. Here we adapt the description of the ones we use in this paper.

misassemblies is the number of positions in the contigs (breakpoints) that satisfy one of the following criteria:
— the left flanking sequence aligns over 1 kbp away from the right flanking sequence on the reference;
— flanking sequences overlap on more than 1 kbp;
— flanking sequences align to different strands.

local misassemblies is the number of positions in the contigs (breakpoints) that satisfy the following conditions:
— the gap or overlap between left and right flanking sequences is less than 1 kbp, and larger than 200 bp (the maximum indel length);
— the left and right flanking sequences both are on the same strand.

Genome fraction (%) is the percentage of aligned bases in the reference genome. A base in the reference genome is aligned if there is at least one contig with at least one alignment to this base. Contigs from repetitive regions may map to multiple places, and thus may be counted multiple times

Table D1 Benchmark 3 v1 contig quast

Instance	ILPs	%gnm	#mis	
Abies_alba	dr-sc	99.32	0	0
	ir-sc	99.32	0	0
Acorus_americanus	ir-sc	99.52	0	0
Agathis_dammara	dr-ir-sc	80.19	0	0
	ir-dr-sc	80.19	0	0
Azima_tetracantha	ir-sc	99.93	0	0
Begonia_pulchrifolia	—	—	—	—
Carpodetus_serratus	ir-sc	98.61	0	0
Circaeaster_agrestis	ir-sc	99.56	0	0
Clematis_repens	ir-sc	99.93	0	0
Commiphora_foliacea	ir-sc	98.34	0	0
Cucumis_hystrix	ir-sc	99.49	0	0
Eucommia_ulmoides	ir-sc	99.01	0	0
Jasminum_tortuosum	ir-sc	99.77	0	0
Juniperus_scopulorum	sc	98.82	0	0
Lamprocapnos_spectabilis	dr-sc	35.11	0	0
Lathyrus_pubescens	dr-sc	98.20	0	0
	ir-sc	98.20	0	0
Lophocereus_schottii	sc	98.63	0	0
Musa_ornata	ir-sc	99.08	0	0
Oenothera_glazioviana	ir-sc	98.50	0	0
Pelargonium_nanum	ir-sc	96.04	0	0
Podocarpus_totara	sc	98.76	0	0
Porphyra_purpura	dr-sc	99.95	0	0
Sagittaria_trifolia	ir-sc	98.55	0	0
Sciadopitys_verticillata	dr-sc	96.24	0	0
	ir-sc	96.24	0	0
Sciaphila_densiflora	sc	99.87	0	0
Selaginella_kraussiana	dr-sc	99.84	0	0
Selaginella_vardei	dr-sc	99.98	0	0
Taxus_baccata	sc	98.07	0	0
Triosteum_pinnatifidum	ir-dr-sc	98.30	0	0
Uvaria_macrophylla	ir-sc	98.35	0	0
Welwitschia_mirabilis	ir-sc	99.17	0	0
Wolffia_australiana	ir-sc	99.83	0	0

Appendix D Supplementary results

References

Andonov R, Djidjev H, François S, et al (2019) Complete assembly of circular and chloroplast genomes based on global optimization. Journal of Bioinformatics and Computational Biology 17(3):1950014. https://doi.org/10.1142/S0219720019500148

Ankenbrand MJ, Pfaff S, Terhoeven N, et al (2018) chloroExtractor: Extraction and assembly of the chloroplast genome from whole genome shotgun data. Journal of Open Source Software 3(21):464. https://doi.org/10.21105/joss.00464

Bendich AJ (2004) Circular chloroplast chromosomes: The grand illusion. The Plant Cell 16(7):1661–1666. https://doi.org/10.1105/tpc.160771

Table D2 benchmark 3 v1 ilp stats

Instance	$\|\mathcal{C}\|$	$\|\mathcal{L}\|$	$\|V\|$	$\|E\|$	Time
Abies_alba	9	28	20	36	0.02
					0.02
Acorus_americanus	5	16	16	40	0.04
					0.05
Agathis_dammara	20	54	50	96	0.20
					0.64
Azima_tetracantha	7	20	16	28	0.00
					0.00
Begonia_pulchrifolia	6	14	12	14	0.00
					0.00
Carpodetus_serratus	13	44	36	76	0.14
					0.21
Circaeaster_agrestis	13	36	44	112	0.37
					4.06
Clematis_repens	6	16	18	38	0.04
					0.04
Commiphora_foliacea	16	38	38	58	0.04
					0.04
Cucumis_hystrix	9	22	22	40	0.02
					0.03
Eucommia_ulmoides	13	32	32	52	0.03
					0.05
Jasminum_tortuosum	10	26	30	68	0.07
					0.13
Juniperus_scopulorum	10	28	20	28	0.00
					0.00
Lamprocapnos_spectabilis	12	36	26	46	0.00
					0.00
Lathyrus_pubescens	17	44	36	52	0.02
					0.02
Lophocereus_schottii	12	36	24	36	0.00
					0.00
Musa_ornata	15	32	46	82	0.24
					0.87
Oenothera_glazioviana	11	30	30	56	0.07
					0.08
Pelargonium_nanum	11	32	34	108	0.17
					0.43
Podocarpus_totara	13	42	26	42	0.00
					0.03
Porphyra_purpura	3	8	8	16	0.00
					0.00
Sagittaria_trifolia	15	32	34	46	0.02
					0.02
Sciadopitys_verticillata	25	70	52	78	0.02
					0.04
Sciaphila_densiflora	4	8	8	8	0.00
					0.00
Selaginella_kraussiana	5	12	14	26	0.01
					0.01
Selaginella_vardei	3	8	8	16	0.00
					0.00
Taxus_baccata	17	42	34	42	0.00
					0.00
Triosteum_pinnatifidum	13	40	32	66	0.07
					0.15
Uvaria_macrophylla	13	30	42	86	0.24
					0.84
Welwitschia_mirabilis	13	34	30	48	0.02
					0.03
Wolffia_australiana	10	28	24	44	0.02
					0.02

Table D3 benchmark 3 v2 ilp stats

Instance	$\|\mathcal{C}\|$	$\|\mathcal{L}\|$	$\|V\|$	$\|E\|$	Time
Agathis_dammara	20	54	52	108	0.22
					0.83
Begonia_pulchrifolia	6	14	14	22	0.00
					0.00
Carpodetus_serratus	13	42	36	74	0.13
					0.20
Jasminum_tortuosum	10	28	30	70	0.08
					0.13
Lamprocapnos_spectabilis	12	36	48	180	0.68
					2.67
Lathyrus_pubescens	17	44	36	52	0.02
					0.02
Lophocereus_schottii	12	34	24	34	0.00
					0.00
Pelargonium_nanum	11	32	36	128	0.33
					0.84
Podocarpus_totara	13	40	26	40	0.00
					0.00
Triosteum_pinnatifidum	13	40	32	66	0.07
					0.18

Bock R, Knoop V (eds) (2012) Genomics of Chloroplasts and Mitochondria, Advances in Photosynthesis and Respiration, vol 35. Springer Netherlands, Dordrecht, https://doi.org/10.1007/978-94-007-2920-9

Chateau A, Giroudeau R (2015) A complexity and approximation framework for the maximization scaffolding problem. Theoretical Computer Science 595:92–106. https://doi.org/10.1016/j.tcs.2015.06.023

Chikhi R, Rizk G (2012) Space-Efficient and Exact de Bruijn Graph Representation Based on a Bloom Filter. In: Raphael B, Tang J (eds) Algorithms in Bioinformatics. Springer, Berlin, Heidelberg, Lecture Notes in Computer Science, pp 236–248, https://doi.org/10.1007/978-3-642-33122-0_19

Coissac E, Hollingsworth PM, Lavergne S, et al (2016) From barcodes to genomes: Extending the concept of DNA barcoding. Molecular Ecology 25(7):1423–1428. https://doi.org/10.1111/mec.13549

Deng XW, Wing RA, Gruissem W (1989) The chloroplast genome exists in multimeric forms. Proceedings of the National Academy of Sciences 86(11):4156–4160. https://doi.org/10.1073/pnas.86.11.4156

Dierckxsens N, Mardulyn P, Smits G (2017) NOVOPlasty: De novo assembly of organelle genomes from whole genome data. Nucleic Acids Research 45(4):e18. https://doi.org/10.1093/nar/gkw955

François S, Andonov R, Lavenier D, et al (2018) Global Optimization for Scaffolding and Completing Genome Assemblies. Electronic Notes in Discrete Mathematics 64:185–194. https://doi.org/10.1016/j.endm.2018.01.020

Gurevich A, Saveliev V, Vyahhi N, et al (2013) QUAST: Quality assessment tool for genome assemblies. Bioinformatics 29(8):1072–1075. https://doi.org/10.1093/bioinformatics/btt086

Gusfield D (2019) The RNA-Folding Problem. In: Gusfield D (ed) Integer Linear Programming in Computational and Systems Biology: An Entry-Level Text and Course. Cambridge University Press, Cambridge, p 105–121, https://doi.org/10.1017/9781108377737.008

Huson DH, Reinert K, Myers EW (2002) The greedy path-merging algorithm for contig scaffolding. Journal of the ACM 49(5):603–615. https://doi.org/10.1145/585265.585267

Jin JJ, Yu WB, Yang JB, et al (2020) GetOrganelle: A fast and versatile toolkit for accurate de novo assembly of organelle genomes. Genome Biology 21(1):241. https://doi.org/10.1186/s13059-020-02154-5

Mandric I, Zelikovsky A (2015) ScaffMatch: Scaffolding algorithm based on maximum weight matching. Bioinformatics 31(16):2632–2638. https://doi.org/10.1093/bioinformatics/btv211

Miller CE, Tucker AW, Zemlin RA (1960) Integer Programming Formulation of Traveling Salesman Problems. Journal of the ACM 7(4):326–329. https://doi.org/10.1145/321043.321046

Palmer JD (1983) Chloroplast DNA exists in two orientations. Nature 301(5895):92–93. https://doi.org/10.1038/301092a0

Palmer JD (1985) Comparative Organization of Chloroplast Genomes. Annual Review of Genetics 19(1):325–354. https://doi.org/10.1146/annurev.ge.19.120185.001545

Salmela L, Mäkinen V, Välimäki N, et al (2011) Fast scaffolding with small independent mixed integer programs. Bioinformatics 27(23):3259–3265. https://doi.org/10.1093/bioinformatics/btr562

Schrijver A (2003) Combinatorial Optimization: Polyhedra and Efficiency. Springer Science & Business Media

Wang J, Wong GKS, Ni P, et al (2002) RePS: A Sequence Assembler That Masks Exact Repeats Identified from the Shotgun Data. Genome Research 12(5):824–831. https://doi.org/10.1101/gr.165102

Wang W, Lanfear R (2019) Long-Reads Reveal That the Chloroplast Genome Exists in Two Distinct Versions in Most Plants. Genome Biology and Evolution 11(12):3372–3381. https://doi.org/10.1093/gbe/evz256

Xiao-Ming Z, Junrui W, Li F, et al (2017) Inferring the evolutionary mechanism of the chloroplast genome size by comparing whole-chloroplast genome sequences in seed plants. Scientific Reports 7(1):1555. https://doi.org/10.1038/s41598-017-01518-5