Algorithms for Molecular Biology

RESEARCH

Open Access

Investigating the complexity of the double distance problems

Marília D. V. Braga[1], Leonie R. Brockmann[1], Katharina Klerx[1] and Jens Stoye[1*]

Abstract

Background Two genomes $\mathbb{A}$ and $\mathbb{B}$ over the same set of gene families form a *canonical* pair when each of them has exactly one gene from each family. Denote by n_* the number of common families of $\mathbb{A}$ and $\mathbb{B}$. Different distances of canonical genomes can be derived from a structure called *breakpoint graph*, which represents the relation between the two given genomes as a collection of cycles of even length and paths. Let c_i and p_j be respectively the numbers of cycles of length i and of paths of length j in the breakpoint graph of genomes $\mathbb{A}$ and $\mathbb{B}$. Then, the breakpoint distance of $\mathbb{A}$ and $\mathbb{B}$ is equal to $n_* - \left(c_2 + \frac{p_0}{2}\right)$. Similarly, when the considered rearrangements are those modeled by the *double-cut-and-join* (DCJ) operation, the rearrangement distance of $\mathbb{A}$ and $\mathbb{B}$ is $n_* - \left(c + \frac{p_e}{2}\right)$, where c is the total number of cycles and p_e is the total number of paths of even length.

Motivation The distance formulation is a basic unit for several other combinatorial problems related to genome evolution and ancestral reconstruction, such as *median* or *double distance*. Interestingly, both median and double distance problems can be solved in polynomial time for the breakpoint distance, while they are NP-hard for the rearrangement distance. One way of exploring the complexity space between these two extremes is to consider a σ_k distance, defined to be $n_* - \left(c_2 + c_4 + \ldots + c_k + \frac{p_0+p_2+\ldots+p_{k-2}}{2}\right)$, and increasingly investigate the complexities of median and double distance for the σ_4 distance, then the σ_6 distance, and so on.

Results While for the median much effort was done in our and in other research groups but no progress was obtained even for the σ_4 distance, for solving the double distance under σ_4 and σ_6 distances we could devise linear time algorithms, which we present here.

Keywords Comparative genomics, Genome rearrangement, Breakpoint distance, Double-cut-and-join (DCJ) distance, Double distance

Introduction

In genome comparison, the most elementary problem is that of computing a *distance* between two given genomes [1], each one being a set of *chromosomes*. Usually a high-level view of a chromosome is adopted, in which each chromosome is represented by a sequence of oriented *genes* and the genes are classified into *families*. The simplest model in this setting is the *breakpoint* model, whose distance consists of somehow quantifying the distinct *adjacencies* between the two genomes, an *adjacency* in a genome being the oriented neighborhood between two genes in one of its chromosomes [2]. Other models rely on large-scale genome *rearrangements*, such as inversions, translocations, fusions and fissions, yielding distances that correspond to the minimum number of rearrangements required to transform one genome into another [3–5].

*Correspondence:
Jens Stoye
jens.stoye@uni-bielefeld.de
[1] Faculty of Technology and Center for Biotechnology (CeBiTec), Bielefeld University, Bielefeld, Germany

Independently of the underlying model, the distance formulation is a basic unit for several other combinatorial problems related to genome evolution and ancestral reconstruction [2]. The *median* problem, for example, has three genomes as input and asks for an ancestor genome that minimizes the sum of its distances to the three given genomes. Other models are related to the *whole genome duplication* (WGD) event [6]. Let the *doubling* of a genome duplicate each of its chromosomes. The *double distance* is the problem that has a *duplicated* genome and a *singular* genome as input and computes the distance between the former and a doubling of the latter. The *halving* problem has a duplicated genome as input and asks for a singular genome whose double distance to the given duplicated genome is minimized. Finally, the *guided halving* problem has a duplicated and a singular genome as input and asks for another singular genome that minimizes the sum of its double distance to the given duplicated genome and its distance to the given singular genome.

Our study relies on the *breakpoint graph*, a structure that represents the relation between two given genomes [7]. When the two genomes are over the same set of gene families and form a *canonical* pair, that is, when each of them has exactly one gene from each family, their breakpoint graph is a collection of cycles of even length and paths. Assuming that both genomes have n_* genes, if we call *k-cycle* a cycle of length k and *k-path* a path of length k, the corresponding breakpoint distance is equal to $n_* - \left(c_2 + \frac{p_0}{2}\right)$, where c_2 is the number of 2-cycles and p_0 is the number of 0-paths [2]. Similarly, when the considered rearrangements are those modeled by the *double-cut-and-join* (DCJ) operation [5], the rearrangement distance is $n_* - \left(c + \frac{p_e}{2}\right)$, where c is the total number of cycles and p_e is the total number of even paths [8].

While the halving problem under both breakpoint and rearrangement distances can be solved in polynomial time [2, 6, 9, 10], median, double distance and guided halving problems can be solved in polynomial time only under the breakpoint distance, but are NP-hard under the rearrangement distance [2]. One way of exploring the complexity space between these two extremes is to consider a σ_k distance [11], defined to be $n_* - \left(c_2 + c_4 + \ldots + c_k + \frac{p_0+p_2+\ldots+p_{k-2}}{2}\right)$, and increasingly investigate the complexities of median, guided halving and double distance under the σ_4 distance, then under the σ_6 distance, and so on. Note that the σ_2 distance is the breakpoint distance and the σ_∞ distance is the DCJ distance. To the best of our knowledge, the guided halving problem has not been studied for this class of problems, while for the median under σ_4 distance much effort has been done in our group and in other research groups (e.g. [11]) but no progress was obtained so far.

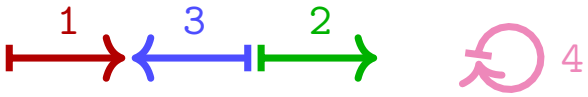

Fig. 1 Representation of linear chromosome $[1\ \overline{3}\ 2]$ and circular chromosome (4)

In contrast, for the double distance, while σ_8 and higher were not yet studied, we succeeded in devising efficient algorithms for σ_4 and σ_6. Our results, which we present here, are built on a variation of the breakpoint graph, called *ambiguous breakpoint graph* [2] and have three main parts. First we show that in any σ_k double distance, including the NP-hard DCJ double distance, all 2-cycles and 0-paths are fulfilled, meaning that the common adjacencies and common telomeres between the compared genomes are always conserved. Then we show that the σ_4 double distance can be computed by a greedy linear time algorithm. Finally we present a non-greedy but still linear time algorithm for the σ_6 double distance.

This paper is an extended version of our two recent works, one restricted to genomes composed exclusively of circular chromosomes [12], whose solution is straightforward, and the second allowing linear chromosomes[13], which is more intricate but can be reduced to particular linear-time solvable instances of the *maximum independent set problem*. Besides putting together the results mentioned above, here we give more details and illustrations concerning the special situations that arose from the inclusion of linear chromosomes and the related instances of the maximum independent set problem.

Background

A *chromosome* is an oriented DNA molecule and can be either linear or circular. We represent a chromosome by its sequence of genes, where each *gene* is an oriented DNA fragment. We assume that each gene belongs to a *family*, which is a set of homologous genes. A gene that belongs to a family $\mathtt{X}$ is represented by the symbol $\mathtt{X}$ itself if it is read in forward orientation or by the symbol $\overline{\mathtt{X}}$ if it is read in reverse orientation. For example, the sequences $[1\ \overline{3}\ 2]$ and (4) represent, respectively, a linear (flanked by square brackets) and a circular chromosome (flanked by parentheses), both shown in Fig. 1, the first composed of three genes and the second composed of a single gene. Note that if a sequence s represents a chromosome K, then K can be equally represented by the *reverse complement* of s, denoted by $\bar{s}$, obtained by reversing the order and the orientation of the genes in s. Moreover, if K is circular, it can be equally represented by any circular

Table 1 Examples of a singular, a duplicated and two doubled genomes, with their sets of families and their multisets of adjacencies. Note that the doubled genomes $\mathbb{B}_1$ and $\mathbb{B}_2$ have exactly the same adjacencies and telomeres

Singular genome (each family occurs once)	$\mathbb{S}=\{(1\ \overline{3}\ 2)\ (4)\ [5\ \overline{6}]\}$	$\begin{cases}\mathcal{F}(\mathbb{S})=\{1,2,3,4,5,6\}\\ \mathcal{A}(\mathbb{S})=\{1^h3^h, 3^t2^t, 2^h1^t, 4^h4^t, 5^h6^h\}\\ \mathcal{T}(\mathbb{S})=\{5^t, 6^t\}\end{cases}$
Duplicated genome (each family occurs twice)	$\mathbb{D}=\{(1\ 2\ \overline{3}\ 1)\ [\overline{3}\ 2]\}$	$\begin{cases}\mathcal{F}(\mathbb{D})=\{1,2,3\}\\ \mathcal{A}(\mathbb{D})=\{1^h2^t, 2^h3^h, 3^t1^t, 1^h1^t, 3^t2^t\}\\ \mathcal{T}(\mathbb{D})=\{3^h, 2^h\}\end{cases}$
Doubled genomes (each adjacency or telomere occurs twice)	$\mathbb{B}_1=\{(1\ 2)\ (1\ 2)\ [3\ 4]\ [3\ 4]\}$ $\mathbb{B}_2=\{(1\ 2\ 1\ 2)\ [3\ 4]\ [3\ 4]\}$	$\begin{cases}\mathcal{F}(\mathbb{B}_i)=\{1,2,3,4\}\\ \mathcal{A}(\mathbb{B}_i)=\{1^h2^t, 2^h1^t, 1^h2^t, 2^h1^t, 3^h4^t, 3^h4^t\}\\ \mathcal{T}(\mathbb{B}_i)=\{3^t, 4^h, 3^t, 4^h\}\end{cases}$

rotation of s or of $\bar{s}$. Recall that a gene is an *occurrence* of a family, therefore distinct genes from the same family are represented by the same symbol.

We can also represent a gene from family $\mathtt{X}$ referring to its *extremities* $\mathtt{X}^h$ (head) and $\mathtt{X}^t$ (tail). The *adjacencies* in a chromosome are the neighboring extremities of distinct genes. The remaining extremities, that are at the ends of linear chromosomes, are *telomeres*. In linear chromosome $[1\ \overline{3}\ 2]$, the adjacencies are $\{1^h3^h, 3^t2^t\}$ and the telomeres are $\{1^t, 2^h\}$. Note that an adjacency has no orientation, that is, an adjacency between extremities 1^h and 3^h can be equally represented by 1^h3^h and by 3^h1^h. In the particular case of a single-gene circular chromosome, e.g. (4), an adjacency exceptionally occurs between the extremities of the same gene (here 4^h4^t).

A *genome* is then a multiset of chromosomes and we denote by $\mathcal{F}(\mathbb{G})$ the set of gene families that occur in genome $\mathbb{G}$. In addition, we denote by $\mathcal{A}(\mathbb{G})$ the multiset of adjacencies and by $\mathcal{T}(\mathbb{G})$ the multiset of telomeres that occur in $\mathbb{G}$. A genome $\mathbb{S}$ is called *singular* if each gene family occurs exactly once in $\mathbb{S}$. Similarly, a genome $\mathbb{D}$ is called *duplicated* if each gene family occurs exactly twice in $\mathbb{D}$. The two occurrences of a family in a duplicated genome are called *paralogs*. A *doubled* genome is a special type of duplicated genome in which each adjacency or telomere occurs exactly twice. These two copies of the same adjacency (respectively same telomere) in a doubled genome are called *paralogous adjacencies* (respectively *paralogous telomeres*). Observe that distinct doubled genomes with circular chromosomes can have exactly the same adjacencies and telomeres, as we show in Table 1, where we also give examples of singular and duplicated genomes.

Comparing canonical genomes

Two genomes $\mathbb{S}_1$ and $\mathbb{S}_2$ are said to be a *canonical pair* when they are singular and have the same gene families, that is, $\mathcal{F}(\mathbb{S}_1)=\mathcal{F}(\mathbb{S}_2)$. Denote by $\mathcal{F}_*$ the set of families occurring in canonical genomes $\mathbb{S}_1$ and $\mathbb{S}_2$, and by $n_*=|\mathcal{F}_*|$ its cardinality. For example, genomes $\mathbb{S}_1=\{(1\ \overline{3}\ 2)\ (4)\}$ and $\mathbb{S}_2=\{(1\ 2)\ (3\ \overline{4})\}$ are canonical with $\mathcal{F}_*=\{1,2,3,4\}$ and $n_*=4$.

Breakpoint graph

The relation between two canonical genomes $\mathbb{S}_1$ and $\mathbb{S}_2$ can be represented by their *breakpoint graph* $BG(\mathbb{S}_1,\mathbb{S}_2)=(V,E)$, that is a multigraph representing the adjacencies of $\mathbb{S}_1$ and $\mathbb{S}_2$ [7]. The vertex set V comprises, for each family $\mathtt{X}$ in $\mathcal{F}_*$, one vertex for the extremity $\mathtt{X}^h$ and one vertex for the extremity $\mathtt{X}^t$. The edge multiset E represents the adjacencies. For each adjacency in $\mathbb{S}_1$ there exists one $\mathbb{S}_1$-edge in E linking its two extremities. Similarly, for each adjacency in $\mathbb{S}_2$ there exists one $\mathbb{S}_2$-edge in E linking its two extremities. Clearly, $BG(\mathbb{S}_1,\mathbb{S}_2)$ can easily be constructed in linear $O(n_*)$ time.

The degree of each vertex can be 0, 1 or 2 and each connected *component* alternates between $\mathbb{S}_1$- and $\mathbb{S}_2$-edges. As a consequence, the components of the breakpoint graph of canonical genomes can be cycles of even length or paths. An even path has one endpoint in $\mathbb{S}_1$ ($\mathbb{S}_1$-*telomere*) and the other in $\mathbb{S}_2$ ($\mathbb{S}_2$-*telomere*), while an odd path has either both endpoints in $\mathbb{S}_1$ or both endpoints in $\mathbb{S}_2$. A vertex that is not a telomere in $\mathbb{S}_1$ nor in $\mathbb{S}_2$ is said to be *non-telomeric*. In the breakpoint graph a non-telomeric vertex has degree 2. We call *i-cycle* a cycle of length i and *j-path* a path of length j. We also denote by c_i the number of i-cycles, by p_j the number of j-paths, by c the total number of cycles and by p_e the total number of even paths. Since the number of telomeres in each genome is even (2 telomeres per linear chromosome), the total number of even paths in the breakpoint graph must be even. An example of a breakpoint graph is given in Fig. 2.

Breakpoint distance

For canonical genomes $\mathbb{S}_1$ and $\mathbb{S}_2$ the *breakpoint distance*, denoted by d_{BP}, is defined as follows [2]:

$$d_{\text{BP}}(\mathbb{S}_1, \mathbb{S}_2) = n_* - \left(|\mathcal{A}(\mathbb{S}_1) \cap \mathcal{A}(\mathbb{S}_2)| + \frac{|\mathcal{T}(\mathbb{S}_1) \cap \mathcal{T}(\mathbb{S}_2)|}{2} \right).$$

For $\mathbb{S}_1 = \{(1\ \overline{3}\ 2)\ [4]\}$ and $\mathbb{S}_2 = \{(1\ 2)\ [3\ \overline{4}]\}$, we have $n_* = 4$. The set of common adjacencies is $\mathcal{A}(\mathbb{S}_1) \cap \mathcal{A}(\mathbb{S}_2) = \{1^t 2^h\}$ and the set of common telomeres is $\mathcal{T}(\mathbb{S}_1) \cap \mathcal{T}(\mathbb{S}_2) = \{4^t\}$, giving $d_{\text{BP}}(\mathbb{S}_1, \mathbb{S}_2) = 2.5$. Since a common adjacency of $\mathbb{S}_1$ and $\mathbb{S}_2$ corresponds to a 2-cycle and a common telomere corresponds to a 0-path in $BG(\mathbb{S}_1, \mathbb{S}_2)$, the breakpoint distance can be rewritten as

$$d_{\text{BP}}(\mathbb{S}_1, \mathbb{S}_2) = n_* - \left(c_2 + \frac{p_0}{2} \right).$$

DCJ distance

Given a genome, a *double cut and join* (DCJ) is the operation that breaks two of its adjacencies or telomeres[1] and rejoins the open extremities in a different way [5]. For example, consider the chromosome $K = [\ 1\ 2\ 3\ 4\ ]$ and a DCJ that cuts K between genes 1 and 2 and between genes 3 and 4, creating segments $1\bullet$, $\bullet 2\ 3\bullet$ and $\bullet 4$ (where the symbols $\bullet$ represent the open ends). If we join the first with the third and the second with the fourth open end, we get $K' = [\ 1\ \overline{3}\ \overline{2}\ 4\ ]$, that is, the described DCJ operation is an inversion transforming K into K'. Besides inversions, DCJ operations can represent several rearrangements, such as translocations, fissions and fusions. The *DCJ distance* d_{DCJ} is then the minimum number of DCJs that transform one genome into the other and can be easily computed with the help of their breakpoint graph [8]:

$$\begin{aligned} d_{\text{DCJ}}(\mathbb{S}_1, \mathbb{S}_2) &= n_* - \left(c + \frac{p_e}{2} \right) \\ &= n_* - \left(c_2 + c_4 + \ldots + c_\infty + \frac{p_0 + p_2 + \ldots + p_\infty}{2} \right). \end{aligned}$$

If $\mathbb{S}_1 = \{(1\ \overline{3}\ 2)\ [4]\}$ and $\mathbb{S}_2 = \{(1\ 2)\ [3\ \overline{4}]\}$, then $n_* = 4$, $c = 1$ and $p_e = 2$ (see Fig. 2). Consequently, their DCJ distance is $d_{\text{DCJ}}(\mathbb{S}_1, \mathbb{S}_2) = 2$.

The class of σ_k distances

Given the breakpoint graph of two canonical genomes $\mathbb{S}_1$ and $\mathbb{S}_2$, for $k \in \{2, 4, 6, \ldots, \infty\}$, we denote by σ_k the cumulative sums $\sigma_k = c_2 + c_4 + \ldots + c_k + \frac{p_0 + p_2 + \ldots + p_{k-2}}{2}$. Then the σ_k distance of $\mathbb{S}_1$ and $\mathbb{S}_2$ is defined to be [11]:

$$d_{\sigma_k}(\mathbb{S}_1, \mathbb{S}_2) = n_* - \sigma_k.$$

It is easy to see that the σ_2 distance equals the breakpoint distance and that the σ_∞ distance equals the DCJ distance, and that the distance decreases monotonously between these two extremes. Moreover, the σ_k distance of two genomes that form a canonical pair can easily be computed in linear time for any $k \geq 2$.

Comparing a singular and a duplicated genome

Let $\mathbb{S}$ be a singular and $\mathbb{D}$ be a duplicated genome over the same n_* gene families, that is, $\mathcal{F}(\mathbb{S}) = \mathcal{F}(\mathbb{D})$ and $n_* = |\mathcal{F}(\mathbb{S})| = |\mathcal{F}(\mathbb{D})|$. The number of genes in $\mathbb{D}$ is twice the number of genes in $\mathbb{S}$ and we need to somehow equalize the contents of these genomes, before searching for common adjacencies and common telomeres of $\mathbb{S}$ and $\mathbb{D}$ or transforming one genome into the other with DCJ operations. This can be done by *doubling* $\mathbb{S}$, with a rearrangement operation mimicking a *whole genome duplication*: it simply consists of doubling each adjacency and each telomere of $\mathbb{S}$. However, when $\mathbb{S}$ has one or more circular chromosomes, it is not possible to find a unique layout of its chromosomes after the doubling: indeed, each circular chromosome can be doubled into two identical circular chromosomes, or the two copies are concatenated to each other in a single circular chromosome. Therefore, in general the doubling of a genome $\mathbb{S}$ results in a set of doubled genomes denoted by $2\mathbb{S}$. Note that $|2\mathbb{S}| = 2^r$, where r is the number of circular chromosomes in $\mathbb{S}$. For example, if $\mathbb{S} = \{(1\ 2)\ [3\ 4]\}$, then $2\mathbb{S} = \{\mathbb{B}_1, \mathbb{B}_2\}$ with $\mathbb{B}_1 = \{(1\ 2)\ (1\ 2)\ [3\ 4]\ [3\ 4]\}$ and $\mathbb{B}_2 = \{(1\ 2\ 1\ 2)\ [3\ 4]\ [3\ 4]\}$ (see Table 1). All genomes in $2\mathbb{S}$ have exactly the same multisets of adjacencies and of telomeres, therefore we can use a special notation for these multisets: $\mathcal{A}(2\mathbb{S}) = \mathcal{A}(\mathbb{S}) \cup \mathcal{A}(\mathbb{S})$ and $\mathcal{T}(2\mathbb{S}) = \mathcal{T}(\mathbb{S}) \cup \mathcal{T}(\mathbb{S})$.

Each family in a duplicated genome can be $\binom{\mathtt{a}}{\mathtt{b}}$-*singularized* by adding the index $\mathtt{a}$ to one of its occurrences and the index $\mathtt{b}$ to the other. A duplicated genome can be entirely singularized if each of its families is singularized. Let $\mathfrak{S}^{\mathtt{a}}_{\mathtt{b}}(\mathbb{D})$ be the set of all possible genomes obtained by all distinct ways of $\binom{\mathtt{a}}{\mathtt{b}}$-singularizing the duplicated genome $\mathbb{D}$. Similarly, we denote by $\mathfrak{S}^{\mathtt{a}}_{\mathtt{b}}(2\mathbb{S})$ the set of all possible genomes obtained by all distinct ways of $\binom{\mathtt{a}}{\mathtt{b}}$-singularizing each doubled genome in the set $2\mathbb{S}$.

[1] A broken adjacency has two open ends and a broken telomere has a single one.

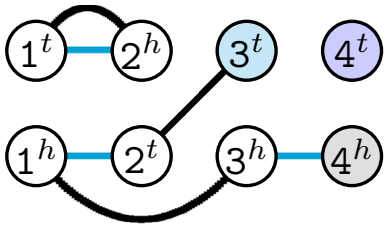

Fig. 2 Breakpoint graph of genomes $\mathbb{S}_1 = \{(1\ 2)\ [3\ \overline{4}]\}$ and $\mathbb{S}_2 = \{(1\ \overline{3}\ 2)\ [4]\}$. Edge types are distinguished by colors: $\mathbb{S}_1$-edges are drawn in blue and $\mathbb{S}_2$-edges are drawn in black. Similarly, vertex types are distinguished by colors: an $\mathbb{S}_1$-telomere is marked in blue, an $\mathbb{S}_2$-telomere is marked in gray, a telomere in both $\mathbb{S}_1$ and $\mathbb{S}_2$ is marked in purple and non-telomeric vertices are white. This graph has one 2-cycle, one 0-path and one 4-path

The class of σ_k double distances

The class of σ_k double distances of a singular genome $\mathbb{S}$ and duplicated genome $\mathbb{D}$ for $k = 2, 4, 6, \ldots$ is defined as follows:

$$d^2_{\sigma_k}(\mathbb{S},\mathbb{D}) = d^2_{\sigma_k}(\mathbb{S},\breve{\mathbb{D}}) = \min_{\mathbb{B}\in\mathfrak{S}^{\mathtt{a}}_{\mathtt{b}}(2\mathbb{S})}\{d_{\sigma_k}(\mathbb{B},\breve{\mathbb{D}})\}, \text{where } \breve{\mathbb{D}} \text{ is any genome in } \mathfrak{S}^{\mathtt{a}}_{\mathtt{b}}(\mathbb{D}).$$

Observe that $d^2_{\sigma_k}(\mathbb{S},\breve{\mathbb{D}}) = d^2_{\sigma_k}(\mathbb{S},\breve{\mathbb{D}}')$ for any $\breve{\mathbb{D}}, \breve{\mathbb{D}}' \in \mathfrak{S}^{\mathtt{a}}_{\mathtt{b}}(\mathbb{D})$.

σ_2 (breakpoint) double distance

The *breakpoint double distance* of $\mathbb{S}$ and $\mathbb{D}$, denoted by $d^2_{\text{BP}}(\mathbb{S},\mathbb{D})$, is equivalent to the σ_2 double distance. For this case the solution can be found easily with a greedy algorithm [2]: each adjacency or telomere of $\mathbb{D}$ that occurs in $\mathbb{S}$ can be fulfilled. If an adjacency or telomere that occurs twice in $\mathbb{D}$ also occurs in $\mathbb{S}$, it can be fulfilled twice in any genome from $2\mathbb{S}$. Then,

$$d^2_{\text{BP}}(\mathbb{S},\mathbb{D}) = 2n_* - |\mathcal{A}(2\mathbb{S}) \cap \mathcal{A}(\mathbb{D})| - \frac{|\mathcal{T}(2\mathbb{S}) \cap \mathcal{T}(\mathbb{D})|}{2}.$$

σ_∞ (DCJ) double distance

For the *DCJ double distance*, that is equivalent to the σ_∞ double distance, the solution space cannot be explored greedily. In fact, computing the DCJ double distance of genomes $\mathbb{S}$ and $\mathbb{D}$ was proven to be an NP-hard problem [2].

The complexity of σ_k double distances

The exploration of the complexity space between the greedy linear time σ_2 (breakpoint) double distance and the NP-hard σ_∞ (DCJ) double distance is the main motivation of this study. In the remainder of this paper we show that both σ_4 and σ_6 double distances can be solved in linear time.

Equivalence of σ_k double distance and σ_k disambiguation

A nice way of representing the solution space of the σ_k double distance is by using a modified version of the breakpoint graph [2].

Ambiguous breakpoint graph

Given a singular genome $\mathbb{S}$ and a duplicated genome $\mathbb{D}$, their *ambiguous breakpoint graph* $ABG(\mathbb{S},\breve{\mathbb{D}}) = (V, E)$ is a multigraph representing the adjacencies of any element in $\mathfrak{S}^{\mathtt{a}}_{\mathtt{b}}(2\mathbb{S})$ and a genome $\breve{\mathbb{D}} \in \mathfrak{S}^{\mathtt{a}}_{\mathtt{b}}(\mathbb{D})$. The vertex set V comprises, for each family $\mathtt{X}$ in $\mathcal{F}(\mathbb{S})$, the two pairs of *paralogous vertices* $\mathtt{X}^h_{\mathtt{a}}$, $\mathtt{X}^h_{\mathtt{b}}$ and $\mathtt{X}^t_{\mathtt{a}}$, $\mathtt{X}^t_{\mathtt{b}}$. We can use the notation $\hat{u}$ to refer to the paralogous counterpart of a vertex u. For example, if $u = \mathtt{X}^h_{\mathtt{a}}$, then $\hat{u} = \mathtt{X}^h_{\mathtt{b}}$.

The edge set E represents the adjacencies. For each adjacency in $\breve{\mathbb{D}}$ there exists one $\breve{\mathbb{D}}$-edge in E linking its two extremities. The $\mathbb{S}$-edges represent all adjacencies occurring in all genomes from $\mathfrak{S}^{\mathtt{a}}_{\mathtt{b}}(2\mathbb{S})$: for each adjacency $\gamma\beta$ of $\mathbb{S}$, we have the *pair of paralogous edges* $\mathcal{E}(\gamma\beta) = \{\gamma_{\mathtt{a}}\beta_{\mathtt{a}}, \gamma_{\mathtt{b}}\beta_{\mathtt{b}}\}$ and the *complementary pair of paralogous edges* $\widetilde{\mathcal{E}}(\gamma\beta) = \{\gamma_{\mathtt{a}}\beta_{\mathtt{b}}, \gamma_{\mathtt{b}}\beta_{\mathtt{a}}\}$. Note that $\widetilde{\mathcal{E}}(\gamma\beta) = \mathcal{E}(\gamma\beta)$. The *square* of $\gamma\beta$ is then $\mathcal{Q}(\gamma\beta) = \mathcal{E}(\gamma\beta) \cup \widetilde{\mathcal{E}}(\gamma\beta)$. The $\mathbb{S}$-edges in the ambiguous breakpoint graph are therefore the squares of all adjacencies in $\mathbb{S}$. Let a_* be the number of squares in $ABG(\mathbb{S},\breve{\mathbb{D}})$. Obviously we have $a_* = |\mathcal{A}(\mathbb{S})| = n_* - \kappa(\mathbb{S})$, where $\kappa(\mathbb{S})$ is the number of linear chromosomes in $\mathbb{S}$. Again, we can use the notation $\hat{e}$ to refer to the paralogous counterpart of an $\mathbb{S}$-edge e. For example, if $e = \gamma_{\mathtt{a}}\beta_{\mathtt{a}}$, then $\hat{e} = \gamma_{\mathtt{b}}\beta_{\mathtt{b}}$. An example of an ambiguous breakpoint graph is shown in Fig. 3 (i).

Each linear chromosome in $\mathbb{S}$ corresponds to four telomeres, called $\mathbb{S}$-*telomeres*, in any element of $2\mathbb{S}$. These four vertices are not part of any square. In other words, the number of $\mathbb{S}$-telomeres in $ABG(\mathbb{S},\breve{\mathbb{D}})$ is $4\kappa(\mathbb{S})$. If $\kappa(\mathbb{D})$ is the number of linear chromosomes in $\mathbb{D}$, the number of telomeres in $\breve{\mathbb{D}}$, also called $\breve{\mathbb{D}}$-*telomeres*, is $2\kappa(\mathbb{D})$.

The class of σ_k disambiguations

Resolving a square $\mathcal{Q}(\cdot) = \mathcal{E}(\cdot) \cup \widetilde{\mathcal{E}}(\cdot)$ corresponds to *choosing* in the ambiguous breakpoint graph either the edges from $\mathcal{E}(\cdot)$ or the edges from $\widetilde{\mathcal{E}}(\cdot)$, while the complementary pair is *masked*. Resolving all squares is called *disambiguating* the ambiguous breakpoint graph. If we number the squares of $ABG(\mathbb{S},\breve{\mathbb{D}})$ from 1 to a_*, a *solution* can be represented by a tuple $\tau = (\mathcal{L}_1, \mathcal{L}_2, \ldots, \mathcal{L}_{a_*})$, where each $\mathcal{L}_i$ contains the pair of paralogous edges

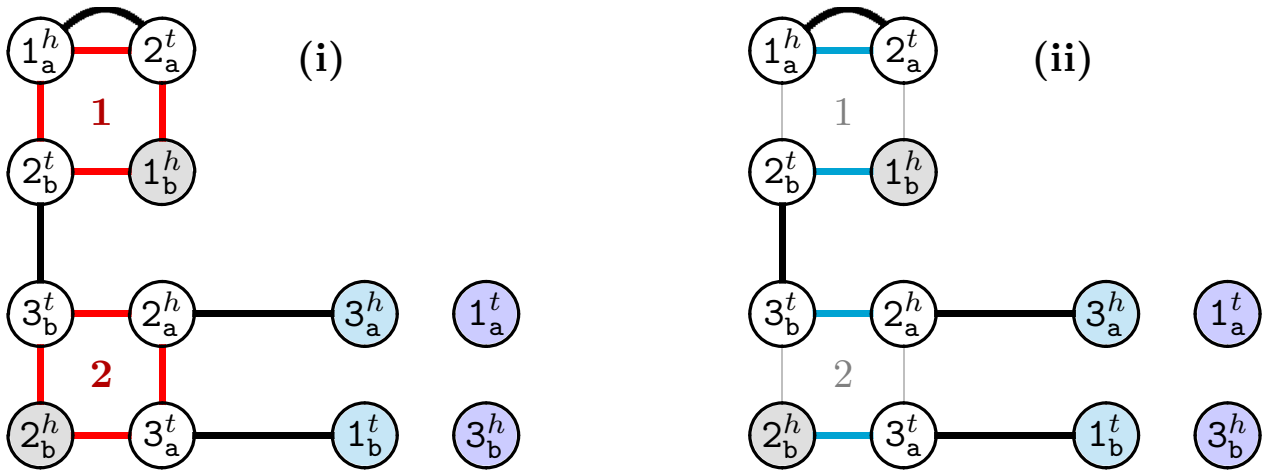

Fig. 3 **i** Ambiguous breakpoint graph $ABG(\mathbb{S}, \breve{\mathbb{D}})$ for genomes $\mathbb{S} = \{[1\ 2\ 3]\}$ and $\breve{\mathbb{D}} = \{[1_a\ 2_a\ \overline{3}_a\ 1_b]\,[\overline{3}_b\ 2_b]\}$. Edge types are distinguished by colors: $\breve{\mathbb{D}}$-edges are drawn in black and $\mathbb{S}$-edges (squares) are drawn in red. **ii** Induced breakpoint graph $BG(\tau, \breve{\mathbb{D}})$ in which all squares are resolved by the solution $\tau = (\{1^h_a 2^t_a, 1^h_b 2^t_b\}, \{2^h_a 3^t_b, 2^h_b 3^t_a\})$, resulting in one 2-cycle, two 0-paths, one 2-path and one 4-path. This is also the breakpoint graph of $\breve{\mathbb{D}}$ and $\mathbb{B} = \{[1_a\ 2_a\ 3_b], [1_b\ 2_b\ 3_a]\} \in \mathfrak{S}^a_b(2\mathbb{S})$. In both **i** and **ii**, vertex types are distinguished by colors: telomeres in $\mathbb{S}$ are marked in blue, telomeres in $\breve{\mathbb{D}}$ are marked in gray, telomeres in both $\mathbb{S}$ and $\breve{\mathbb{D}}$ are marked in purple and non-telomeric vertices are white

(either $\mathcal{E}_i$ or $\widetilde{\mathcal{E}_i}$) that are chosen (kept) in the graph for square $\mathcal{Q}_i$. The graph *induced* by τ is a simple breakpoint graph, which we denote by $BG(\tau, \breve{\mathbb{D}})$. Figure 3 (ii) shows an example.

Given a solution τ, let c_i and p_j be, respectively, the number of cycles of length i and of paths of length j in $BG(\tau, \breve{\mathbb{D}})$. The k-score of τ is then the sum $\sigma_k = c_2 + c_4 + \ldots + c_k + \frac{p_0+p_2+\ldots+p_{k-2}}{2}$. The minimization problem of computing the σ_k double distance of $\mathbb{S}$ and $\mathbb{D}$ is equivalent to finding a solution τ so that the k-score of τ is maximized [2]. We call the latter (maximization) problem σ_k *disambiguation*. As already mentioned, for σ_2 the double distance can be solved in linear time and for σ_∞ the double distance is NP-hard. Therefore the same is true, respectively, for the σ_2 and the σ_∞ disambiguations. Conversely, if we determine the complexity of solving the σ_k disambiguation for any $k \geq 4$, this will automatically determine the complexity of solving the σ_k double distance.

An optimal solution for the σ_k disambiguation of $ABG(\mathbb{S}, \breve{\mathbb{D}})$ gives its *k-score*, denoted by $\sigma_k(ABG(\mathbb{S}, \breve{\mathbb{D}}))$. Note that, since an optimal σ_k disambiguation is also a σ_{k+2} disambiguation, although possibly not optimal, the k-score of $ABG(\mathbb{S}, \breve{\mathbb{D}})$ can not decrease as k increases.

Approach for solving the σ_k disambiguation

A *player* of the σ_k disambiguation is either a valid cycle whose length is at most k or a valid even path whose length is at most $k-2$. In order to solve the σ_k disambiguation, a natural approach is to visit $ABG(\mathbb{S}, \breve{\mathbb{D}})$ and search for players. For describing how the graph can be screened, we need to introduce the following concepts. Two $\mathbb{S}$-edges in $ABG(\mathbb{S}, \breve{\mathbb{D}})$ are *incompatible* when they belong to the same square and are not paralogous. A component in $ABG(\mathbb{S}, \breve{\mathbb{D}})$ is *valid* when it does not contain any pair of incompatible edges. Note that a valid component necessarily alternates $\mathbb{S}$-edges and $\breve{\mathbb{D}}$-edges. Two valid components $C \neq C'$ in $ABG(\mathbb{S}, \breve{\mathbb{D}})$ are either *intersecting*, when they share at least one vertex, or *disjoint*. It is obvious that any solution τ of $ABG(\mathbb{S}, \breve{\mathbb{D}})$ is composed of disjoint valid components.

Given a solution $\tau = (\mathcal{L}_1, \mathcal{L}_2, \ldots, \mathcal{L}_i \ldots, \mathcal{L}_{a_*})$, the *switching* operation of the i-th element of τ is denoted by $\widetilde{s}(\tau, i)$ and replaces value $\mathcal{L}_i$ by $\widetilde{\mathcal{L}_i}$ resulting in $\tau' = (\mathcal{L}_1, \mathcal{L}_2, \ldots, \widetilde{\mathcal{L}_i} \ldots, \mathcal{L}_{a_*})$. A choice of paralogous edges resolving a given square $\mathcal{Q}_i$ can be *fixed* for any solution, meaning that $\mathcal{Q}_i$ can no longer be switched. In this case, $\mathcal{Q}_i$ is itself said to be *fixed*.

First steps to solve the σ_k disambiguation

In this section we describe a greedy linear time algorithm for the σ_4 disambiguation and give some general results related to any σ_k disambiguation.

Common adjacencies and telomeres are conserved

Let τ be an optimal solution for σ_k disambiguation of $ABG(\mathbb{S}, \breve{\mathbb{D}})$. If a player $C \in BG(\tau, \breve{\mathbb{D}})$ is disjoint from any

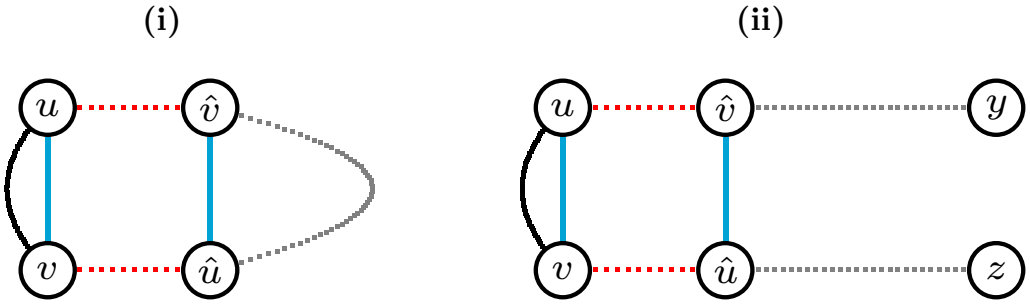

Fig. 4 (**i**) The gray path connecting vertex $\hat{v}$ to vertex $\hat{u}$ is necessarily odd with length at least one and alternates $\check{\mathbb{D}}$- and $\mathbb{S}$-edges. The 2-cycle $C = (uv)$ intersects the longer cycle $D = (u\hat{v} \ldots \hat{u}v)$. Any solution containing (red edges) $\widetilde{\mathcal{E}} = \{u\hat{v}, \hat{u}v\}$ induces D and can be improved by switching $\widetilde{\mathcal{E}}$ to (blue edges) $\mathcal{E} = \{uv, \hat{u}\hat{v}\}$, inducing, instead of D, the 2-cycle C and cycle $D' = (\hat{v} \ldots \hat{u})$ (which is shorter than D). (**ii**) The gray paths connecting vertex $\hat{v}$ to telomere y and vertex $\hat{u}$ to telomere z alternate $\check{\mathbb{D}}$- and $\mathbb{S}$-edges. The 2-cycle $C = (uv)$ intersects the longer path $P = y \ldots \hat{v}uv\hat{u} \ldots z$. Any solution containing (red edges) $\widetilde{\mathcal{E}} = \{u\hat{v}, \hat{u}v\}$ induces P and can be improved by switching $\widetilde{\mathcal{E}}$ to (blue edges) $\mathcal{E} = \{uv, \hat{u}\hat{v}\}$, inducing, instead of P, the 2-cycle C and path $P' = y \ldots \hat{v}\hat{u} \ldots z$ (which is of the same type, but 2 edges shorter than P)

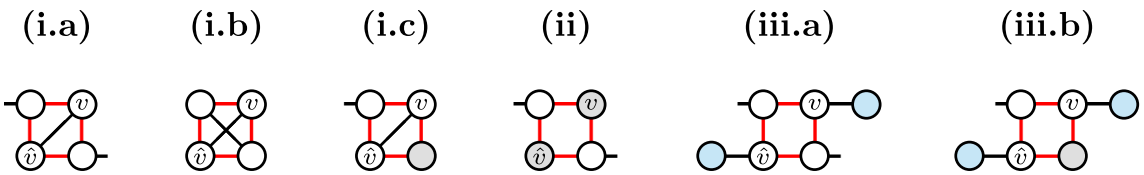

Fig. 5 Possible symmetric squares in the ambiguous breakpoint graph

player distinct from C in any other optimal solution, then C must be part of all optimal solutions and is itself said to be *optimal*.

Lemma 1 *For any σ_k disambiguation, all existing 0-paths and 2-cycles in $ABG(\mathbb{S}, \mathbb{D})$ are optimal.*

Proof While any 0-path is an isolated vertex and obviously optimal, the optimality of every 2-cycle is less obvious but still holds, as illustrated in Fig. 4. □

This lemma is a generalization of the (breakpoint) σ_2 disambiguation and guarantees that all common adjacencies and telomeres are conserved in any σ_k double distance, including the NP-hard (DCJ) σ_∞ case. All 0-paths are isolated vertices that do not integrate squares, therefore they are selected independently of the choices for resolving the squares. A 2-cycle, in its turn, always includes one $\mathbb{S}$-edge from some square (such as square 1 in Fig. 3). From now on we assume that squares that have at least one $\mathbb{S}$-edge in a 2-cycle are fixed so that all existing 2-cycles are induced.

Symmetric squares can be fixed arbitrarily

Let a *symmetric square* in $ABG(\mathbb{S}, \check{\mathbb{D}})$ either (i) have a $\check{\mathbb{D}}$-edge connecting a pair of paralogous vertices, or (ii) have $\check{\mathbb{D}}$-telomeres in one pair of paralogous vertices, or (iii) have $\check{\mathbb{D}}$-edges directly connected to $\mathbb{S}$-telomeres inciding in one pair of paralogous vertices, as illustrated in Fig. 5. Note that, for any σ_k disambiguation, the two ways of resolving each of these squares would lead to solutions with the same score, therefore each of them can be fixed arbitrarily. From now on we assume that $ABG(\mathbb{S}, \check{\mathbb{D}})$ has no symmetric squares.

A linear time greedy algorithm for the σ_4 disambiguation

Differently from 2-cycles, two valid 4-cycles can intersect with each other. But, since our graph is free of symmetric squares, two valid 2-paths cannot intersect with each other. Moreover, since a 2-path has no $\check{\mathbb{D}}$-edge connecting squares, a 4-cycle and a 2-path cannot intersect with each other. In this setting, it is clear that, for the σ_4 disambiguation, any valid 2-path is always optimal.

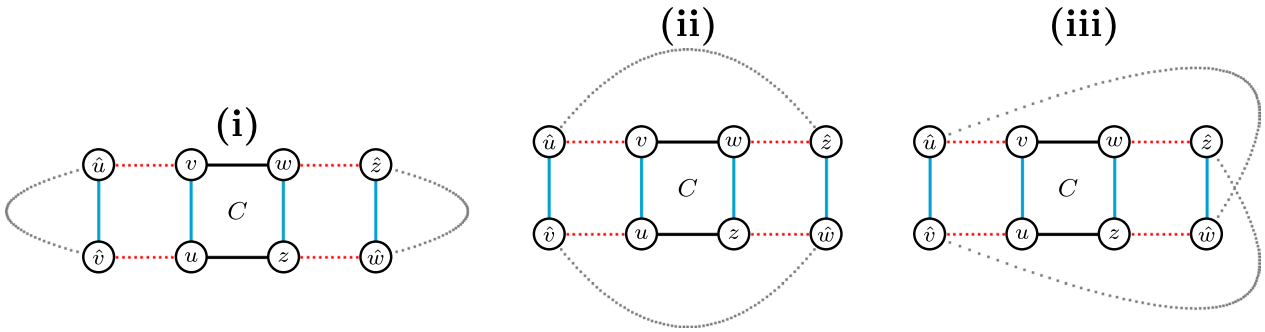

Fig. 6 Illustration of the co-optimality of every valid 4-cycle not intersecting a 2-cycle in the σ_4 disambiguation. In each of these pictures, each gray path is necessarily odd with length at least one and alternates $\check{\mathbb{D}}$- and $\mathbb{S}$-edges. Furthermore, the 4-cycle $C = (uvwz)$ is displayed in the center, induced by blue edges. In (**i**) it is easy to see that any optimal solution is induced by the blue edges and includes, besides the cycle C, cycles $(\hat{u}\ldots\hat{v})$ and $(\hat{w}\ldots\hat{z})$. In **ii** an optimal solution includes 4-cycle C and cycle $C' = (\hat{u}\hat{v}\ldots\hat{w}\hat{z}\ldots)$. If the connection between $\hat{v}$ and $\hat{w}$ is a single edge, then another optimal solution is induced by the red edges, including 4-cycle $D = (u\hat{v}\hat{w}z)$ and cycle $D' = (v\hat{u}\ldots\hat{z}w)$. And if additionally the connection between $\hat{u}$ and $\hat{z}$ is a single edge, then both C' and D' are also 4-cycles. In (**iii**) any optimal solution is induced by the blue edges and includes 4-cycle C and cycle $(\hat{u}\hat{v}\ldots\hat{z}\hat{w}\ldots)$, which is also a 4-cycle when the connections between $\hat{v}$ and $\hat{z}$ and between $\hat{u}$ and $\hat{w}$ are single edges

Furthermore, a 4-cycle that does intersect with another one is always optimal and two intersecting 4-cycles are always part of two co-optimal solutions:

Lemma 2 *Any valid 4-cycle that is disjoint from a 2-cycle in* $ABG(\mathbb{S}, \mathbb{D})$ *is induced by an optimal solution of* σ_4 *disambiguation.*

Proof All possible patterns are represented in Fig. 6: A valid 4-cycle C (in the center) connecting two squares and the three distinct possibilities of linking the four open ends. In all cases the valid 4-cycle C is either optimal or co-optimal. □

An optimal solution of σ_4 disambiguation can then be obtained greedily: after fixing squares containing edges that are part of 2-cycles, traverse the remainder of the graph and, for each valid 2-path or 4-cycle C that is found, fix the square(s) containing $\mathbb{S}$-edges that are part of C, so that C is induced. When this part is accomplished the remaining squares can be fixed arbitrarily.

Pruning $ABG(\mathbb{S}, \check{\mathbb{D}})$ for the σ_6 disambiguation

A player in the σ_6 disambiguation can be either a {2,4}-*path*, that is a valid 2- or 4-path, or a {4,6}-*cycle*, that is a valid 4- or 6-cycle. It is easy to see that players can intersect with each other. Moreover, for the σ_6 disambiguation, not every player is induced by at least one optimal solution. For that reason, a greedy algorithm does not work here and a more elaborated procedure is required. The first step is a linear time preprocessing in which from $ABG(\mathbb{S}, \check{\mathbb{D}})$ first all edges are removed that are incompatible with the existing 2-cycles, and then all remaining edges that cannot be part of a player. This results in a {6}*-pruned* ambiguous breakpoint graph $PG(\mathbb{S}, \check{\mathbb{D}})$.

The first step is easily achieved by a simple graph traversal in which for each $\check{\mathbb{D}}$-edge uv it is tested whether both ends connect to the same $\mathbb{S}$-edge uv. If this is the case, the two incident $\mathbb{S}$-edges $u\hat{v}$ and $v\hat{u}$ are removed from the graph, separating the 2-cycle (uv). Then, in the

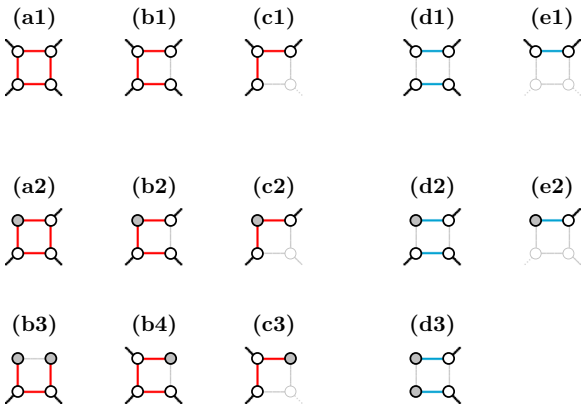

Fig. 7 Possible (partial) squares of $PG(\mathbb{S}, \check{\mathbb{D}})$. Shadowed parts represent the pruned elements (since they do not count for the score, it is not relevant to differentiate whether the pruned vertices are telomeres or not). The top line represents squares whose preserved elements include no telomere. The middle and the bottom line represent squares whose preserved elements include telomeres, marked in gray. Note that all of these are $\check{\mathbb{D}}$-telomeres ($\mathbb{S}$-telomeres are not part of any square). Cases (a1–a2), (b1–b4) and (c1–c3) are ambiguous, while cases (d1–d3) and (e1–e2) are resolved

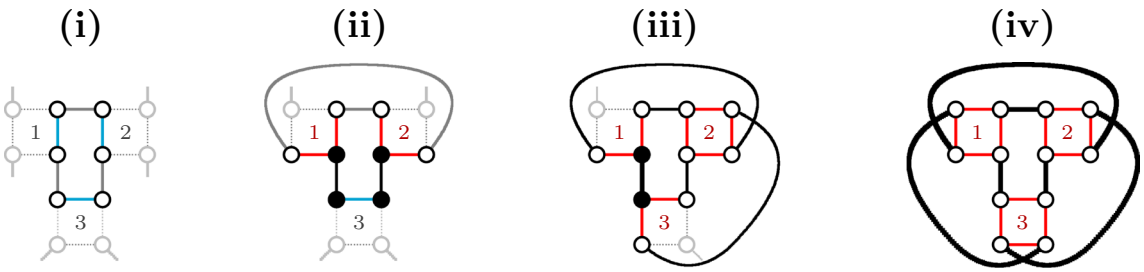

Fig. 8 (**i**) Resolved component (score = 1): a 6-cycle alternating (black) $\check{\mathbb{D}}$- and (blue) $\mathbb{S}$-edges, without intersections. (**ii**) Two 6-cycles share one $\check{\mathbb{D}}\mathbb{S}\check{\mathbb{D}}$-path composed of the two black $\check{\mathbb{D}}$-edges with the blue $\mathbb{S}$-edge in between. (**iii**) Unsaturated triplet with score = 1: every $\check{\mathbb{D}}\mathbb{S}\check{\mathbb{D}}$-path including the same $\check{\mathbb{D}}$-edge (the thick black one) occurs in two distinct 6-cycles. The thick black $\check{\mathbb{D}}$-edge occurs in four 6-cycles, all other black edges occur in two 6-cycles. (**iv**) Saturated triplet with score = 2: every $\check{\mathbb{D}}\mathbb{S}\check{\mathbb{D}}$-path occurs in two distinct 6-cycles, every black edge occurs in four 6-cycles

second step, for any remaining edge e, its 6-neighborhood (which has constant size in a graph of degree at most three) is exhaustively explored for the existence of a player involving e. If no such player is found, e is deleted. Each of these two steps clearly takes linear time $O(|ABG(\mathbb{S},\check{\mathbb{D}})|)$, and what remains is exactly the desired graph $PG(\mathbb{S},\check{\mathbb{D}})$.

The edges that are not pruned and are therefore present in $PG(\mathbb{S},\check{\mathbb{D}})$ are said to be *preserved*. As shown in Fig. 7, for any given square the pruned graph might preserve either (a1–a2) all edges, or (b1–b4) only three edges, or (c1–c3) only two edges each one from a distinct pair of paralogous edges, or (d1–d3) only two edges from the same pair of paralogous edges, or (e1–e2) a single edge. While the squares are still ambiguous in cases (a1–a2), (b1–b4) and (c1–c3), in cases (d1–d3) and (e1–e2) they are already resolved and can be fixed according to the preserved paralogous edges in cases (d1–d3) and (e1–e2). Additionally, if none of its edges is part of a player, a square is completely pruned out and is arbitrarily fixed in $ABG(\mathbb{S},\check{\mathbb{D}})$.

The smaller pruned graph $PG(\mathbb{S},\check{\mathbb{D}})$ has all relevant parts required for finding an optimal solution of σ_6 disambiguation, therefore the 6-scores of both graphs are the same: $\sigma_6(ABG(\mathbb{S},\check{\mathbb{D}})) = \sigma_6(PG(\mathbb{S},\check{\mathbb{D}}))$. A clear advantage here is that the pruned graph might be split into smaller connected components, and it is obvious that the disambiguation problem can be solved independently for each one of them. Any square that is still ambiguous in $PG(\mathbb{S},\check{\mathbb{D}})$ is called a {6}-*square*. Each connected component G of $PG(\mathbb{S},\check{\mathbb{D}})$ is of one of the two types:

1 Ambiguous: G includes at least one {6}-square;
2 Resolved (trivial): G is either a simple valid 0-, 2- or 4-path or a simple valid 2-, 4- or 6-cycle.

Let $\mathcal{C}$ and $\mathcal{P}$ be the sets of resolved components, so that $\mathcal{C}$ has all resolved cycles and $\mathcal{P}$ has all resolved paths. Furthermore, let $\mathcal{M}$ be the set of ambiguous components of $PG(\mathbb{S},\check{\mathbb{D}})$. If we denote by $\sigma_6(M)$ the 6-score of an ambiguous component $M \in \mathcal{M}$, the 6-score of $PG(\mathbb{S},\check{\mathbb{D}})$ can be computed with the formula:

$$\sigma_6(PG(\mathbb{S},\check{\mathbb{D}})) = |\mathcal{C}| + \frac{|\mathcal{P}|}{2} + \sum_{M\in\mathcal{M}} \sigma_6(M).$$

Solving the σ_6 disambiguation corresponds then to finding, for each ambiguous component $M \in \mathcal{M}$, an optimal solution including only the {6}-squares of M. From now on, by $\mathbb{S}$-edge, $\mathbb{S}$-telomere, $\check{\mathbb{D}}$-edge and $\check{\mathbb{D}}$-telomere, we are referring only to the elements that are preserved in $PG(\mathbb{S},\check{\mathbb{D}})$.

Intersection between players of the σ_6 disambiguation

Let a $\check{\mathbb{D}}\mathbb{S}\check{\mathbb{D}}$-*path* be a subpath of three edges, starting and ending with a $\check{\mathbb{D}}$-edge. This is the largest segment that can be shared by two players: although there is no room to allow distinct $\{2,4\}$-paths and/or valid 4-cycles to share a $\check{\mathbb{D}}\mathbb{S}\check{\mathbb{D}}$-path in a graph free of symmetric squares, a $\check{\mathbb{D}}\mathbb{S}\check{\mathbb{D}}$-path can be shared by at most two valid 6-cycles. Furthermore, if distinct $\check{\mathbb{D}}\mathbb{S}\check{\mathbb{D}}$-paths intersect at the same $\check{\mathbb{D}}$-edge e and each of them occurs in two distinct 6-cycles, then the $\check{\mathbb{D}}$-edge e occurs in four distinct valid 6-cycles.

In Fig. 8 we characterize this exceptional situation, which consists of the occurrence of a *triplet*, defined to be an ambiguous component composed of exactly three connected ambiguous squares in which at most two vertices, necessarily in distinct squares, are pruned out. In a *saturated* triplet, the squares in each pair are connected to each other by two $\check{\mathbb{D}}$-edges connecting paralogous vertices in both squares; if a single $\check{\mathbb{D}}$-edge is missing, that is, the corresponding vertices have outer connections, we have an *unsaturated* triplet. This structure and its score can be easily identified, therefore we will assume that our

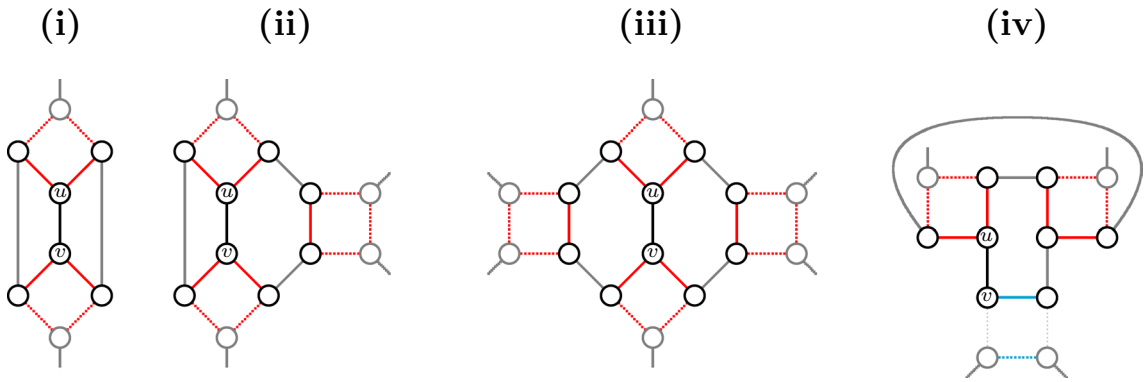

Fig. 9 Patterns free of triplets and symmetric squares showing a $\check{\mathbb{D}}$-edge uv in two distinct intersecting {4,6}-cycles which themselves do not intersect 2-cycles. (**i**–**iii**) The edge uv connects two distinct squares and is part of two {4,6}-cycles whose intersection is only uv. (**iv**) The edge uv is part of two 6-cycles whose intersection is a $\check{\mathbb{D}}\mathbb{S}\check{\mathbb{D}}$-path starting in uv. Here one square (marked in blue) is clearly fixed: if this square could be switched, this would merge each of the two existing 6-cycles into a longer cycle

graph is free from triplets. With this condition, $\check{\mathbb{D}}$-edges can be shared by at most two players:

Proposition 1 *Any $\check{\mathbb{D}}$-edge is part of either one or two (intersecting) players in a graph free of symmetric squares and triplets.*

Proof Recall that a $\check{\mathbb{D}}\mathbb{S}\check{\mathbb{D}}$*-path* is a subpath of three edges, starting and ending with a $\check{\mathbb{D}}$-edge. It is easy to see that, without symmetric squares, there is no "room" to allow distinct 4-paths and/or 4-cycles to share a $\check{\mathbb{D}}\mathbb{S}\check{\mathbb{D}}$ -path. In contrast, at most two valid 6-cycles can share a $\check{\mathbb{D}}\mathbb{S}\check{\mathbb{D}}$-path as illustrated in Fig. 8. And if the $\mathbb{S}$-edge in the middle of the shared $\check{\mathbb{D}}\mathbb{S}\check{\mathbb{D}}$-path is in an ambiguous square, we have the exceptional case of a triplet, where a $\check{\mathbb{D}}$-edge occurs in more than two players. This case can be treated separately in a preprocessing step, so that we can assume that our graph is free of triplets.

Let an $\mathbb{S}\check{\mathbb{D}}\mathbb{S}$*-path* be a subpath of three edges, starting and ending with an $\mathbb{S}$-edge. Obviously there is no "room" to allow two players to share an $\mathbb{S}\check{\mathbb{D}}\mathbb{S}$-path: (i) there are two ways of adding a $\check{\mathbb{D}}$-edge to a $\mathbb{S}\check{\mathbb{D}}\mathbb{S}$-path for obtaining a valid 4-path but they are incompatible therefore at most one can exist; or (ii) the two ends of the $\mathbb{S}\check{\mathbb{D}}\mathbb{S}$-path must incide in the same $\check{\mathbb{D}}$-edge, giving a single way of obtaining a 4-cycle; or (iii) any valid 6-cycle including the given $\mathbb{S}\check{\mathbb{D}}\mathbb{S}$-path needs to have both extra $\check{\mathbb{D}}$-edges inciding at both ends, then there can be only one way of filling the "gap" with a last $\mathbb{S}$-edge.

Now let an *open* 2-path be an $\mathbb{S}$-edge adjacent to a $\check{\mathbb{D}}$-edge such that at most one of the two includes a telomere. Considering the case of paths, in the absence of symmetric squares there is no possibility of having two 4-paths sharing an open 2-path. And considering the case of cycles, it is obvious that two $\{4, 6\}$-cycles sharing the same open 2-path must share the same $\check{\mathbb{D}}\mathbb{S}\check{\mathbb{D}}$-path, which falls in the same particular case of a triplet mentioned before.

Finally, it is easy to see that a $\check{\mathbb{D}}$-edge can occur in more than one player (general cases for cycles are illustrated in Fig. 9). However, it can only occur in more than two players if it is part of distinct $\check{\mathbb{D}}\mathbb{S}\check{\mathbb{D}}$-paths such that each of them occurs in distinct players. By construction we can see that this can only happen in a triplet (Fig. 8) or if the graph has symmetric squares. It follows that, without symmetric squares and triplets, each $\check{\mathbb{D}}$-edge occurs in at most two distinct players. □

Proposition 2 *Any $\mathbb{S}$-edge of a $\{6\}$-square is part of exactly one player in a graph free of symmetric squares and triplets.*

Proof If an $\mathbb{S}$-edge e is in a $\{6\}$-square $\mathcal{Q}$, it "shares" either the same $\check{\mathbb{D}}$-edge or the same $\check{\mathbb{D}}$-telomere d with another $\mathbb{S}$-edge e' from the same square $\mathcal{Q}$. In this case the $\check{\mathbb{D}}$-edge/telomere d is part of exactly two players and each of the $\mathbb{S}$-edges e and e' must be part of exactly one player. □

In the next sections we present the most relevant contribution of this work: an algorithm to solve the σ_6 disambiguation in linear time.

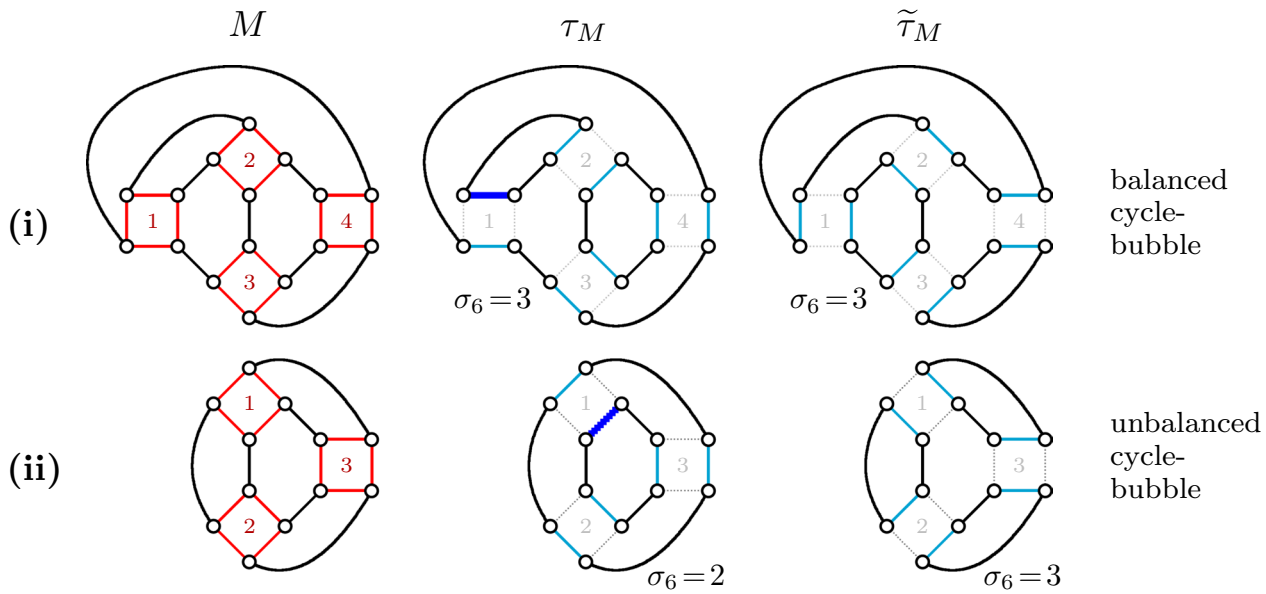

Fig. 10 Example of execution of Algorithm 1 in cycle-bubbles. In both cases the algorithm starts on the dark blue edge of square 1. In (**i**) we have a balanced cycle-bubble, for which the resulting straight disambiguation and its complementary alternative have the same score (co-optimality). In (**ii**) we have an unbalanced cycle-bubble, for which the resulting straight disambiguation and its complementary alternative have distinct scores

Solving the σ_6 disambiguation for circular genomes

For the case of *circular genomes*, which are those exclusively including circular chromosomes, the ambiguous breakpoint graph has no telomeres, therefore all players are cycles. In this case, we call each ambiguous component a *cycle-bubble*.

Two {6}-squares $\mathcal{Q}$ and $\mathcal{Q}'$ are *neighbors* when a vertex of $\mathcal{Q}$ is connected to a vertex of $\mathcal{Q}'$ by a $\breve{\mathbb{D}}$-edge. Any $\mathbb{S}$-edge e of a {6}-square $\mathcal{Q}$ in a cycle-bubble M is part of exactly one {4,6}-cycle (Proposition 2) and both $\breve{\mathbb{D}}$-edges inciding at the endpoints of e would clearly induce the same {4,6}-cycle. For that reason, the choice of e (and its paralogous edge $\hat{e}$) implies a unique way of resolving all neighbors of $\mathcal{Q}$, and, by propagating this to the neighbors of the neighbors and so on, all squares of M are resolved, resulting in what we call *straight solution* τ_M (see Algorithms 1 and 2). Then we can immediately obtain the *complementary* alternative solution $\widetilde{\tau}_M$, by switching all ambiguous squares of τ_M. A cycle-bubble is said to be *unbalanced* if $\sigma_6(\tau_M) \neq \sigma_6(\widetilde{\tau}_M)$ or *balanced* if $\sigma_6(\tau_M) = \sigma_6(\widetilde{\tau}_M)$. If M is unbalanced, its score is given either by τ_M or by $\widetilde{\tau}_M$ (the maximum among the two). If M is balanced, its score is given by both τ_M and $\widetilde{\tau}_M$ (co-optimality). Examples are given in Fig. 10.

Algorithm 1 StraightBubbleSolution

Input: A cycle-bubble M whose {6}-squares are numbered $\mathcal{Q}_1, \mathcal{Q}_2, \ldots, \mathcal{Q}_m$
Output: A solution τ_M of M
1: $e \leftarrow$ any $\mathbb{S}$-edge in $\mathcal{Q}_1$;
2: $\tau_M[1] \leftarrow \{e, \hat{e}\}$;
3: **for** $i \leftarrow 2, \ldots, m$ **do** $\tau_M[i] \leftarrow \emptyset$;
4: ResolveNeighbors(τ_M, e); /* recursive procedure */
5: **if** $\hat{e}$ is an $\mathbb{S}$-edge in M **then** /* the paralogous $\mathbb{S}$-edge $\hat{e}$ is also in M */
6: ResolveNeighbors$(\tau_M, \hat{e})$; /* recursive procedure */
7: **return** τ_M

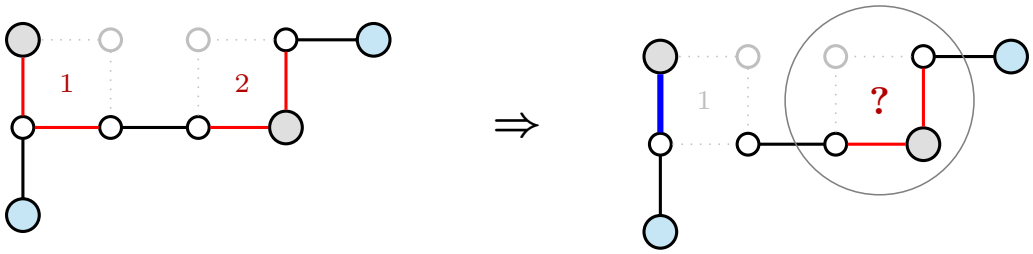

Fig. 11 Example showing that the straight algorithm does not work with paths: if we start on the dark blue edge of square number 1, we cannot propagate the effect of this choice to the neighbor square

Algorithm 2 ResolveNeighbors

Input: A partially filled solution τ_M and an $\mathbb{S}$-edge uv of cycle-bubble M
/* $\mathbb{S}$-edge uv is adjacent to two $\check{\mathbb{D}}$-edges uz and vw */
if vertex z is not in a resolved or fixed square **then**
  $i \leftarrow$ index in τ_M of square containing z;
  $e \leftarrow \mathbb{S}$-edge zx of $\mathcal{Q}_i$ forming a $\{4,6\}$-cycle with uv and uz;
  $\tau_M[i] \leftarrow \{e, \hat{e}\}$;
  if $\hat{e}$ is an $\mathbb{S}$-edge in M **then**
    ResolveNeighbors$(\tau_M, \hat{e})$;
if vertex w is not in a resolved or fixed square **then**
  $j \leftarrow$ index in τ_M of square containing w;
  $f \leftarrow \mathbb{S}$-edge wy of $\mathcal{Q}_j$ forming a $\{4,6\}$-cycle with uv and vw;
  $\tau_M[j] \leftarrow \{f, \hat{f}\}$;
  if $\hat{f}$ is an $\mathbb{S}$-edge in M **then**
    ResolveNeighbors$(\tau_M, \hat{f})$;
return

Solving the σ_6 disambiguation with linear chromosomes

For genomes with linear chromosomes, the ambiguous components might include paths besides cycle-bubbles. In the presence of paths, the straight algorithm unfortunately does not work (see Fig. 11). We must then proceed with an additional characterization of each ambiguous component M of $PG(\mathbb{S}, \check{\mathbb{D}})$, splitting the disambiguation of M into smaller subproblems.

As we will present in the following, the solution for arbitrarily large components can be split into two types of problems, which are analogous to solving the *maximum independent set* of auxiliary subgraphs that are either simple paths or double paths. In both cases, the solutions can be obtained in linear time.

Intersection graph of an ambiguous component

The auxiliary *intersection graph* $\mathcal{I}(M)$ of an ambiguous component M has a vertex with weight $\frac{1}{2}$ for each {2,4} -path and a vertex with weight 1 for each {4,6}-cycle of M. Furthermore, if two distinct players intersect, we have an edge between the respective vertices. The intersection graphs of all ambiguous components can be built during the pruning procedure without increasing its linear time complexity.

Note that an independent set of maximum weight in $\mathcal{I}(M)$ corresponds to an optimal solution of M. Although in general this problem is NP-hard, in our case the underlying ambiguous component M imposes a regular structure to its intersection graph, allowing us to find such an independent set in linear time.

If two {2,4}-paths intersect in their $\mathbb{S}$-telomere, this intersection must include the incident $\check{\mathbb{D}}$-edge. Therefore, when we say that an intersection occurs at an $\mathbb{S}$ -telomere, this automatically means that the intersection is the $\check{\mathbb{D}}$-edge inciding in an $\mathbb{S}$-telomere. A valid 4-cycle has two $\check{\mathbb{D}}$-edges and a valid 6-cycle has three $\check{\mathbb{D}}$ -edges. Besides the one at the $\mathbb{S}$-telomere, a valid 4-path has one $\check{\mathbb{D}}$-edge while a valid 2-path has none - therefore the latter cannot intersect with a {4,6}-cycle. When we say that 4-paths and/or {4,6}-cycles intersect with each other in a $\check{\mathbb{D}}$-edge, we refer to an *inner* $\check{\mathbb{D}}$-edge and not one inciding in an $\mathbb{S}$-telomere.

Since the contribution of each cycle in the score is twice as much as the contribution of a path, we make a distinction between two types of subgraphs of an

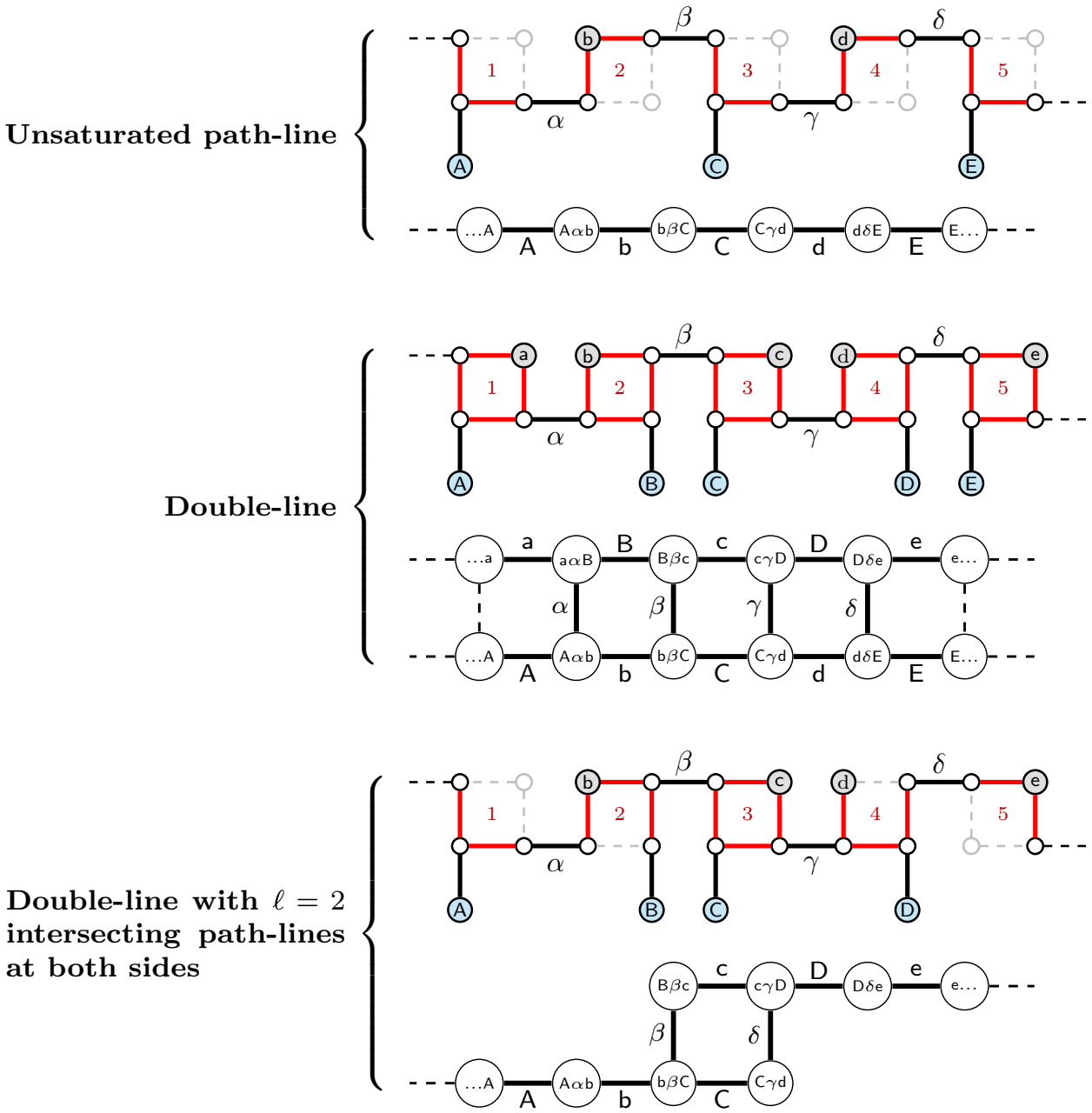

Fig. 12 Examples of an unsaturated path-line, a double-line and the intersection between a double-line and two unsaturated path-lines

intersection graph $\mathcal{I}(M)$, which can correspond to cycle-bubbles or path-flows.

Path-flows in the intersection graph

A *path-flow* in $\mathcal{I}(M)$ is a maximal connected subgraph whose vertices correspond to {2,4}-paths. A *path-line* of length ℓ in a path-flow is a series of ℓ paths, such that each pair of consecutive paths intersect at a telomere. Assume that the vertices in a path-line are numbered from left to right with integers $1, 2, \ldots, \ell$. A *double-line* consists of two parallel path-lines of the same length ℓ, such that vertices with the same number in both lines intersect in a $\check{\mathbb{D}}$-edge and are therefore connected by an edge. A 2-path has no free $\check{\mathbb{D}}$-edge, therefore a double-line is exclusively

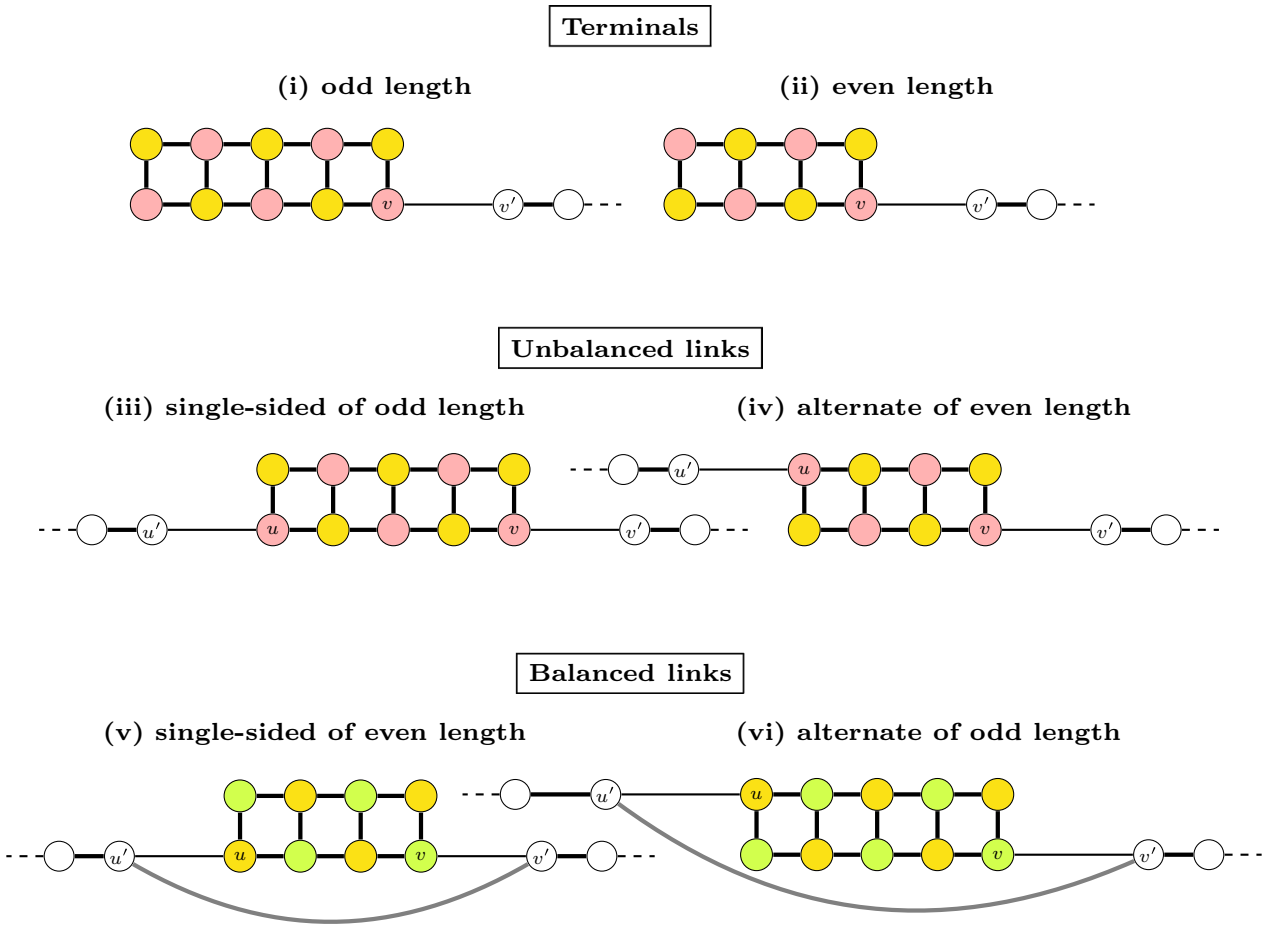

Fig. 13 Types of double-line: terminal, balanced and unbalanced links. The yellow solution that in cases (**i**–**ii**) leaves v unselected and in cases (**iii**–**iv**) leaves u and v unselected can be fixed so that an independent set of the adjacent unsaturated path-line(s) can start at v' (and u'). In cases (**v**–**vi**) either the yellow or the green solution will be fixed later; it will be the one compatible with the selected independent set of the unsaturated path-line ending in u' concatenated to the one starting in v'

composed of 4-paths. If a path-line is part of a double-line, it is *saturated*, otherwise it is *unsaturated*. Since each 4-path of a double-line has a $\check{\mathbb{D}}$-edge intersection with another and each 4-path can have only one $\check{\mathbb{D}}$-edge intersection, no vertex of a double-line can be connected to a cycle in $\mathcal{I}(M)$. Examples of an unsaturated path-line and a double-line are given in Fig. 12.

Let us assume that a double-line is always represented with one upper path-line and one lower path-line. A double-line of length ℓ has 2ℓ vertices and exactly two independent sets of maximum weight, each one with ℓ vertices and weight $\frac{\ell}{2}$: one includes the paths with odd numbers in the upper line and the paths with even numbers in the lower line, while the other includes the paths with even numbers in the upper line and the paths with odd numbers in the lower line. Since a double-line cannot intersect with cycles, it is clear that at least one of these independent sets will be part of a global optimal solution for $\mathcal{I}(M)$. In other words, not only the two possible local optimal solutions and their (common) weight are known, but it is guaranteed that at least one of them will be part of a global optimal solution. A maximal double-line can be of three different types:

1 Isolated: corresponds to the complete graph $\mathcal{I}(M)$. Here the double line can be cyclic. If ℓ is even, in both upper and lower lines of a cyclic double-line, the last vertex intersects at a telomere with the first vertex. If ℓ is odd, this connection of a cyclic double-line is "twisted": the last vertex of the upper line intersects at a telomere with the first vertex of the lower line, and the first vertex of the upper line intersects at a telomere with the last vertex of the lower line. Being cyclic or not, any of the two optimal local solutions can be fixed.

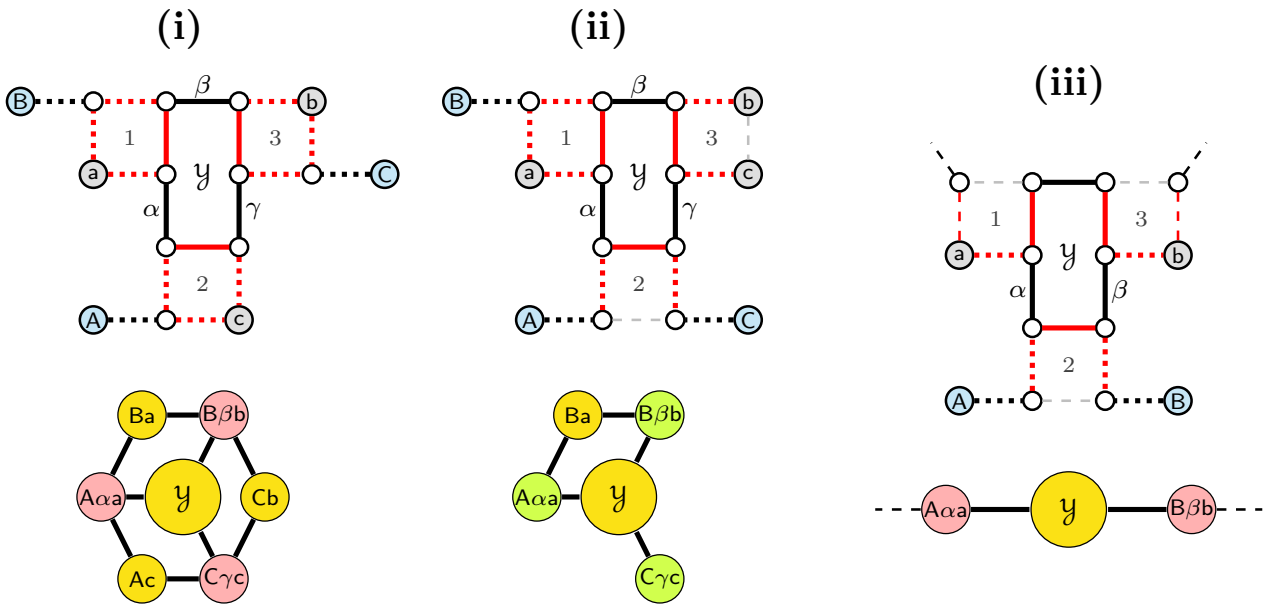

Fig. 14 Underlying pruned subgraphs and corresponding intersection graphs of a bubble with a single 6-cycle $\mathcal{Y}$ (solid edges). Dotted edges are exclusive to paths and dashed gray edges are pruned out. In **i** and **ii**, $\mathcal{Y}$ intersects with three valid 4-paths Aαa, Bβb and Cγc. In **i**, the yellow solution including $\mathcal{Y}$ would also include the three 2-paths Ab, Bc and Ca, being clearly superior. In **ii**, the yellow solution including $\mathcal{Y}$ would still include the 2-path Ba, having the same score of the green solution with three 4-paths. In any of the two cases, the underlying graph cannot be extended. In **iii**, $\mathcal{Y}$ has plug connections with unsaturated path-lines starting at 4-paths Aαa and Bβb (both can be extended)

2 Terminal: intersects with one unsaturated path-line, and, without loss of generality, the intersection involves the vertex v located at the rightmost end of the lower line. Here at least one of the two optimal local solutions would leave v unselected; we can safely fix this option. (See Fig. 13**(i)** and **(ii)**.)
3 Link: intersects with unsaturated lines at both ends. The intersections can be:

   (a) Single-sided: both occur at the ends of the same saturated line, or
   (b) Alternate: the left intersection occurs at the end of one saturated line and the right intersection occurs at the end of the other.

Let v' be the outer vertex connected to a vertex v belonging to the link at the right and u' be the outer vertex connected to a vertex u belonging to the link at the left. Let a *balanced link* be alternate of odd length, or single-sided of even length. In contrast, an *unbalanced link* is alternate of even length, or single-sided of odd length. If the link is unbalanced, one of the two local optimal solutions leaves both u and v unselected; we can safely fix this option. (See Fig. 13iii and iv.) If the link is balanced, we cannot fix the solution before-hand, but we can reduce the problem, by removing the connections uu' and vv' and adding the connection $u'v'$. Since both u' and v' must be the ends of unsaturated lines, this procedure simply concatenates these two lines into a single unsaturated path-line. (See Fig. 13v and vi.) Finding a maximum independent set of the remaining unsaturated path-lines is a trivial problem that will be solved last; depending on whether one of the vertices u' and v' is selected in the end, we can fix the solution of the original balanced link.

Intersection between path-flows and cycle-bubbles

If an ambiguous component has only cycles, its solution can be easily obtained with the straight algorithm presented in the previous section. More intricate is when an ambiguous component M includes cycles and paths. In this case we redefine a *cycle-bubble* as corresponding to a maximal connected subgraph of $\mathcal{I}(M)$ whose vertices correspond to {4,6}-cycles. Let H be the subgraph of M including all edges that compose the cycles of a cycle-bubble. An optimal solution for H is either the straight solution τ_H, given by Algorithm 1, or its alternative $\widetilde{\tau}_H$. Recall that if both τ_H and $\widetilde{\tau}_H$ have the same

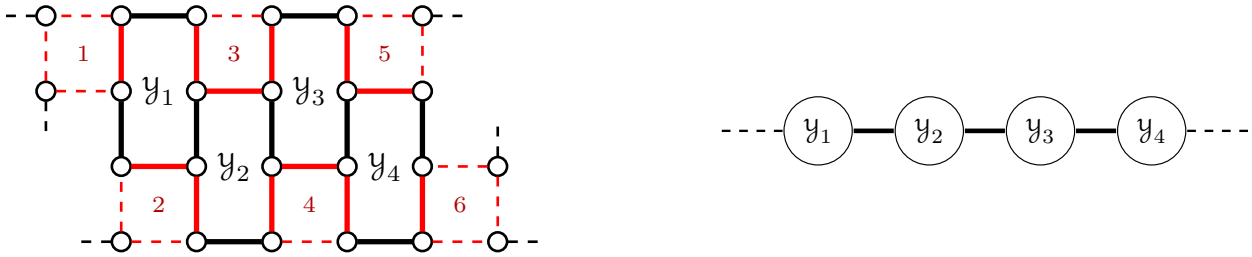

Fig. 15 Cycle-bubble of type cycle-line and its intersection graph

score, then H is said to be *balanced*, otherwise it is said to be *unbalanced*.

Proposition 3 *Let an ambiguous component M have cycle-bubbles H_1,..., H_q. There is an optimal solution for M including, for each $i = 1, ..., q$: (1) the optimal solution for H_i, if H_i is unbalanced; or (2) either τ_{H_i} or $\widetilde{\tau}_{H_i}$, if H_i is balanced.*

Proof We will analyze the cases by increasing the size of the maximal subgraph containing intersecting cycles:

1 *A* $\{4,6\}$*-cycle C that does not intersect with any other* $\{4,6\}$*-cycle*: (a) if C is a 4-cycle, it can intersect with at most two valid 4-paths; therefore there is an optimal solution including C; (b) if C is a 6-cycle, it can intersect with at most three valid 4-paths, but if it intersects with three valid 4-paths there will be at least one valid 2-path P compatible with C; therefore there is an optimal solution including C and P (see Fig. 14ii).
2 *Two* $\{4,6\}$*-cycles C and C' intersecting with each other but not with any other* $\{4,6\}$*-cycle*: Since valid 4-cycles have less edges for intersection, let us assume without loss of generality that both C and C' are 6-cycles. Their intersection (illustrated in Additional file 1: Figs. A7 and A8) can be:

   (a) a $\check{\mathbb{D}}\mathbb{S}\check{\mathbb{D}}$-path, and in this case each cycle can intersect with at most one valid 4-path, therefore there is an optimal solution including either C or C';
   (b) a single $\check{\mathbb{D}}$-edge, and in this case each cycle can intersect with two valid 4-paths, therefore there is an optimal solution including either C or C'.

As the size of the bubble grows, there is less space for intersecting paths, and each cycle intersects with at most one path. In general, the best we can get by replacing cycles by paths are co-optimal solutions. □

As a consequence of Proposition 3, if a cycle-bubble is unbalanced, its optimal solution can be fixed so that the unsaturated path-lines around it can be treated separately. Similarly, if a balanced cycle-bubble H has a single intersection involving a cycle C and a path P (that can be the first vertex of an unsaturated path-line), then we can immediately fix the solution of H that does not contain C.

Balanced cycle-bubbles intersecting with at least two paths

If a cycle-bubble H is balanced and intersects with at least two paths, then it requires a special treatment. However, as we will see, here the only case that can be arbitrarily large is easy to handle. Let a cycle-bubble be a *cycle-line* when it consists of a series of valid 6-cycles, such that each pair of consecutive cycles intersect at a $\check{\mathbb{D}}$-edge (see Fig. 15).

Proposition 4 *Cycle-bubbles involving 9 or more cycles must be a cycle-line.*

Proof In Fig. 16 (whose steps are more elaborated in Additional file 1: Figs. S1–S6) we show that, if a bubble is not a line, it reaches its "capacity" with at most 8 cycles. □

Besides having its size limited to 8 cycles, the more complex a non-linear cycle-bubble becomes, the less space it has for paths around it. The solutions for these few exceptional bounded cases are described in the end of this section.

Our focus now is the remaining situation of a balanced cycle-line with intersections involving at least two cycles. Recall that cycles can only intersect with unsaturated path-lines. An intersection between a cycle- and a path-line is a *plug connection* when it occurs between vertices that are at the ends of both lines.

Proposition 5 *Cycle-lines of length at least 4 can only have plug connections.*

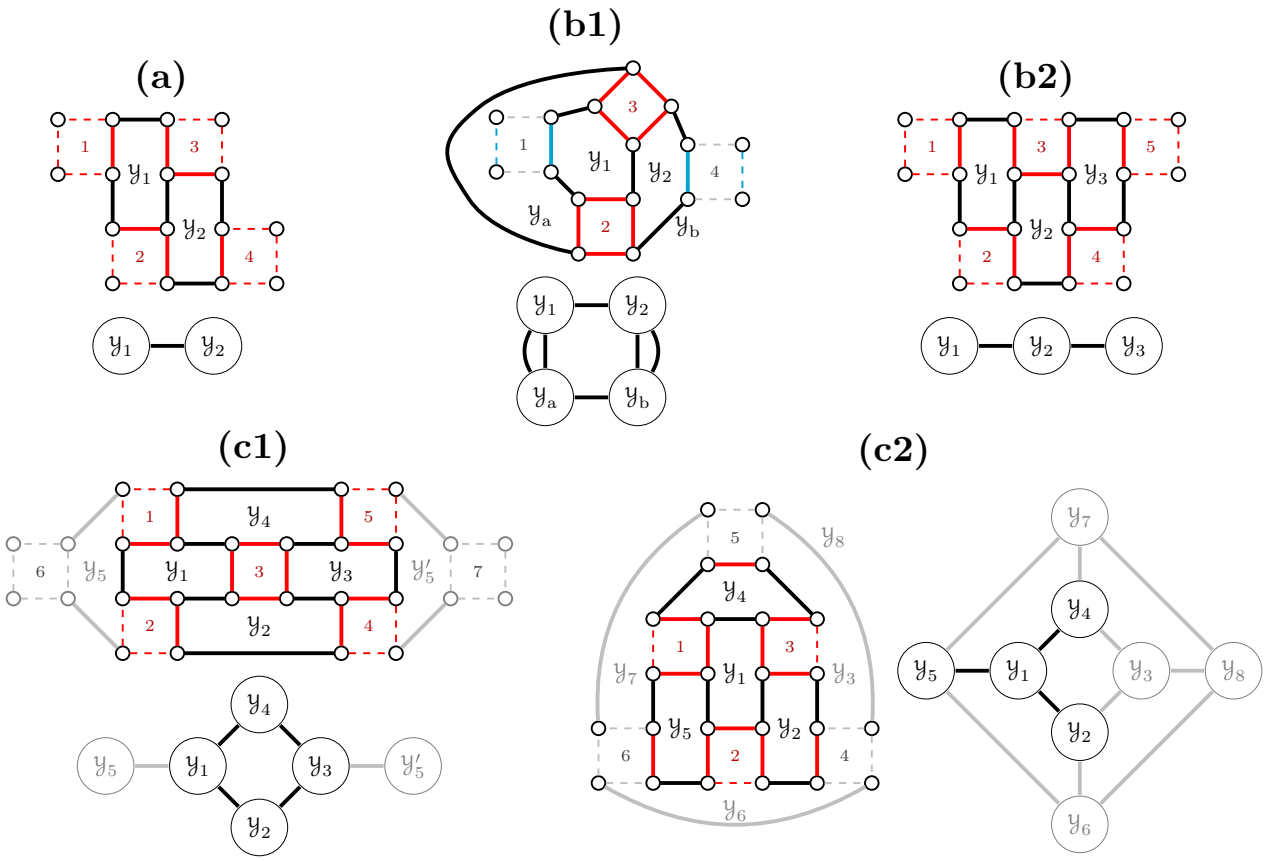

Fig. 16 While a cycle-line can be arbitrarily large, by increasing the complexity of a bubble we quickly saturate the space for adding cycles to it. Starting with (**a**) a simple cycle-line of length two, we can either (**b1**) connect the open vertices of squares 2 and 3, obtaining a cyclic cycle-line of length 4 that cannot be extended, or (**b2**) extend the line so that it achieves length three. From (**b2**) we can obtain (**c1**) a cyclic cycle-line of length 4 that can be extended first by adding cycle $\mathcal{Y}_5$ next to $\mathcal{Y}_1$ and then either adding $\mathcal{Y}'_5$ next to $\mathcal{Y}_3$ or closing $\mathcal{Y}_6$, $\mathcal{Y}_7$ and $\mathcal{Y}_8$ so that we get (**c2**). In both cases no further extensions are possible. Note that (**c2**) can also be obtained by extending a cycle-line of length three and transforming it into a *star* with three branches, that can still be extended by closing $\mathcal{Y}_3$, $\mathcal{Y}_6$, $\mathcal{Y}_7$ and $\mathcal{Y}_8$. (These steps are more elaborated in of Additional file 1: Figs. S1–S6)

Proof If a cycle-line has length at least four, its underlying graph has only "room" for intersections with 4-paths next to its leftmost or rightmost cycles. See the illustration in Fig. 17.

For arbitrarily large instances, the last missing case is that of a balanced cycle-line with plug connections at both sides, called a *balanced link*. The procedure here is the same as that for double-lines that are balanced links, where the local solution can only be fixed after fixing those of the outer connections (see Fig. 17ii).

Exceptional bounded cases.

Balanced cycle-lines with two cycles can have connections to path-lines that are not plugs, but the number of cases is again limited. In most of them (shown in Additional file 1: Fig. S7) the bubble is saturated and the paths around cannot be connected to extendable path-lines. For these bubbles all paths are over the same squares of the cycles, therefore the straight algorithm would give the two overall alternatives including the paths around each of these bubbles, and the best solution can be immediately fixed.

In another case (shown in Additional file 1: Fig. S8i) there is one extendable path-line, but the local solution (including the bubble and the paths that are over the same squares) is unbalanced, therefore also here we can fix the best among the two overall alternatives given by the straight algorithm.

In the last two cases (shown in Additional file 1: Fig. S8ii, iii) there are extendable path-lines, and the

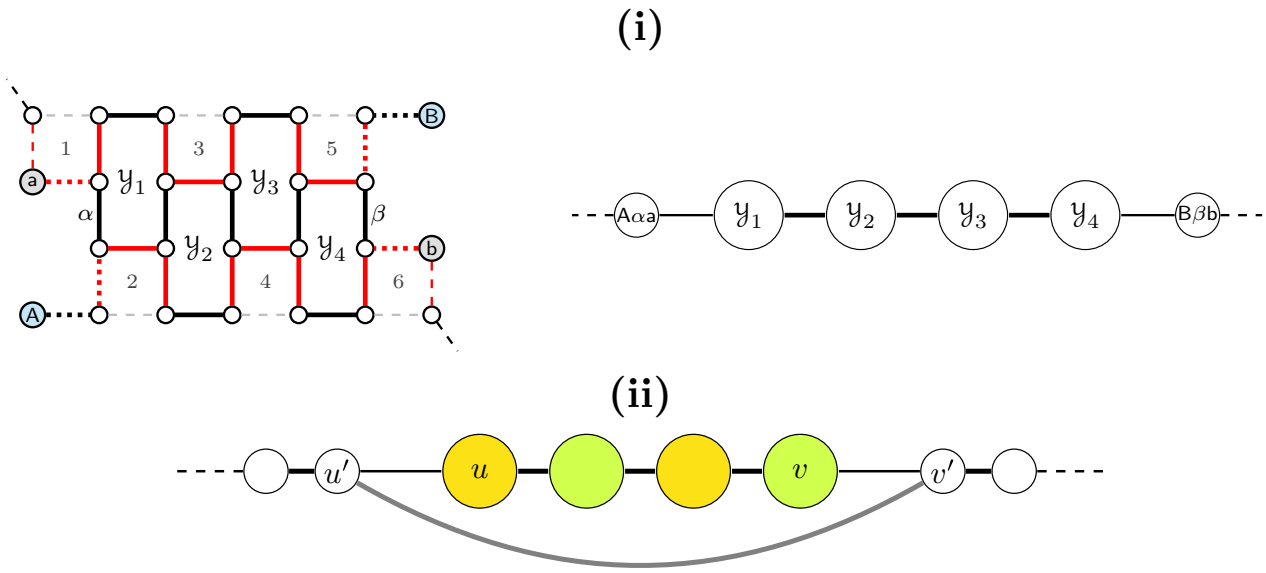

Fig. 17 (**i**) A cycle-line of length 4 or larger only allows plug connections. In contrast, cycle-lines of lengths 1–3 admit other types of connections (see Figs. 14 and A7, A8, the last two in Additional file 1). (**ii**) If the cycle-line has even length and plug connections at both sides, we have a balanced link: either the yellow or the green solution will be fixed later; it will be the one compatible with the selected independent set of the unsaturated path-line ending in u' concatenated to the one starting in v'

local solutions (including the bubble and the paths that are over the same squares) are balanced. In the first case, there is only one extendable path-line and we can fix the solution that does not include the last "visible" path of the path-line. The second case is analogous to cycle-lines of type balanced link, with the difference that here the lines are already concatenated; the local solution can then only be fixed after fixing those of the outer connections.

Concerning non-linear cycle-bubbles, there are only four distinct cases that need to be considered: one case of a non-linear bubble with two 6-cycles (Additional file 1: Fig. S7iii) and three cases of non-linear bubbles with four 6-cycles (Additional file 1: Fig. S9). In all of these four cases, the bubble is saturated and the paths around cannot be connected to extendable path-lines. Indeed, also for these bubbles all paths are over the same squares of the cycles, therefore the straight algorithm would give the two overall alternatives including the paths around each of these bubbles, and the best among these solutions can be immediately fixed.

What remains is a set of independent unsaturated path-lines

For each remaining unsaturated path-line, an optimal solution can be trivially found as follows. First assume that in an unsaturated path-line of length ℓ the paths are numbered from one end to the other with $1, 2, \ldots, \ell$. The solution that selects all paths with odd numbers must be optimal, with some particularities if the path-line is cyclic: in this case the initial vertex for the sequential numbering is arbitrarily chosen; furthermore, if ℓ is odd, the path whose number is ℓ must be excluded from the solution. Then, depending on the connections between the selected vertices of the unsaturated path-line and vertices from balanced links that are double-lines or cycle-lines, the compatible solutions for the latter ones are also fixed. See examples in Fig. 18.

Final remarks and discussion

Given a singular genome $\mathbb{S}$ and a duplicated genome $\mathbb{D}$ over the same set of gene families, the double distance of $\mathbb{S}$ and $\mathbb{D}$ aims to find the smallest distance between $\mathbb{D}$ and any element from the set $2\mathbb{S}$, that contains all possible genome configurations obtained by doubling the chromosomes of $\mathbb{S}$. Different underlying genomic distance measures give rise to different double distances: the breakpoint double distance of $\mathbb{S}$ and $\mathbb{D}$ is an easy problem that can be greedily solved in linear time, while computing the DCJ double distance of $\mathbb{S}$ and $\mathbb{D}$ is NP-hard. Our study is an exploration of the complexity space between these two extremes.

We considered a class of genomic distance measures called σ_k distances, for $k = 2, 4, 6, \ldots, \infty$, which are between the breakpoint (σ_2) and the DCJ (σ_∞) distance. In this work we presented linear time algorithms for computing the double distance under the σ_4, and under

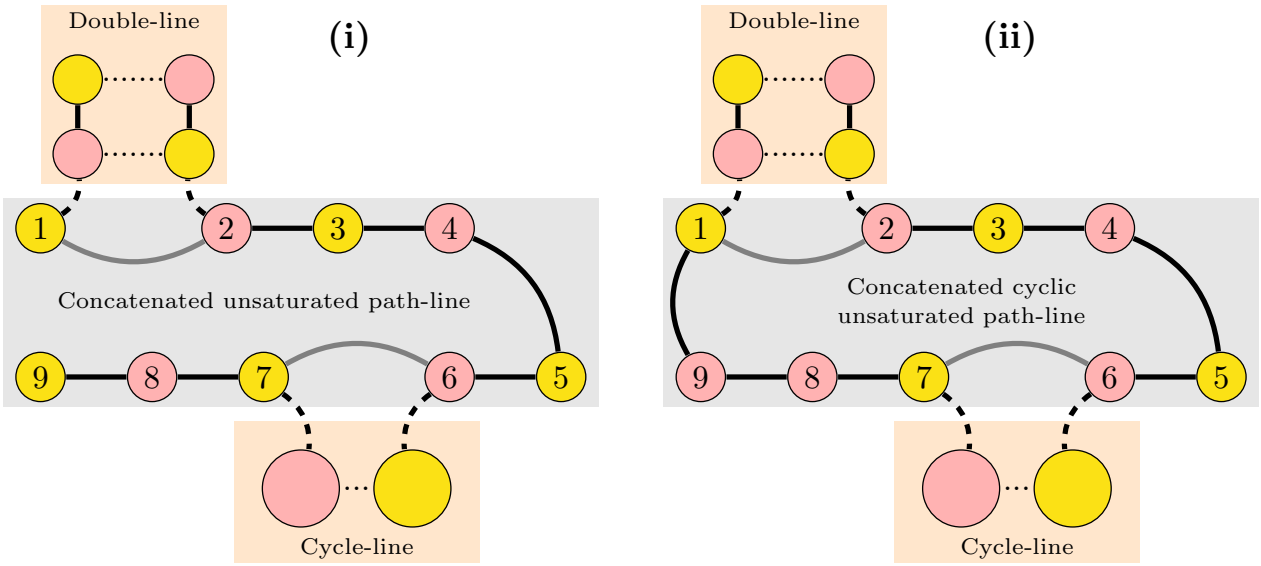

Fig. 18 (**i**) The same concatenated unsaturated path-line can intersect with several balanced links (double-lines and/or cycle-lines). The optimal (yellow) solution is found by first selecting the paths with odd numbers in the unsaturated path-line and then, for each balanced link, selecting the compatible set of vertices among the two alternating choices. (**ii**) If the unsaturated path-line is cyclic and has odd length, the last path must be excluded from the optimal solution

the σ_6 distance. Our solution relies on a variation of the breakpoint graph called ambiguous breakpoint graph.

The solutions we found so far are greedy with all players being optimal in σ_2, greedy with all players being co-optimal in σ_4 and non-greedy with non-optimal players in σ_6, all of them running in linear time. More specifically for the σ_6 case, after a pre-processing that fixes symmetric squares and triplets, at most two players share an edge. However we can already observe that, as k grows, the number of players sharing a same edge also grows. For that reason, we believe that, if for some $k \geq 8$ the complexity of the σ_k double distance is found to be NP-hard, the complexity is also NP-hard for any $k' > k$. We expect that when we find the smallest k for which the σ_k double distance is NP-hard we will be able to confirm this conjecture. In any case, the natural next step in our research is to study the σ_8 double distance.

Besides the double distance, other combinatorial problems related to genome evolution and ancestral reconstruction, including median and guided halving, have the distance problem as a basic unit. And, analogously to the double distance, these problems can be solved in polynomial time (but differently from the double distance, not greedy and linear) when they are built upon the breakpoint distance, while they are NP-hard when they are built upon the DCJ distance [2]. Therefore, a challenging avenue of research is doing the same exploration for both median and guided halving problems under the class of σ_k distances. In both cases it seems possible to adopt variations of the breakpoint graph. To the best of our knowledge, the guided halving problem has not yet been studied for any σ_k distance except $k = 2$ and $k = \infty$, while for the median much effort for the σ_4 distance has been done but no progress was obtained so far. A reason for this difference of progress between double distance and median is probably related to the underlying approaches. While the double distance can be solved by *removing* paralogous edges from the ambiguous breakpoint graph, solving the median requires *adding* new edges (representing the adjacencies of the median genome) to an extended (*multiple*) breakpoint graph, and the combinatorial space of the distinct possibilities of doing that could not yet be described.

Supplementary Information

The online version contains supplementary material available at https://doi.org/10.1186/s13015-023-00246-y.

Additional file 1. This file contains additional figures for supporting the proofs of statements in this manuscript.

Acknowledgements
We would like to thank Cedric Chauve for bringing our attention to the class of σ_k distances as a means for studying the hardness bound between the breakpoint distance and the DCJ distance in combinatorial problems related to genome evolution. Thanks also to Eloi Araujo, Daniel Doerr and Fábio H. V. Martinez for helping us studying the median problem under this class.

Author Contributions
MDVB and JS devised the study. MDVB, LRB and KK worked out the technical parts. All authors wrote and approved the manuscript.

Funding
Open Access funding enabled and organized by Projekt DEAL.

Availability of data and materials
Not applicable.

Declarations

Ethics approval and consent to participate
Not applicable.

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

Received: 31 March 2023 Accepted: 10 November 2023
Published online: 04 January 2024

References

1. Sankoff D. Edit distance for genome comparison based on non-local operations. In: Manber U, editor. Proceedings of CPM 1992, LNCS, vol. 644. Berlin: Springer; 1992. p. 121–35. https://doi.org/10.1007/3-540-56024-6_10.
2. Tannier E, Zheng C, Sankoff D. Multichromosomal median and halving problems under different genomic distances. BMC Bioinformatics. 2009;10:120. https://doi.org/10.1186/1471-2105-10-120.
3. Hannenhalli S, Pevzner PA. Transforming men into mice polynomial algorithm for genomic distance problem. In: Hannenhalli S, editor. Proceedings of FOCS 1995. Milwaukee: IEEE; 1995. p. 581–92. https://doi.org/10.1109/SFCS.1995.492588.
4. Hannenhalli S, Pevzner PA. Transforming cabbage into turnip: polynomial algorithm for sorting signed permutations by reversals. J ACM. 1999;46(1):1–27. https://doi.org/10.1145/300515.300516.
5. Yancopoulos S, Attie O, Friedberg R. Efficient sorting of genomic permutations by translocation, inversion and block interchange. Bioinformatics. 2005;21(16):3340–6. https://doi.org/10.1093/bioinformatics/bti535.
6. El-Mabrouk N, Sankoff D. The reconstruction of doubled genomes. SIAM J Comput. 2003;32(3):754–92. https://doi.org/10.1137/S009753970037177.
7. Bafna V, Pevzner PA. Genome rearrangements and sorting by reversals. In: Bafna V, editor. Proceedings of FOCS 1993. Palo Alto: IEEE; 1993. p. 148–57. https://doi.org/10.1109/SFCS.1993.366872.
8. Bergeron A, Mixtacki J, Stoye J. A unifying view of genome rearrangements. In: Bucher P, Moret BM, editors. Proceedings of WABI 2006, LNBI, vol. 4175. Zurich: Springer; 2006. p. 163–73. https://doi.org/10.1007/11851561_16.
9. Alekseyev M, Pevzner PA. Colored de Bruijn graphs and the genome halving problem. IEEE/ACM Trans Comput Biol Bioinform. 2008;4(1):98–107. https://doi.org/10.1109/TCBB.2007.1002.
10. Mixtacki J. Genome halving under DCJ revisited. In: Hu X, Wang J, editors. Proceedings of COCOON 2008, LNCS, vol. 5092. Dalian: Springer; 2008. p. 276–86. https://doi.org/10.1007/978-3-540-69733-6_28
11. Chauve C. Personal communication in Dagstuhl Seminar no. 18451—genomics, pattern avoidance, and statistical mechanics. 2018.
12. Braga MDV, Brockmann LR, Klerx K, Stoye J. A linear time algorithm for an extended version of the breakpoint double distance. Proceedings of WABI 2022, LIPIcs 242(13). Dagstuhl Publishing; 2022. https://doi.org/10.4230/LIPIcs.WABI.2022.13.
13. Braga MDV, Brockmann LR, Klerx K, Stoye J. On the class of double distance problems. In: Jahn K, Vinai T, editors. Proceedings of Recomb-CG 2023, LNBI, vol. 13883. Istanbul: Springer; 2023. p. 35–50. https://doi.org/10.1007/978-3-031-36911-7_3

Publisher's Note