Integer Programming Approach for Nested Pairs Genome Scaffolding

Victor Epain, Rumen Andonov

Integer Programming Approach for Nested Pairs Genome Scaffolding

Victor Epain and Rumen Andonov

Univ. Rennes, Inria, IRISA, F-35000 Rennes, France
{victor.epain, rumen.andonov}@irisa.fr

Abstract. Scaffolding step in the genome assembly aims to determine the order and the orientation of a huge number of previously assembled genomic fractions (*contigs/scaffolds*). Here we introduce a particular case of this problem and denote it by *Nested Pairs Scaffolding.* We formulate it as an optimisation problem and propose an integer programming formulation for its resolution. The performed computational results on real and synthetic data show an excellent behaviour of our formulation.

Keywords: *De novo* genome assembly · Contig · Scaffolding · Elementary longest path problem · Integer programming · NP-Hard problem

1 Introduction

Genome assembly aims to build the genome from a large amount of small DNA sequences named reads. This process can be roughly separated into two main steps. First, reads are merged into longer sequences named *contigs.* Each contig has potentially two *orientations* (arbitrarily defined as *forward* and *reverse*). However, a contig contributes to the genome build with only one (but unknown which one) of its two orientations. In the second stage, called *scaffolding* — the main focus of this study, the contigs are put in the correct order and orientation towards the completion of the final assembly. Usually this step uses additional data *e.g.* given distances between contigs [1], or homology references from near-species [7]. In contrast, we show here that chloroplasts' genomes can be assembled without any raw additional data, but only using the knowledge that they possess highly conserved circular and quadripartite structures, with a pair of dispersed inverted repeat regions [2] (*c.f.* for illustration Subfigure 1a). Inverted repeats correspond to occurrences of contigs paired with other occurrences of them but in reverse orientation. Therefore paired contigs' positions on the assembled sequence must satisfy the nested pairs pattern.

We formulate the above constraints in terms of integer linear program where the objective is to maximise the nested pairs number. This results in finding a couple of longest contiguous inverted repeats and we find the genome scaffolding as a solution of an elementary path problem with additional constraints for nested pairs. To the best of our knowledge, such an approach has not been previously applied for genome assembly. A similar approach has been carried out for RNA folding problem that aims to predict the secondary structure of an RNA molecule, given only its nucleotide sequence [5]. However, in contrast to the latter where the sequence elements correspond to nucleotide bases with known indices and the goal is to find nested pairs alignment, in our case the positions of the contigs are variables and correspond to their locations in an unknown elementary path with additional constraints for nested pairs.

2 Problem Description

Biological data provide DNA fragments from both of the DNA strands molecule's clones. These fragments are pre-assembled in longer fragments called *contigs*, so that redundant data from clones are compressed in one. We aim here to finish the assembly (*scaffolding* step) using the available chloroplast genome's structure information.

Finishing the assembly is possible thanks to the given succession relations (*links*) between contigs with orientation attribute. Because of repeats in the genome, some contigs can be in several copies in the solution, both orientations combined. An upper bound of the number of copies for each contig, called *multiplicity*, is provided (*c.f.* Table 1 for an instance of input data: set of contigs and set of links). Furthermore, we are sure there is one contig which has only one copy (we call it *seed contig*).

Chloroplasts' genome has well-described structure, especially concerning repeated regions [2]. One particular case of such a structure is given in Subfigure 1a. Our finishing assembly approach is based on reconstructing inverted repeats, by considering them as nested couples of contiguous inverted regions. Thus, we model the chloroplast scaffolding step as building a sequence of oriented contigs, starting from and ending with the seed contig, which maximises the number of nested and contiguous inverted fragments.

2.1 Definitions and Notations

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

Contigs and Links Input data are described by a contigs set $\mathcal{C}$ and contigs successions set $\mathcal{L}$ (links). Table 1 shows an example of input data.

Any contig $c \in \mathcal{C}$ is characterized by its identifier $c_{id} \in \mathbb{N}$, its length $c_{len} \in \mathbb{N}^*$ (*i.e.* the number of characters in c), its multiplicity $c_{mult} \in \mathbb{N}^*$ (an upper bound of the number of occurrences of c).

Links are succession relations between a couple of oriented contigs, thus $\mathcal{L} \subset (\mathcal{C} \times \mathbb{B})^2$. Each contig c is provided here with its orientation $c_{or} \in \mathbb{B}$, where the binary value 0 stands for *forward* (f), *i.e.* the original contig's sequence, while the binary value 1 stands for *reverse* (r), *i.e.* the reverse complement of the contig's sequence. By convention, $(orc, ord) \in \mathcal{L} \iff (\overline{ord}, \overline{orc}) \in \mathcal{L}$, where $\overline{orc}$ denotes the reverse complement of orc.

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

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

c_{id}	c_{len}	c_{mult}
2	18914	2
3	19212	1
4	88398	1
5	7596	2

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

c_{id}	c_{or}	d_{id}	d_{or}
2	1	3	1
2	1	3	0
2	0	5	1
4	0	5	0
4	1	5	0

Multiplied Doubled Contigs Graph (MDCG) Based on this input and following the below rules we construct a directed graph $G = (V, E)$ called a *multiplied doubled contigs graph (MDCG)*[1] where:

i. Each contig $c \in \mathcal{C}$ is doubled according forward and reverse orientation. So $\forall v \in V$ such as $v_{id} = c_{id}$, v has an orientation attribute v_{or} equals to 0 or 1 respectively for forward or reverse orientation. We denote by $\overline{v}$ the reverse orientation of v (so that $v_{or} = 1 - \overline{v}_{or}$, and the other attributes remain the same)
ii. Moreover, each doubled contig is multiplied by its multiplicity. Thus, if contig c is multiplied k times, it generates a set of $2k$ vertices $v_0, \dots, v_{k-1}$, $\overline{v}_0, \dots, \overline{v}_{k-1}$ where all these vertices have the same identifier c_{id}. Let $0 \leq v_{occ} < c_{mult}$ be an occurrence of a multiplied contig c.

Denote $N = \sum\limits_{c \in \mathcal{C}} c_{mult}$, hence $|V| = 2N$. Furthermore, set $M = \sum\limits_{v \in V} v_{len}$.

The set E of edges is constructed based on the links data $\mathcal{L}$ and applying the same multiplication rules as for the contigs. Furthermore, for any edge in E, its reverse is also added in E in order to validate the feature $(u, v) \in E \iff \overline{(u, v)} = (\overline{v}, \overline{u}) \in E$.

2.2 Modelling the Scaffolding Problem as a Path Search in a Graph: A Mixed Integer Linear Programming Formulation

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

In order to define a path in a graph, usually one needs to designate its start and its terminal vertex. We use here for this purpose the given seed contig. We denote its corresponding vertex by sc and we suppose that it is in its forward

[1] The graph modeling descibed here has been yet applied in our previous scaffolding papers [1,3]. However, the notation (MDCG) is used here firstly.

orientation. We also know that it is present in the graph with only one occurrence. Furthermore, we replace sc by two new vertices s and t, both in forward orientation and with only one occurrence, where s gets all outgoing sc edges, while t gets all incoming sc edges. Note that we add neither $\overline{s}$ nor $\overline{t}$ and that $N_s^- = N_t^+ = \emptyset$, where $\emptyset$ denotes the empty set.

Path Definition Vertices s and t will be used as the source (start) and the target (terminal), respectively, of the path we are looking for. We associate a variable $i_v \in [0, 1]$ with any vertex $v \in V \setminus \{s, t\}$ to encode whether v is, or isn't in the solution path. Moreover, for each vertex v, only one of its orientations participates in the path, which we formulate as:

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

Equations (2)-(7) below follow the single-flow formulation of the $s - t$ path that is used in the field [8] and that we have adapted to the scaffolding problem specificities in our previous papers [1,3].

We introduce binary variables for the edges:

$$\forall e \in E: \quad x_e \in \mathbb{B} \tag{2}$$

If a vertex $v \notin \{s, t\}$ is an intermediate vertex in the path, then v possesses exactly one incoming/outgoing edge. If v is not in the path, then no edge inputs/exits v, *i.e.*

$$\sum_{v \in N_s^+} x_{sv} = 1, \text{ and } \sum_{u \in N_t^-} x_{ut} = 1 \tag{3}$$

$$\forall v \in V \setminus \{s,t\}: \quad i_v = \sum_{u \in N_v^-} x_{uv} = \sum_{w \in N_v^+} x_{vw} \tag{4}$$

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

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

Then,

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

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

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

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

Constraints (1) – (7) define an elementary path from s to t. In the sequel, we add to this model non-intersecting inverted fragments constraints.

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

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

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

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

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

The condition on the identifier and the occurrences attributes assures that for each pair $(p,q) \in PInvF$, its permutation pair $(q,p) \notin PInvF$, and generates the minimum number of items for building all distinct solutions. The set $PInvF$ contains unordered pairs of inverted fragments that must be checked to be admissible solutions or not in respect to the below definition.

Let $\big((i,j),(k,l)\big) \in PInvF$ and consider the positions of these inverted fragments in the genome. In order to illustrate the options for their relative locations, and without loss of generality, we assume that $pos_i < pos_j, pos_k < pos_l$ and that the interval $[\![pos_i, pos_j]\!]$ is smaller or equal to the interval $[\![pos_k, pos_l]\!]$. Only one of the following three cases holds:

- $[\![pos_i, pos_j]\!] \cap [\![pos_k, pos_l]\!] = \emptyset$
- $[\![pos_i, pos_j]\!] \subseteq [\![pos_k, pos_l]\!]$
- $[\![pos_i, pos_j]\!] \cap [\![pos_k, pos_l]\!] \neq \emptyset$ and $[\![pos_i, pos_j]\!] \subsetneq [\![pos_k, pos_l]\!]$

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

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

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

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

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

$$\alpha_u^v \leq pos_u + pos_v \qquad (10)$$

where $M = \sum_{v \in V} v_{len}$. Note that $\alpha_u^v = 1 - \alpha_v^u$, hence only one of the variables α_u^v and α_v^u needs to be maintained.

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

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

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

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

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

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

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

$$\forall (u, v) \in InvF : \quad 2 \times m(u, v) \leq i_u + i_v \tag{13}$$

Inverted Repeats Contiguity Inverted fragments represent subsequences in the inverted repeats regions. In this section we study how to assemble them in a contiguous way, and how to make these assembled regions as long as possible. We achieve this goal by assembling *adjacent inverted fragments*. Those are the extremities of two edges such that only one of them (considered as *canonical edge*) belongs to a set called $AInvF$. Thus, since $AInvF$ contains only canonical edges, it is necessary to retrieve:

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

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

- the canonical edge (u, v) and its adjacency reverse `adjrev(u,v)` participate in the solution, *i.e.*

$$\forall (u, v) \in AInvF : \quad 2 \times adjif(u, v) \leq x_{uv} + x_{\texttt{adjrev}(u,v)} \tag{14}$$

- the associated inverted fragments do not intersect, *i.e.*

$$\forall (u, v) \in AInvF : 2 \times adjif(u, v) \leq m(\texttt{invfrag}(u)) + m(\texttt{invfrag}(v)) \tag{15}$$

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

$$\max \quad \sum_{p \in InvF} m(p) + \sum_{(u,v) \in AInvF} adjif(u, v)$$
$$\text{subject to constraints (1) to (15)}$$

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

3 Complexity Analysis

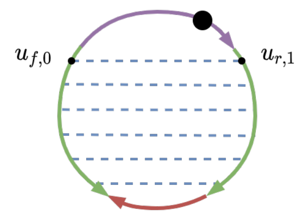
Proposition 1. *Nested Pairs Scaffolding Problem* $(NPSP)$ *is* NP-Hard.

Proof. By reduction from the longest path problem from vertex s to vertex t $(LPSTP)$, known to be NP-Hard [6].

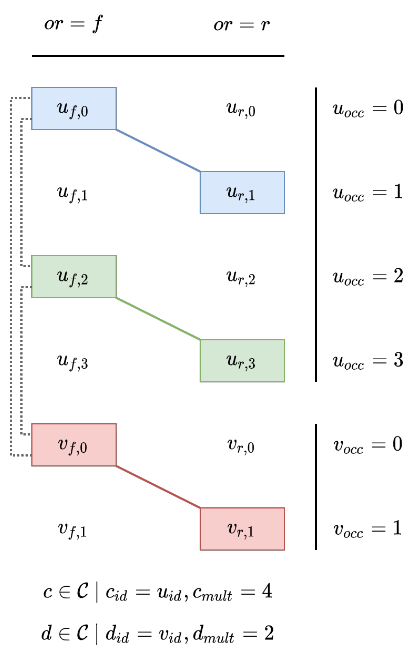
Consider an instance $\mathbb{I} \in LPSTP$ that is composed of a directed graph $G = (V, E)$ and two vertices $s \in V$ and $t \in V$. We shall build an instance transform function $mdgf$ such that $\mathbb{I} \in LPSTP \iff mdgf(\mathbb{I}) \in NPSP$. As illustrated in Figure 2, function $mdgf$ transforms 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).

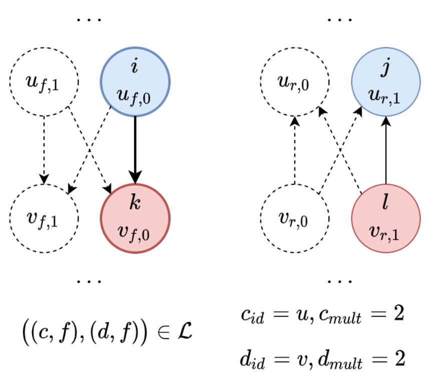
All subgraphs G'_* in (b) are quasi-clones of $G_{V\setminus\{s,t\}}$ in (a). In the graphs $G'_{f0} = (V'_{f0}, E'_{f0})$ and $G'_{f1} = (V'_{f1}, E'_{f1})$ edges are oriented in the same direction as in $G_{V\setminus\{s,t\}}$. Vertices in V'_{f0}, V'_{f1} have forward orientation attribute, and respectively 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 constructed based on G'_* graphs' data. We assume that inverted fragments are composed of vertices from V_{f0} and V_{r1} uniquely, (*i.e.* $\forall (u, v) \in InvF, u \in V_{f0}, v \in V_{r1}$). This is visualised by blue dashed vertical lines from G'_{f0} to G'_{r1}. Vertices i_f and i_r allow the transition from G'_{f0} and G'_{f1} to G'_{r0} and G'_{r1}.

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



(a) Chloroplast genome's structure





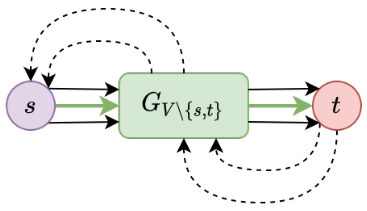
(c) Adjacent inverted fragments

(b) Inverted fragments and pairs

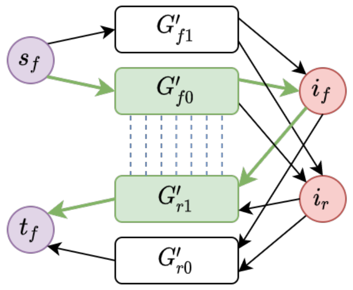
Fig. 1. Inverted fragments sets illustrations. **(a)** Illustration of one well known chloroplasts' circular genome structure: purple and red regions are unique, while green ones are *inverted repeats* (one is obtained by reversing and complementing the sequence of the other). Black dots are contigs, and the biggest one represents the starter. For a given contig $u_{f,0}$ on the left-side green region, the contig in front of it (on the right-side green region) has the reverse orientation (so r). Because orientations are mutually exclusive, it is necessary to chose an occurrence that differs 0 (so 1). The couple $u_{f,0}, u_{r,1}$ is defined as inverted fragment. Thus, inverted repeats can be modelled as a sequence of nested inverted fragments (as illustrated with blue dashed lines). **(b)** Two vertices $i, j \in V^2 \mid i_{id} = j_{id}$ are inverted fragments if they are coloured identically. They belong to the $InvF$ set. As an illustration of `invfrag` function, note that `invfrag`$(u_{f,0}) = u_{r,1}$, and `invfrag`$(u_{r,3}) = u_{f,2}$. Inverted fragments linked by a dashed line represent pairs that belong to $PInvF$ set (here three). Two inverted fragments $(p, q) \in PInvF$ can be chosen in the solution only if they do not intersect. **(c)** $(i, j) \in InvF$ and $(k, l) \in InvF$ are two adjacent inverted fragments in MDCG. Remind that $k \in N_i^+ \iff j \in N_l^+$. We aim to maximise the number of such adjacent inverted fragments in the solution. We do not consider dashed edges in $AInvF$ because their extremities participate in no one of the $InvF$'s item. Note that edge $(i, k) \in E$ (in bold) is a canonical — edge $(l, j) \in E$ can be retrieved by `adjrev`$((i, k)) = (l, j)$.

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 finding the longest path $v_{f0,1}\cdots v_{f0,n}$ in G'_{f0} and its reverse $v_{r1,n}\cdots v_{r1,1}$ in G'_{r1}. These two paths have the same length, and it is equal to the length of the longest path in G.

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



(a) Directed graph G



(b) Directed graph G'

Fig. 2. Transformation of a directed graph G for $LPSTP$ to a directed graph G' for $NPSP$. Edges in bold-green in both subfigures correspond to the solution path for $LPSTP$ and $NPSP$ problems respectively. **(a)** As the longest path exits s and enters t, dashed edges do not participate in the solution. **(b)** Blue dashed line between vertices in G'_{f0} and G'_{r1} visualise the reverse inverted fragments.

4 Computational Results

The described ILP formulation was implemented in Python3 using the PuLP package where we run Gurobi solver with academic licence. All the instances have been executed on a Linux laptop computer (32GB RAM, Intel® Core™ i7-10610U CPU @ 1.80GHz ×8).

Tables 2 and 3 respectively report results for real genomic and artificial data. In both cases, they share results categories: $|V|$ and $|E|$ are respectively the number of vertices and edges in MDCG; **time** (in second): average LP relaxation time (above), and B&B time (below); **opt**: optimal value for the LP relaxation (above), the integer solution (below); **%gap**: gap value of the linear relaxation bound with respect to the optimal integer value; **nodes**: number of B&B nodes exploration; **iter**: number of iterations for the LP relaxation (above) and for the B&B phase (below).

4.1 Genomic data

The method was tested on instances from biological data sets. When all links between contigs are provided, the computed solution enables to assemble the contigs and to output final sequences (fewer and longer contigs). As here the genomes are known, it is possible to asses our solution. We used for this purpose QUAST — a well known assembly evaluation tool [4]. Each instance has been run 10 times. Table 2 reports some solver run statistics and major QUAST measures.

The first observation concerns the excellent behavior of our model. The average gap of the linear relaxation bound (UB) with respect to the optimal integer value (Opt), computed as $100 \times \frac{(UB-Opt)}{UB}$ (MIP gap), differs from zero for only one of the reported instances. All the instances have been solved at the first B&B node and extremely faster.

The second observation concerns the quality of the obtained results (*c.f.* last two columns of Table 2). The known reference genomes are well covered by the fragments built by our approach (%gnm column) and the number of wrong contigs' succession choice (*misassemblies*, #mis column) is small.

The genomic instances suffer from particular difficulties (data noise and misses). The case of *Coffea arabica* was not solved because of data lack, hence absence of path between the source and target vertex. These issues are related with data generation tool and are beyond the scope of this paper.

4.2 Artificial data

In order to better analyze the algorithmic limit and the behavior of our linear model, we applied it to artificially generated data instances (contigs, links and multiplicities). Each instance has been run 5 times. The obtained results are reported in Table 3. They confirm the above mentioned characteristics of our integer formulation: the gap values stay extremely small even for the largest instance and all instances have been solved at the first B&B node.

5 Conclusion

This paper focusses on the scaffolding problem in the case of chloroplasts genome. We propose a directed graph that fits the specificities of the genomic sequences input data well. Working with the fact that the genome structure for this particular case is well known, we model the problem as a search for an elementary path with additional constraints for nested pairs location for a set of given vertices. We prove that the underlying optimisation problem, that we call Nested Pairs Genome Scaffolding is NP-Hard. We also design an integer programming formulation for this problem and apply our code for solving genomic and artificial data instances. The obtained results demonstrate that our formulation is very tight — the MIP gap value stays extremely small and we successfully solve very large instances at the first B&B node.

Concerning the quality of the results: biological data undergo several type of noises *e.g.* contigs were obtained from both plants DNA and chloroplasts

Table 2. Benchmark of chloroplast reference genomes. **instance**: species name; **size** (in base-pairs): genome length in number of nucleotides; **%gnm**: percentage of genome length covered by the solution computed by our approach; **#mis**: number of misassemblies in the proposed solution.

instance	size	$\|V\|$	$\|E\|$	time	opt	% gap	nodes	iter	% gnm	# mis
Aloysia	154699	42	74	$< .01$	9	0	0	181	99.85	0
citriodora				.04	9			242		
Altheaea	159987	10	24	-	-	0	0	-	99.91	0
officinalis				.01	1			0		
Amborella	162686	106	168	.03	17	0	0	604	98.82	0
trichopoda				.15	17			742		
Citrus	160101	174	358	.14	19	0	0	1741	80.32	2
limon				.3	19			2013		
Coffea	155188	134	222	-	-	-	0	-	-	-
arabica				.01				0		
Dendrobium	152018	36	54	$< .01$	5	0	1	90	97.77	2
nobile				.03	5			90		
Digitalis	153108	48	86	.01	13	0	0	350	83.15	0
lanata				.07	13			350		
Lupinus	151921	54	224	.05	15.5	3.23	1	928	99.98	0
albus				.76	15			3632		

Table 3. Benchmark on artificial data. **id**: instance's identifier; $U1$, IR (so $\overline{IR}$) and $U2$ are the regions illustrated in Subfigure 1a, and respectively correspond to purple, green left/right-side, and red regions. $|U1|$, $|IR|$ and $|U2|$ report number of contigs (above) and sum of contig's multiplicity (below), for respectively the purple region, the green regions (one version) and the red region.

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

DNA, and sequences suffer from sequencing errors *i.e.* some nucleotides can be substituted with others in the sequence. Further investigations are needed to deeper understand the impact of the sequencing technology on the quality of genome assembly. This is the subject of another study we are currently working on with experts of the field.

6 Acknowledgement

The authors are thankful to Dominique Lavenier, who exposed the NPSP problem to them and provided the data for the first benchmark. They are furthermore thankful to Sven Schrinner and Gunnar Klau for the valuable discussions about the problem's complexity.

References

1. Rumen Andonov, Hristo Djidjev, Sebastien François, and Dominique Lavenier. Complete assembly of circular and chloroplast genomes based on global optimization. *Journal of Bioinformatics and Computational Biology*, 17(3):1950014, June 2019.
2. Ralph Bock and Volker Knoop, editors. *Genomics of Chloroplasts and Mitochondria*, volume 35 of *Advances in Photosynthesis and Respiration.* Springer Netherlands, Dordrecht, 2012.
3. Sebastien François, Rumen Andonov, Dominique Lavenier, and Hristo Djidjev. Global Optimization for Scaffolding and Completing Genome Assemblies. *Electronic Notes in Discrete Mathematics*, 64:185–194, 2018.
4. Alexey Gurevich, Vladislav Saveliev, Nikolay Vyahhi, and Glenn Tesler. QUAST: quality assessment tool for genome assemblies. *Bioinformatics*, 29(8):1072–1075, April 2013.
5. Dan Gusfield. The RNA-Folding Problem. In Dan Gusfield, editor, *Integer Linear Programming in Computational and Systems Biology: An Entry-Level Text and Course*, pages 105–121. Cambridge University Press, Cambridge, 2019.
6. Alexander Schrijver. *Combinatorial Optimization: Polyhedra and Efficiency.* Springer Science & Business Media, February 2003. Google-Books-ID: mqGeSQ6dJycC.
7. Shanmugavel Senthilkumar, Kandasamy Ulaganathan, and Modhumita Ghosh Dasgupta. Reference-based assembly of chloroplast genome from leaf transcriptome data of Pterocarpus santalinus. *3 Biotech*, 11(8):393, August 2021.
8. Leonardo Taccari. Integer programming formulations for the elementary shortest path problem. *European Journal of Operational Research*, 252(1):122–130, July 2016.