A Linear Time Algorithm for an Extended Version of the Breakpoint Double Distance

Marília D. V. Braga[1]
Faculty of Technology and Center for Biotechnology (CeBiTec), Bielefeld University, Germany

Leonie R. Brockmann
Faculty of Technology and Center for Biotechnology (CeBiTec), Bielefeld University, Germany

Katharina Klerx
Faculty of Technology and Center for Biotechnology (CeBiTec), Bielefeld University, Germany

Jens Stoye
Faculty of Technology and Center for Biotechnology (CeBiTec), Bielefeld University, Germany

Abstract

Two genomes over the same set of gene families form a *canonical* pair when each of them has exactly one gene from each family. A genome is *circular* when it contains only circular chromosomes. Different distances of canonical circular 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. Then, the breakpoint distance is equal to $n - c_2$, where n is the number of genes and c_2 is the number of cycles of length 2. Similarly, when the considered rearrangements are those modeled by the *double-cut-and-join* (DCJ) operation, the rearrangement distance is $n - c$, where c is the total number of cycles.

The distance problem 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 $n - (c_2 + c_4 + \ldots + c_k)$, and increasingly investigate the complexities of median and double distance for the σ_4 distance, then the σ_6 distance, and so on. 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.

2012 ACM Subject Classification Applied computing → Bioinformatics

Keywords and phrases Comparative genomics, genome rearrangement, breakpoint distance, double-cut-and-join (DCJ) distance, double distance

Digital Object Identifier 10.4230/LIPIcs.WABI.2022.13

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.

1 Introduction

In genome comparison, the most elementary problem is that of computing a *distance* between two given *genomes* [10], 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

[1] Corresponding author

is the *breakpoint* model, whose distance consists of somehow quantifying the points of dissimilarity between the two genomes, a *point* in a genome being the oriented neighborhood between two genes in one of its chromosomes [11]. 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 [6, 7, 12].

Independently of the underlying model, the distance problem is a basic unit for several other combinatorial problems related to genome evolution and ancestral reconstruction [11]. 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 [5]. 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.

In this work, we assume that all considered genomes contain only circular chromosomes and are therefore *circular*. Our study relies on the *breakpoint graph*, a structure that represents the relation between two given genomes [2]. 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. If we call *k-cycle* a cycle of length k and assume that both genomes have n genes, their breakpoint distance is equal to $n - c_2$, where c_2 is the number of 2-cycles [11]. Similarly, when the considered rearrangements are those modeled by the *double-cut-and-join* (DCJ) operation [12], the rearrangement distance is $n - c$, where c is the total number of cycles [3].

While the halving problem under both breakpoint and rearrangement distances can be solved in polynomial time [1, 5, 9, 11], 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 [11]. One way of exploring the complexity space between these two extremes is to consider a σ_k distance [4], defined to be $n - (c_2 + c_4 + \ldots + c_k)$, 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. [4]) but no progress was obtained so far.

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* [11] 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 are fulfilled, meaning that the common adjacencies between the compared genomes are always preserved. 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.

Recall that the results we present in this work consider only circular genomes, for which the underlying breakpoint graph is more regular and composed of cycles only. With linear chromosomes the graph also includes paths that may be of odd or of even length. Often the studies of genomic distances for genomes with linear chromosomes can be adapted to circular genomes and *vice-versa* [8], and we believe the same could be done for the problems that we study here, as we discuss in the end of the paper.

2 Definitions and 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 direct orientation or by the symbol $\overline{\mathtt{X}}$ if it is read in reverse orientation. For example, the circular string $(\mathtt{1}\,\overline{\mathtt{3}}\,\mathtt{2})$ (flanked by parentheses) represents a circular chromosome K, shown in Figure 1, composed of three genes from the families $\mathtt{1}$, $\mathtt{2}$ and $\mathtt{3}$. Note that K can be equally represented by any circular rotation of the given string and additionally by $(\overline{\mathtt{2}}\,\mathtt{3}\,\overline{\mathtt{1}})$ or any of its circular rotations.

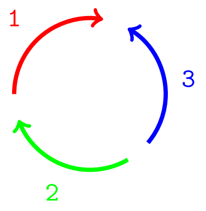


Figure 1 Graphical representation of circular chromosome $K = (\mathtt{1}\overline{\mathtt{3}}\mathtt{2})$.

We can also represent a gene from family $\mathtt{X}$ referring to its *extremities* $\mathtt{X}^h$ (head) and $\mathtt{X}^t$ (tail). For example, we could represent the circular chromosome K above by $(\mathtt{1}^t\mathtt{1}^h\mathtt{3}^h\mathtt{3}^t\mathtt{2}^t\mathtt{2}^h)$ or $(\mathtt{2}^h\mathtt{2}^t\mathtt{3}^t\mathtt{3}^h\mathtt{1}^h\mathtt{1}^t)$, or by any of their circular rotations. Recall that a gene is an *occurrence* of a family, therefore distinct genes from the same family are represented by the same symbol. The *adjacencies* in a chromosome are the neighboring extremities of distinct genes. In the given example, the adjacencies in K are $\{\mathtt{1}^h\mathtt{3}^h, \mathtt{3}^t\mathtt{2}^t, \mathtt{2}^h\mathtt{1}^t\}$. Note that an adjacency has no orientation, that is, an adjacency between extremities $\mathtt{1}^h$ and $\mathtt{3}^h$ can be equally represented by $\mathtt{1}^h\mathtt{3}^h$ and by $\mathtt{3}^h\mathtt{1}^h$. In the particular case of a single-gene circular chromosome, e.g. $(\mathtt{4})$, an adjacency exceptionally occurs between the extremities of the same gene (here $\mathtt{4}^h\mathtt{4}^t$).

A *genome* is then a set of chromosomes and we denote by $\mathcal{F}(\mathbb{G})$ the set of gene families that occur in the chromosomes of genome $\mathbb{G}$. In addition, we denote by $\mathcal{A}(\mathbb{G})$ the multiset of adjacencies that occur in the chromosomes of $\mathbb{G}$. 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 occurs exactly twice. These two copies of the same adjacency in a doubled genome are called *paralogous adjacencies*. Observe that distinct doubled genomes can have exactly the same adjacencies, as we show in Table 1, where we also give examples of singular and duplicated genomes.

2.1 Comparing canonical genomes

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

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.

Singular genome (each family occurs once)	$\mathbb{S} = \{(\mathtt{1}\,\overline{\mathtt{3}}\,\mathtt{2})\,(\mathtt{4})\}$	$\begin{cases}\mathcal{F}(\mathbb{S}) = \{\mathtt{1},\mathtt{2},\mathtt{3},\mathtt{4}\} \\ \mathcal{A}(\mathbb{S}) = \{\mathtt{1}^h\mathtt{3}^h, \mathtt{3}^t\mathtt{2}^t, \mathtt{2}^h\mathtt{1}^t, \mathtt{4}^h\mathtt{4}^t\}\end{cases}$
Duplicated genome (each family occurs twice)	$\mathbb{D} = \{(\mathtt{1}\,\mathtt{2}\,\overline{\mathtt{3}}\,\mathtt{1}\,\overline{\mathtt{3}}\,\mathtt{2})\}$	$\begin{cases}\mathcal{F}(\mathbb{D}) = \{\mathtt{1},\mathtt{2},\mathtt{3}\} \\ \mathcal{A}(\mathbb{D}) = \{\mathtt{1}^h\mathtt{2}^t, \mathtt{2}^h\mathtt{3}^h, \mathtt{3}^t\mathtt{1}^t, \mathtt{1}^h\mathtt{3}^h, \mathtt{3}^t\mathtt{2}^t, \mathtt{2}^h\mathtt{1}^t\}\end{cases}$
Doubled genomes (each adj. occurs twice)	$\mathbb{B}_1 = \{(\mathtt{1}\,\mathtt{2}\,\mathtt{3})\,(\mathtt{1}\,\mathtt{2}\,\mathtt{3})\}$ $\mathbb{B}_2 = \{(\mathtt{1}\,\mathtt{2}\,\mathtt{3}\,\mathtt{1}\,\mathtt{2}\,\mathtt{3})\}$	$\begin{cases}\mathcal{F}(\mathbb{B}_i) = \{\mathtt{1},\mathtt{2},\mathtt{3}\} \\ \mathcal{A}(\mathbb{B}_i) = \{\mathtt{1}^h\mathtt{2}^t, \mathtt{2}^h\mathtt{3}^t, \mathtt{3}^h\mathtt{1}^t, \mathtt{1}^h\mathtt{2}^t, \mathtt{2}^h\mathtt{3}^t, \mathtt{3}^h\mathtt{1}^t\}\end{cases}$

2.1.1 Breakpoint distance

A simple way of comparing canonical genomes $\mathbb{S}_1$ and $\mathbb{S}_2$ is by searching for their *common adjacencies*, which occur in both $\mathbb{S}_1$ and $\mathbb{S}_2$. For circular canonical genomes $\mathbb{S}_1$ and $\mathbb{S}_2$ the *breakpoint distance*, denoted by d_{BP}, can be computed as follows [11]:

$$\mathrm{d}_{\mathrm{BP}}(\mathbb{S}_1, \mathbb{S}_2) = n_* - |\mathcal{A}(\mathbb{S}_1) \cap \mathcal{A}(\mathbb{S}_2)|, \quad \text{where } n_* = |\mathcal{F}_*|.$$

For $\mathbb{S}_1 = \{(\mathtt{1}\,\overline{\mathtt{3}}\,\mathtt{2})\,(\mathtt{4})\}$ and $\mathbb{S}_2 = \{(\mathtt{1}\,\mathtt{2})\,(\mathtt{3}\,\overline{\mathtt{4}})\}$, the common adjacencies are $\mathcal{A}(\mathbb{S}_1) \cap \mathcal{A}(\mathbb{S}_2) = \{\mathtt{1}^t\mathtt{2}^h\}$. Since $n_* = 4$, their breakpoint distance is $\mathrm{d}_{\mathrm{BP}}(\mathbb{S}_1, \mathbb{S}_2) = 3$.

2.1.2 DCJ distance and breakpoint graph

Another way of comparing two genomes is by searching for the mininum number of rearrangements transforming one genome into the other. A very useful model for this task is called *double cut and join* (DCJ) [12]. Basically, given a genome, a DCJ is the operation that breaks two adjacencies and rejoins the open extremities in a different way.

For example, consider the chromosome $K = (\mathtt{1}\,\mathtt{2}\,\mathtt{3}\,\mathtt{4})$ and a DCJ that cuts K between genes $\mathtt{1}$ and $\mathtt{2}$ and between genes $\mathtt{3}$ and $\mathtt{4}$, creating segments $\bullet\mathtt{2}\,\mathtt{3}\bullet$ and $\bullet\mathtt{4}\,\mathtt{1}\bullet$ (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 $K' = (\mathtt{1}\,\overline{\mathtt{3}}\,\overline{\mathtt{2}}\,\mathtt{4})$, that is, the described DCJ operation is an inversion transforming K into K'. Besides inversions, in circular genomes a DCJ operation can also represent a circular fission or a circular fusion. The *DCJ distance* $\mathrm{d}_{\mathrm{DCJ}}$ is then the minimum number of DCJ operations that transform one genome into the other.

The DCJ distance can be easily computed with the help of the following structure. Given two canonical circular genomes $\mathbb{S}_1$ and $\mathbb{S}_2$, their *breakpoint graph* $BG(\mathbb{S}_1, \mathbb{S}_2) = (V, E)$ is a multigraph representing the adjacencies of $\mathbb{S}_1$ and $\mathbb{S}_2$ [2]. 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. An example is given in Figure 2.

The breakpoint graph of canonical circular genomes is a simple collection of cycles of even length, where each cycle alternates between $\mathbb{S}_1$-edges and $\mathbb{S}_2$-edges. We call *k-cycle* a cycle of length k. Note that a common adjacency of $\mathbb{S}_1$ and $\mathbb{S}_2$ corresponds to a 2-cycle in $BG(\mathbb{S}_1, \mathbb{S}_2)$. Furthermore, if $\mathbb{S}_1 = \mathbb{S}_2$, their breakpoint graph is composed of exactly n_* 2-cycles. Otherwise, if $\mathbb{S}_1 \neq \mathbb{S}_2$, their breakpoint graph is composed of $c < n_*$ cycles. It has

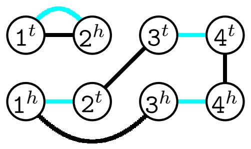


Figure 2 Breakpoint graph of genomes $\mathbb{S}_1 = \{(1\,\overline{3}\,2)\,(4)\}$ and $\mathbb{S}_2 = \{(1\,2)\,(3\,\overline{4})\}$. The colors distinguish the edge types: $\mathbb{S}_1$-edges are drawn in black and $\mathbb{S}_2$-edges are drawn in blue.

been shown that one DCJ operation can create at most one new cycle. Such a DCJ is called a *split DCJ*. Moreover, it was proven that it is possible to transform one genome into another with split DCJs only. Therefore, the DCJ distance of two genomes $\mathbb{S}_1$ and $\mathbb{S}_2$ can be directly derived from their breakpoint graph [3]:

$$\mathrm{d}_{\mathrm{DCJ}}(\mathbb{S}_1, \mathbb{S}_2) = n_* - c, \quad \text{where } n_* = |\mathcal{F}_*| \text{ and } c \text{ is the number of cycles in } BG(\mathbb{S}_1, \mathbb{S}_2).$$

If $\mathbb{S}_1 = \{(1\,\overline{3}\,2)\,(4)\}$ and $\mathbb{S}_2 = \{(1\,2)\,(3\,\overline{4})\}$, then $n_* = 4$ and $c = 2$ (see Figure 2). Consequently, their DCJ distance is $\mathrm{d}_{\mathrm{DCJ}}(\mathbb{S}_1, \mathbb{S}_2) = 2$.

2.1.3 The class of σ_k distances

For $k = 2, 4, 6, \ldots$, we denote by c_k the number of k-cycles in $BG(\mathbb{S}_1, \mathbb{S}_2)$ and by σ_k the cumulative sums $\sigma_k = c_2 + c_4 + \ldots + c_k$. Since a common adjacency of genomes $\mathbb{S}_1$ and $\mathbb{S}_2$ corresponds to a 2-cycle in $BG(\mathbb{S}_1, \mathbb{S}_2)$, their breakpoint distance can be rewritten as

$$\mathrm{d}_{\mathrm{BP}}(\mathbb{S}_1, \mathbb{S}_2) = n_* - c_2 = n_* - \sigma_2.$$

Similarly, since we have $c = c_2 + c_4 + \ldots + c_\infty$, the DCJ distance can be rewritten as

$$\mathrm{d}_{\mathrm{DCJ}}(\mathbb{S}_1, \mathbb{S}_2) = n_* - (c_2 + c_4 + \ldots + c_\infty) = n_* - \sigma_\infty.$$

Generalizing the formulas above, we can define the class σ_k-DIST of σ_k distances of two canonical circular genomes $\mathbb{S}_1$ and $\mathbb{S}_2$, for $k = 2, 4, 6, \ldots, \infty$, as follows:

$$\mathrm{d}_{\sigma_k}(\mathbb{S}_1, \mathbb{S}_2) = n_* - (c_2 + c_4 + \ldots + c_k) = n_* - \sigma_k.$$

2.2 Comparing a singular to a duplicated genome

Let $\mathbb{S}$ be a circular singular genome and $\mathbb{D}$ be a circular duplicated genome such that $\mathcal{F}(\mathbb{S}) = \mathcal{F}(\mathbb{D})$. Note that the number of adjacencies in $\mathbb{D}$ is twice the number of adjacencies in $\mathbb{S}$. In order to search for common adjacencies of $\mathbb{S}$ and $\mathbb{D}$ or to transform one genome into the other with DCJ operations, we need to somehow equalize the contents of these genomes. This can be done by *doubling* the singular genome $\mathbb{S}$. This rearrangement operation mimics a *whole genome duplication* of $\mathbb{S}$ and consists of doubling each adjacency of $\mathbb{S}$. However, when $\mathbb{S}$ is circular, it is not possible to find a unique layout of its chromosomes after the doubling: indeed, each circular chromosome of $\mathbb{S}$ can be doubled into two identical circular chromosomes, or the two copies are concatenated to each other in a single circular chromosome. Therefore, the doubling of a circular genome $\mathbb{S}$ results in a set of doubled genomes denoted by $2{\cdot}\mathbb{S}$. Note that $|2{\cdot}\mathbb{S}| = 2^r$, where r is the number of (circular) chromosomes in $\mathbb{S}$. For example, if $\mathbb{S} = \{(1\,2\,3)\}$, then $2{\cdot}\mathbb{S} = \{\mathbb{B}_1, \mathbb{B}_2\}$ with $\mathbb{B}_1 = \{(1\,2\,3)\,(1\,2\,3)\}$ and $\mathbb{B}_2 = \{(1\,2\,3\,1\,2\,3)\}$. Since all genomes in $2{\cdot}\mathbb{S}$ have exactly the same multiset of adjacencies, we can refer to its adjacencies as $\mathcal{A}(2{\cdot}\mathbb{S}) = \mathcal{A}(\mathbb{B})$ where $\mathbb{B}$ is any doubled genome in $2{\cdot}\mathbb{S}$.

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 $\mathfrak{S}^{\mathtt{a}}_{\mathtt{b}}(2\cdot\mathbb{S})$ the set of all possible genomes obtained by all distinct ways of $\binom{\mathtt{a}}{\mathtt{b}}$-singularizing each doubled genome in the set $2\cdot\mathbb{S}$.

2.2.1 Breakpoint double distance

The *breakpoint double distance* of $\mathbb{S}$ and $\mathbb{D}$, denoted by $\mathrm{d}^2_{\mathrm{BP}}(\mathbb{S},\mathbb{D})$, is defined as follows [11]:

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

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

Although the search space of this optimization problem can be huge, the solution can be found easily with a greedy algorithm [11]: Each adjacency of $\mathbb{D}$ that occurs in $\mathbb{S}$ can be fulfilled. If an adjacency that occurs twice in $\mathbb{D}$ also occurs in $\mathbb{S}$, it can be fulfilled twice in any genome from $2\cdot\mathbb{S}$. Then,

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

2.2.2 DCJ double distance and ambiguous breakpoint graph

Extending the ideas above to the DCJ model, the formulation of the *DCJ double distance* follows:

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

Here the solution space cannot be explored greedily. In fact, computing the DCJ double distance of circular genomes $\mathbb{S}$ and $\mathbb{D}$ is an NP-hard problem [11]. However, an interesting relation between its solutions and a modified breakpoint graph was established.

Given a singular genome $\mathbb{S}$ and a duplicated genome $\mathbb{D}$, their *ambiguous breakpoint graph* $ABG(\mathbb{S},\check{\mathbb{D}}) = (V,E)$ is a multigraph representing the adjacencies of any element in $\mathfrak{S}^{\mathtt{a}}_{\mathtt{b}}(2\cdot\mathbb{S})$ and a genome $\check{\mathbb{D}}$ 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 $\check{\mathbb{D}}$ there exists one $\check{\mathbb{D}}$-edge in E linking its two extremities. The $\mathbb{S}$-edges represent all adjacencies occurring in all genomes from $\mathfrak{S}^{\mathtt{a}}_{\mathtt{b}}(2\cdot\mathbb{S})$: For each adjacency $\gamma\beta$ of $\mathbb{S}$, we have the *pair of paralogous edges* $\mathcal{P}(\gamma\beta) = \{\gamma_{\mathtt{a}}\beta_{\mathtt{a}}, \gamma_{\mathtt{b}}\beta_{\mathtt{b}}\}$ and the *complementary pair of paralogous edges* $\widetilde{\mathcal{P}}(\gamma\beta) = \{\gamma_{\mathtt{a}}\beta_{\mathtt{b}}, \gamma_{\mathtt{b}}\beta_{\mathtt{a}}\}$. Note that $\widetilde{\widetilde{\mathcal{P}}}(\gamma\beta) = \mathcal{P}(\gamma\beta)$. The *square* of $\gamma\beta$ is then $\mathcal{Q}(\gamma\beta) = \mathcal{P}(\gamma\beta) \cup \widetilde{\mathcal{P}}(\gamma\beta)$. The $\mathbb{S}$-edges in the ambiguous breakpoint graph are therefore the squares of all adjacencies in $\mathbb{S}$. The number of squares obviously equals $|\mathcal{F}(\mathbb{S})|$. 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 Figure 3 (a).

Resolving a square $\mathcal{Q}(\cdot) = \mathcal{P}(\cdot)\cup\widetilde{\mathcal{P}}(\cdot)$ corresponds to *choosing* in the ambiguous breakpoint graph either the edges from $\mathcal{P}(\cdot)$ or the edges from $\widetilde{\mathcal{P}}(\cdot)$, while the complementary pair is *masked*. Resolving all squares is called *disambiguating* the ambiguous breakpoint graph. If we number the squares of $ABG(\mathbb{S},\check{\mathbb{D}})$ from 1 to $n_* = |\mathcal{F}(\mathbb{S})|$, a *disambiguation* can be

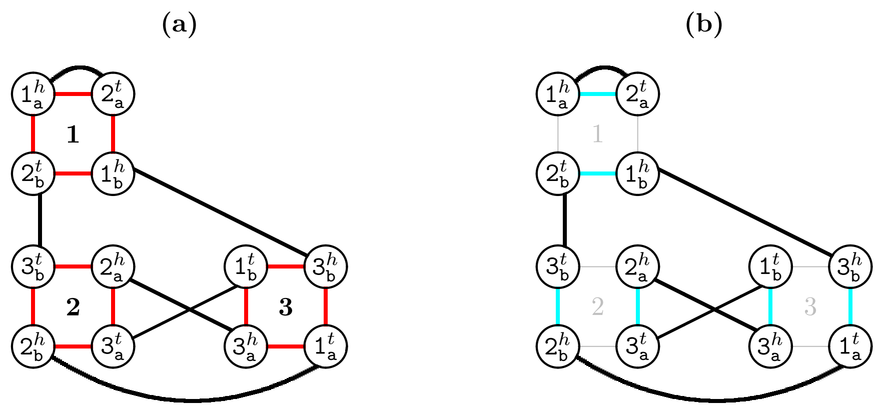


Figure 3 (a) Ambiguous breakpoint graph $ABG(\mathbb{S},\check{\mathbb{D}})$ for genomes $\mathbb{S} = \{(1\,2\,3)\}$ and $\check{\mathbb{D}} = \{(1_a\,2_a\,\overline{3}_a\,1_b\,\overline{3}_b\,2_b)\}$. The colors distinguish the edge types: $\check{\mathbb{D}}$-edges are drawn in black and $\mathbb{S}$-edges (squares) are drawn in red. (b) Induced breakpoint graph $BG(\tau,\check{\mathbb{D}})$ in which all squares are resolved by the disambiguation $\tau = (\{1^h_a 2^t_a, 1^h_b 2^t_b\}, \{2^h_a 3^t_a, 2^h_b 3^t_b\}, \{3^h_a 1^t_b, 3^h_b 1^t_a\})$, resulting in three cycles (a 2-, a 4- and a 6-cycle). This is also the breakpoint graph of $\check{\mathbb{D}}$ and $\mathbb{B} = \{(1_a\,2_a\,3_a\,1_b\,2_b\,3_b)\} \in \mathfrak{S}^a_b(2\cdot\mathbb{S})$.

represented by a tuple $\tau = (\mathcal{L}_1, \mathcal{L}_2, \ldots, \mathcal{L}_{n_*})$, where each $\mathcal{L}_i$ contains the pair of paralogous edges (either $\mathcal{P}_i$ or $\widetilde{\mathcal{P}}_i$) 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 $BG(\tau,\check{\mathbb{D}})$. Figure 3 (b) shows an example.

Computing the DCJ double distance of $\mathbb{S}$ and $\mathbb{D}$ is equivalent to finding a disambiguation τ so that the number of cycles in $BG(\tau,\check{\mathbb{D}})$ is maximized [11]. As already mentioned, this problem is NP-hard.

2.2.3 The class of σ_k double distances

We can now define the class σ_k-2Dist of σ_k double distances of a singular circular genome $\mathbb{S}$ and duplicated circular genome $\mathbb{D}$ for $k = 2, 4, 6, \ldots$ as follows:

$$\mathrm{d}^2_{\sigma_k}(\mathbb{S},\mathbb{D}) = \mathrm{d}^2_{\sigma_k}(\mathbb{S},\check{\mathbb{D}}) = \min_{\mathbb{B}\in\mathfrak{S}^a_b(2\cdot\mathbb{S})} \{\mathrm{d}_{\sigma_k}(\mathbb{B},\check{\mathbb{D}})\}, \text{ where } \check{\mathbb{D}} \text{ is any genome in } \mathfrak{S}^a_b(\mathbb{D}).$$

The complexity of (breakpoint) σ_2-2Dist being linear and the complexity of (DCJ) σ_∞-2Dist being NP-hard, the goal of our research is to increasingly determine, for $k = 4, 6, \ldots$, the complexity of each σ_k-2Dist. In this process we search for unveiling the boundary in which the complexity changes from polynomial to NP-hard: if for some $k \geq 4$ the complexity is found to be NP-hard, it is very likely that the complexity is also NP-hard for any $k' > k$.

So far we accomplished this task for the σ_4 and the σ_6 double distances, showing that both problems can be solved in linear time, as we will explain in the next section.

3 Solving the σ_k double distance problem

Similarly to the DCJ double distance, each σ_k-2Dist of $\mathbb{S}$ and $\mathbb{D}$ can be computed by finding a disambiguation τ of $ABG(\mathbb{S},\check{\mathbb{D}})$ so that the number of cycles of length at most k in the resulting breakpoint graph $BG(\tau,\check{\mathbb{D}})$ is maximized. We call the latter problem σ_k-Max.

In order to solve σ_k-Max, one idea is to visit $ABG(\mathbb{S},\check{\mathbb{D}})$ and search for candidate cycles. For describing how the graph can be screened, we need to introduce the following concepts. Two $\mathbb{S}$-edges in $ABG(\mathbb{S},\check{\mathbb{D}})$ are *incompatible* when they belong to the same square and are not paralogous. A cycle in $ABG(\mathbb{S}, \check{\mathbb{D}})$ is *valid* when it does not contain any pair of incompatible edges. Note that a valid cycle necessarily alternates $\mathbb{S}$-edges and $\check{\mathbb{D}}$-edges. Two valid cycles $C \neq C'$ in $ABG(\mathbb{S}, \check{\mathbb{D}})$ are either *intersecting*, when they share at least one edge, or *disjoint*.

It is obvious that any disambiguation τ of $ABG(\mathbb{S}, \check{\mathbb{D}})$ is composed of disjoint valid cycles. Let the *k-score* of τ, denoted by $\sigma_k(\tau)$, be the number of cycles of length at most k in $BG(\tau, \check{\mathbb{D}})$. The *$k$-score* of $ABG(\mathbb{S}, \check{\mathbb{D}})$ is the score of an optimal disambiguation for σ_k-MAX. The *switching* operation of the i-th element of a disambiguation $\tau = (\mathcal{L}_1, \mathcal{L}_2, \ldots, \mathcal{L}_i \ldots, \mathcal{L}_{n_*})$ is denoted by $\widetilde{s}(\tau, i)$ and replaces value $\mathcal{L}_i$ by $\widetilde{\mathcal{L}_i}$ resulting in $\tau' = (\mathcal{L}_1, \mathcal{L}_2, \ldots, \widetilde{\mathcal{L}_i} \ldots, \mathcal{L}_{n_*})$. A choice of paralogous edges resolving a given square $\mathcal{Q}_i$ can be *fixed* for any disambiguation, meaning that the pair assigned to $\mathcal{Q}_i$ can no longer be switched. In this case, $\mathcal{Q}_i$ is itself said to be *fixed*.

3.1 Common adjacencies are preserved in any σ_k double distance

Let τ be an optimal disambiguation for σ_k-MAX of $ABG(\mathbb{S}, \check{\mathbb{D}})$. If a cycle $C \in BG(\tau, \check{\mathbb{D}})$ is disjoint from any cycle distinct from C in any other optimal disambiguation, then C must be part of all optimal disambiguations and is itself said to be *optimal*.

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

Proof. The proof is sketched in Figure 4. It is clear that any 2-cycle C in $ABG(\mathbb{S}, \mathbb{D})$ is valid. Suppose that an optimal disambiguation τ induces a cycle $D \neq C$, such that C and D intersect. Note that τ cannot induce C. Since two 2-cycles cannot intersect with each other, it is clear that $|D| > 2$. Let $\mathcal{Q}_i$ be the square containing the $\mathbb{S}$-edge that is present in C and let $\tau' = \widetilde{s}(\tau, i)$. The disambiguation τ' induces the same cycles as τ, except that D is split into and replaced by C and D'. Note that $|C| = 2 \leq k$ and $|D'| = |D| - 2$, therefore we have $\sigma_k(\tau') > \sigma_k(\tau)$, contradicting the assumption that τ is optimal. ◀

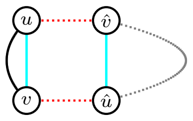


Figure 4 Illustration of the optimality of every 2-cycle. The gray path connecting vertices $\hat{v}$ and $\hat{u}$ is necessarily odd with length at least one and alternates $\check{\mathbb{D}}$- and $\mathbb{S}$-edges. The 2-cycle $C = (uv)$ intersects the longer cycle $D = (u\hat{v} \ldots \hat{u}v)$. Any disambiguation containing (red edges) $\widetilde{\mathcal{P}} = \{u\hat{v}, \hat{u}v\}$ induces D and can be improved by switching $\widetilde{\mathcal{P}}$ to (blue edges) $\mathcal{P} = \{uv, \hat{u}\hat{v}\}$, inducing, instead of D, the 2-cycle C and the cycle $D' = (\hat{v} \ldots \hat{u})$ (which is shorter than D).

This lemma is a generalization of the (breakpoint) σ_2-MAX and guarantees that all common adjacencies are preserved in any σ_k-2DIST, including the NP-hard (DCJ) σ_∞-2DIST. A square that has at least one $\mathbb{S}$-edge in a 2-cycle is called a *$\{2\}$-square*. From now on we assume that these $\{2\}$-squares are fixed so that all existing 2-cycles are induced.

3.2 A linear time greedy algorithm for the σ_4 double distance

Differently from 2-cycles, two valid 4-cycles can intersect with each other. However, two intersecting 4-cycles are always part of two co-optimal disambiguations of σ_4-MAX.

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

Proof. All possible patterns are represented in Figure 5: (a) two co-optimal valid 4-cycles within a single square; (b) – (d) 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. ◀

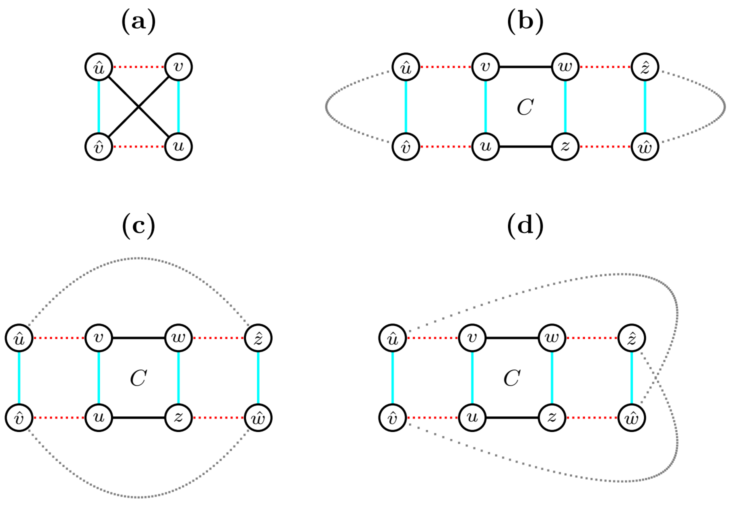


Figure 5 Illustration of the σ_4-MAX co-optimality of every valid 4-cycle not intersecting a 2-cycle. In this picture, each gray path is necessarily odd with length at least one and alternates $\check{\mathbb{D}}$- and $\mathbb{S}$-edges. In (a) any optimal solution includes either the blue edges inducing 4-cycle $(uv\hat{v}\hat{u})$ or the red edges inducing 4-cycle $(u\hat{v}v\hat{u})$. Parts (b) – (c) show the 4-cycle $C = (uvwz)$ in the center, induced by blue edges. In (b) 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 (c) 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 (d) 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.

A consequence of this lemma is that an optimal disambiguation of σ_4-MAX can be obtained greedily: After fixing the squares containing edges that are part of 2-cycles, traverse the remainder of the graph and, for each valid 4-cycle C that is found, fix the square(s) containing $\mathbb{S}$-edges that are part of C. When this part is accomplished the remaining squares can be fixed arbitrarily.

3.3 A linear time algorithm for the σ_6 double distance

In this section we may refer to a valid 4- or 6-cycle as a $\{4..6\}$-cycle. It is easy to see that $\{4..6\}$-cycles can intersect with each other. Moreover, for the σ_6-MAX, not every $\{4..6\}$-cycle is induced by at least one optimal disambiguation. For that reason, a greedy algorithm does not work here. Still, we can solve σ_6-MAX in linear time, as we will describe in the following.

Pruning the ambiguous breakpoint graph

We proceed with a 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 $\{4..6\}$-cycle. This results in a $\{6\}$-*pruned* ambiguous breakpoint graph $ABG_{\{6\}}(\mathbb{S}, \check{\mathbb{D}})$.

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 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 $\{4..6\}$-cycle involving e. If no such cycle 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 $ABG_{\{6\}}(\mathbb{S}, \check{\mathbb{D}})$.

Note that, for any square $\mathcal{Q}_i$ with $1 \le i \le n_*$, graph $ABG_{\{6\}}(\mathbb{S}, \check{\mathbb{D}})$ might contain either (a) all edges, or (b) only three edges, or (c) only two edges each one being from a distinct pair of paralogous edges, or (d) only two edges being from the same pair of paralogous edges, or (e) a single edge, or (f) no edge (see Figure 6). In cases (a), (b) and (c), $\mathcal{Q}_i$ is still ambiguous, while for cases (d), (e) and (f) $\mathcal{Q}_i$ is already resolved. We assume that these resolved squares are fixed according to the remaining paralogous edges in cases (d) and (e) or arbitrarily in case (f).

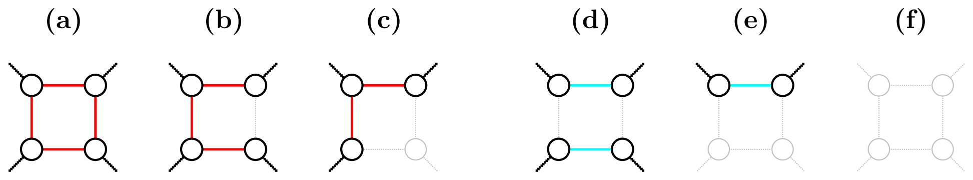


Figure 6 Possible (partial) squares of prunned $ABG_{\{6\}}(\mathbb{S}, \check{\mathbb{D}})$. Shadowed parts represent the removed elements: $\mathbb{S}$-edges that are incompatible with 2-cycles and/or edges and vertices that cannot be part of a $\{4..6\}$-cycle. Cases (a), (b) and (c) are ambiguous, cases (d) and (e) are resolved and case (f) is arbitrarily resolved.

Let the 6-*score* of $ABG_{\{6\}}(\mathbb{S}, \check{\mathbb{D}})$ be the score of a disambiguation that maximizes $c_2 + c_4 + c_6$ in $ABG_{\{6\}}(\mathbb{S}, \check{\mathbb{D}})$. Obviously, a disambiguation giving the 6-score of $ABG_{\{6\}}(\mathbb{S}, \check{\mathbb{D}})$ is an optimal disambiguation for σ_6-MAX. Therefore we can search for optimally resolving the remaining ambiguous squares by analyzing the smaller pruned graph $ABG_{\{6\}}(\mathbb{S}, \check{\mathbb{D}})$. Furthermore, the problem can be solved independently for each of the connected components of $ABG_{\{6\}}(\mathbb{S}, \check{\mathbb{D}})$, so that the result of σ_6-MAX is the sum $\sum_{G \in ABG_{\{6\}}(\mathbb{S}, \check{\mathbb{D}})} \sigma_6(G)$, where $\sigma_6(G)$ is the 6-score (maximum number of disjoint 2- and $\{4..6\}$-cycles) of component G.

Describing the connected components of the pruned graph

We will now describe the properties of the pruned graph $ABG_{\{6\}}(\mathbb{S}, \check{\mathbb{D}})$. An $\mathbb{S}$-edge (respectively $\check{\mathbb{D}}$-edge) that is present in $ABG_{\{6\}}(\mathbb{S}, \check{\mathbb{D}})$ is called an $\mathbb{S}_{\{6\}}$-edge (respectively $\check{\mathbb{D}}_{\{6\}}$-edge). Any square that is still ambiguous in $ABG_{\{6\}}(\mathbb{S}, \check{\mathbb{D}})$ is called a $\{6\}$-*ambiguous square.* A $\{6\}$-ambiguous square $\mathcal{Q}_i$ is a $\{6\}$-*neighbor* of another $\{6\}$-ambiguous square $\mathcal{Q}_j$ when a vertex of $\mathcal{Q}_i$ is connected to a vertex of $\mathcal{Q}_j$ by a $\check{\mathbb{D}}_{\{6\}}$-edge.

▶ **Proposition 3.** *Each connected component G of $ABG_{\{6\}}(\mathbb{S}, \check{\mathbb{D}})$ is of one of the two types:*
1. Ambiguous*: G includes at least one $\{6\}$-ambiguous square and no 2-cycle;*
2. Resolved (trivial)*: G is a simple valid 2-, 4- or 6-cycle;*

Proof. By construction all 2-cycles are disconnected from the other components of the graph $ABG_{\{6\}}(\mathbb{S}, \check{\mathbb{D}})$. Therefore, if a component G of $ABG_{\{6\}}(\mathbb{S}, \check{\mathbb{D}})$ has a $\{6\}$-ambiguous square, G cannot include any 2-cycle. Now let G be a connected component that does not include a $\{6\}$-ambiguous square. Then any vertex in G has exactly one incident $\mathbb{S}$-edge and one incident $\check{\mathbb{D}}$-edge. Therefore G must be a simple valid 2-, 4- or 6-cycle. ◀

Let $\mathcal{R}$ be the set of resolved and $\mathcal{B}$ be the set of ambiguous components of $ABG_{\{6\}}(\mathbb{S}, \check{\mathbb{D}})$. The result of σ_6-MAX can be then computed with the formula $|\mathcal{R}| + \sum_{G \in \mathcal{B}} \sigma_6(G)$.

Computing the 6-score of an ambiguous component of the pruned graph

For solving σ_6-MAX, we will now describe the only missing part: a procedure that computes the 6-score of an ambiguous component $G \in \mathcal{B}$.

▶ **Proposition 4.** *Any $\check{\mathbb{D}}_{\{6\}}$-edge is part of either one or two (intersecting) $\{4..6\}$-cycles.*

Proof. By construction any $\check{\mathbb{D}}_{\{6\}}$-edge is part of at least one $\{4..6\}$-cycle. The remainder of the proof can be found in Figure 7 and in supplementary Figures 8-10 (in the appendix), which display all possible patterns showing a $\check{\mathbb{D}}$-edge in two distinct intersecting $\{4..6\}$-cycles which themselves do not intersect 2-cycles. In all cases an exhaustive search shows that the same $\check{\mathbb{D}}$-edge cannot be part of a third $\{4..6\}$-cycle. ◀

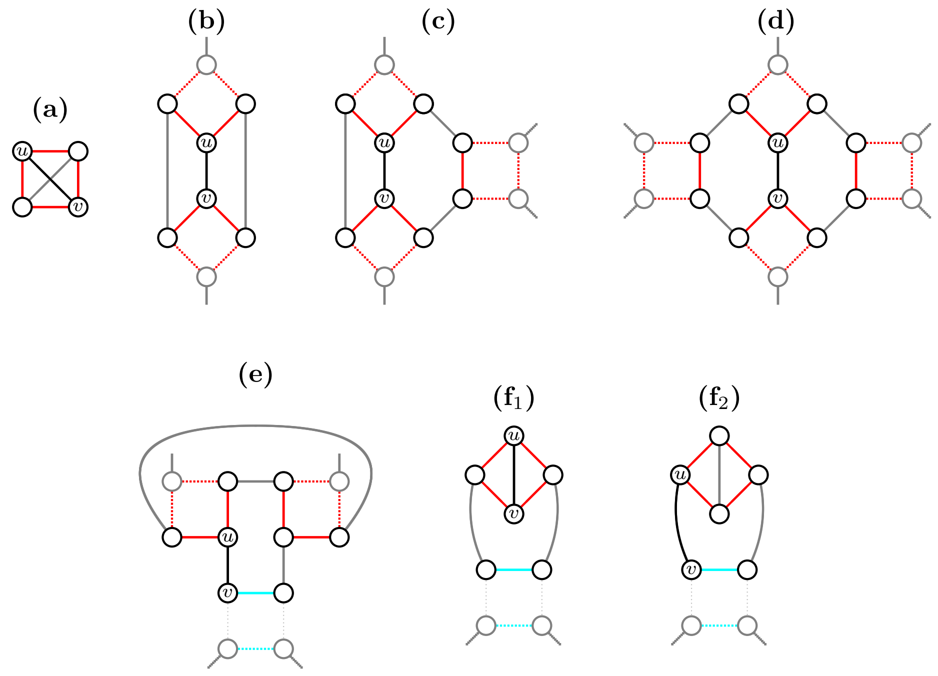


Figure 7 Patterns showing a $\check{\mathbb{D}}$-edge uv in two distinct intersecting $\{4..6\}$-cycles which themselves do not intersect 2-cycles. (a) The edge uv is part of two 4-cycles within the same square. (b) – (d) The edge uv connects two distinct squares and is part of two $\{4..6\}$-cycles whose intersection is only uv. (e) The edge uv is part of two 6-cycles whose intersection is a 3-path starting in uv. (f$_1$) The edge uv connects vertices of the same square and is part of two 6-cycles. (f$_2$) The edge uv is one of the other two $\check{\mathbb{D}}$-edges in the 6-cycles of (f$_1$). In all cases an exhaustive search shows that uv cannot be part of a third $\{4..6\}$-cycle (see supplementary Figures 8-10 in the appendix). Furthermore, in each one of the cases (e) – (f$_2$) 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.

▶ **Proposition 5.** *Any $\mathbb{S}_{\{6\}}$-edge of a $\{6\}$-ambiguous square $\mathcal{Q}_i$ is part of exactly one $\{4..6\}$-cycle.*

Proof. If an $\mathbb{S}_{\{6\}}$-edge e is in a $\{6\}$-ambiguous square $\mathcal{Q}_i$, it "shares" the same $\check{\mathbb{D}}_{\{6\}}$-edge d with another $\mathbb{S}_{\{6\}}$-edge e' from the same square $\mathcal{Q}_i$. In this case the $\check{\mathbb{D}}_{\{6\}}$-edge d is part of exactly two $\{4..6\}$-cycles and each of the $\mathbb{S}_{\{6\}}$-edges e and e' can be part of only one $\{4..6\}$-cycle. ◀

The proposition above immediately implies the following:

▶ **Corollary 6.** *Choosing an $\mathbb{S}_{\{6\}}$-edge e of a $\{6\}$-ambiguous square $\mathcal{Q}_i$ (and its paralogous edge $\hat{e}$) implies a unique disambiguation of all $\{6\}$-neighbors of $\mathcal{Q}_i$.*

Consider an ambiguous component G of $ABG_{\{6\}}(\mathbb{S}, \check{\mathbb{D}})$ and denote by τ_G a disambiguation including only the $\{6\}$-ambiguous squares of G. Now let τ_G be obtained with Algorithm 1, using a recursion of the statement of Corollary 6, as described in Algorithm 2. Since the recursion first tests whether each neighbor was already resolved or fixed (lines 1 and 8), it visits and resolves each $\{6\}$-ambiguous square of G exactly once. The resolving procedure itself visits the 6-neighborhood of up to two $\mathbb{S}_{\{6\}}$-edges and can be done in constant time. Therefore the whole recursive procedure takes linear time $O(m)$, where m is the number of $\{6\}$-ambiguous squares in G.

Algorithm 1 StraightComponentDisambiguation.

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

Algorithm 2 ResolveNeighbors.

```
Input: A partially filled disambiguation $\tau_G$ and an $\mathbb{S}$-edge $uv$ of component $G$
    /* $\mathbb{S}$-edge $uv$ is adjacent to two $\check{\mathbb{D}}_{\{6\}}$-edges $uz$ and $vw$ */
1: if vertex $z$ is not in a resolved or fixed square then
2:    $i \leftarrow$ index in $\tau_G$ of square containing $z$;
3:    $e \leftarrow$ $\mathbb{S}$-edge $zx$ of $\mathcal{Q}_i$ forming a $\{4..6\}$-cycle with $uv$ and $uz$;
4:    $\tau_G[i] \leftarrow \{e, \hat{e}\}$;
5:    ResolveNeighbors($\tau_G, e$);
6:    if $\hat{e}$ is an $\mathbb{S}_{\{6\}}$-edge then
7:       ResolveNeighbors($\tau_G, \hat{e}$);
8: if vertex $w$ is not in a resolved or fixed square then
9:    $j \leftarrow$ index in $\tau_G$ of square containing $w$;
10:   $f \leftarrow$ $\mathbb{S}$-edge $wy$ of $\mathcal{Q}_j$ forming a $\{4..6\}$-cycle with $uv$ and $vw$;
11:   $\tau_G[j] \leftarrow \{f, \hat{f}\}$;
12:   ResolveNeighbors($\tau_G, f$);
13:   if $\hat{f}$ is an $\mathbb{S}_{\{6\}}$-edge then
14:      ResolveNeighbors($\tau_G, \hat{f}$);
15: return
```

Let $\widetilde{s}(\tau_G)$ be the disambiguation obtained by switching all squares in τ_G. Examples of the two disambiguations τ_G and $\widetilde{s}(\tau_G)$ are given in supplementary Figures 11, 12 and 13 in the appendix. Now denote by τ_G^* a disambiguation with highest 6-score among τ_G and $\widetilde{s}(\tau_G)$.

▶ **Corollary 7.** *The disambiguation τ_G^* is optimal.*

4 Discussion and open problems

Several combinatorial problems related to genome evolution and ancestral reconstruction, including median, guided halving and double distance, have the distance problem as a basic unit. Interestingly, for circular genomes these three problems can be solved in polynomial time when they are built upon the breakpoint distance, while they are NP-hard when they are built upon the (rearrangement) DCJ distance.

Our study started as an exploration of the complexity space of the double distance between these two extremes. Therefore we considered a new 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 the results of investigating the complexity of the double distance first for the σ_4, then for the σ_6 distance, assuming that the given genomes have only circular chromosomes. For both cases we devised linear time algorithms, built on a variation of the breakpoint graph called ambiguous breakpoint graph.

The breakpoint graph of genomes with linear chromosomes includes, besides the cycles of even length, paths of odd and even length. It is known that the breakpoint double distance of genomes with linear chromosomes can also be solved in linear time [11]. Furthermore, we know that each linear chromosome in the singular genome $\mathbb{S}$ "removes" from the ambiguous breakpoint graph one ambiguous square (corresponding to the four vertices that will then be at the end of paths), reducing the number of choices to be made. We have not yet worked out the details, but we see no reason why our linear time algorithms for the σ_4 and the σ_6 double distances cannot be extended to take genomes with linear chromosomes into account.

More far-reaching, we conjecture 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 $k' > k$: as k grows each edge of the ambiguous breakpoint graph may be part of a larger number of valid cycles of length at most k, making the description of the combinatorial space more complex. We expect that by finding the smallest k for which the σ_k double distance is NP-hard we will be able to confirm our conjecture. In any case, the natural next step in our research is to study the σ_8 double distance.

A more 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, 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 breakpoint graph, and the combinatorial space of the distinct possibilities of doing that could not yet be described.

References

1 Max Alekseyev and Pavel A. Pevzner. Colored de Bruijn graphs and the genome halving problem. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 4(1):98–107, 2008. doi:10.1109/TCBB.2007.1002.

2 Vineet Bafna and Pavel A. Pevzner. Genome rearrangements and sorting by reversals. In *Proceedings of FOCS 1993*, pages 148–157, 1993. `doi:10.1109/SFCS.1993.366872`.

3 Anne Bergeron, Julia Mixtacki, and Jens Stoye. A unifying view of genome rearrangements. In *Proceedings of WABI 2006*, volume 4175 of *LNBI*, pages 163–173, 2006. `doi:10.1007/11851561_16`.

4 Cedric Chauve. Personal communication in Dagstuhl Seminar no. 18451 - Genomics, Pattern Avoidance, and Statistical Mechanics, November 2018.

5 Nadia El-Mabrouk and David Sankoff. The reconstruction of doubled genomes. *SIAM Journal on Computing*, 32(3):754–792, 2003. `doi:10.1137/S0097539700377177`.

6 Sridhar Hannenhalli and Pavel A. Pevzner. Transforming men into mice (polynomial algorithm for genomic distance problem). In *Proceedings of FOCS 1995*, pages 581–592. IEEE Press, 1995. `doi:10.1109/SFCS.1995.492588`.

7 Sridhar Hannenhalli and Pavel A. Pevzner. Transforming cabbage into turnip: polynomial algorithm for sorting signed permutations by reversals. *Journal of the ACM*, 46(1):1–27, 1999. `doi:10.1145/300515.300516`.

8 João Meidanis, Maria Emília M. T. Walter, and Zanoni Dias. Reversal distance of signed circular chromosomes. Technical Report IC–00–23, Institute of Computing, University of Campinas, Brazil, 2000. URL: `www.ic.unicamp.br/~reltech/2000/00-23.pdf`.

9 Julia Mixtacki. Genome halving under DCJ revisited. In *Proceedings of COCOON 2008*, volume 5092 of *LNCS*, pages 276–286. Springer Verlag, 2008. `doi:10.1007/978-3-540-69733-6_28`.

10 David Sankoff. Edit distance for genome comparison based on non-local operations. In *Proceedings of CPM 1992*, volume 644 of *LNCS*, pages 121–135, 1992. `doi:10.1007/3-540-56024-6_10`.

11 Eric Tannier, Chunfang Zheng, and David Sankoff. Multichromosomal median and halving problems under different genomic distances. *BMC Bioinformatics*, 10:120, 2009. `doi:10.1186/1471-2105-10-120`.

12 Sophia Yancopoulos, Oliver Attie, and Richard Friedberg. Efficient sorting of genomic permutations by translocation, inversion and block interchange. *Bioinformatics*, 21(16):3340–3346, 2005. `doi:10.1093/bioinformatics/bti535`.

A Supplementary figures

In Figures 8, 9 and 10 we show the exhaustive exploration of the most complex patterns (b) – (d) of Figure 7, showing that uv (the black edge connecting black vertices) can only be part of two intersecting $\{4..6\}$-cycles. Some patterns have very similar structure. If this is the case for patterns P and P' and when the distance (length of shortest path) from uv to the open ends in P' is bigger than in P, we say that P' is *worse* than P. In other words, if uv cannot be in a third $\{4..6\}$-cycle in P, the same is true for P'.

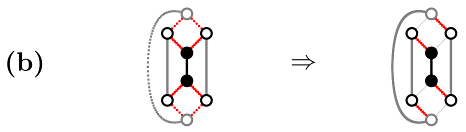
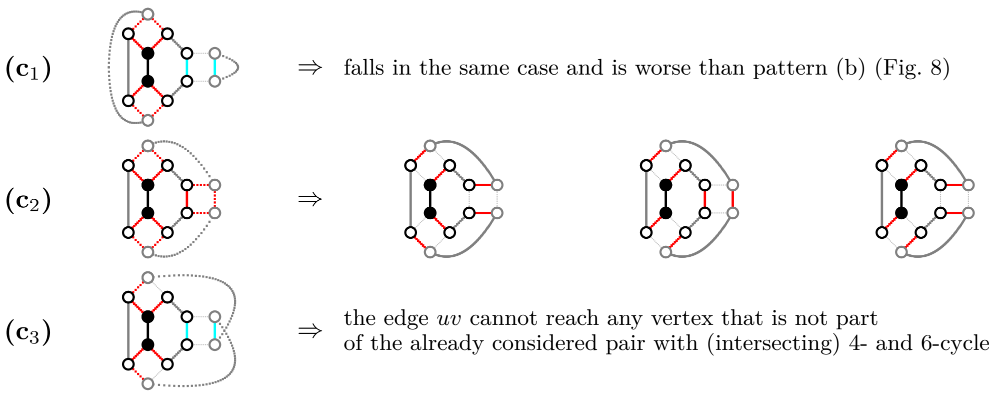


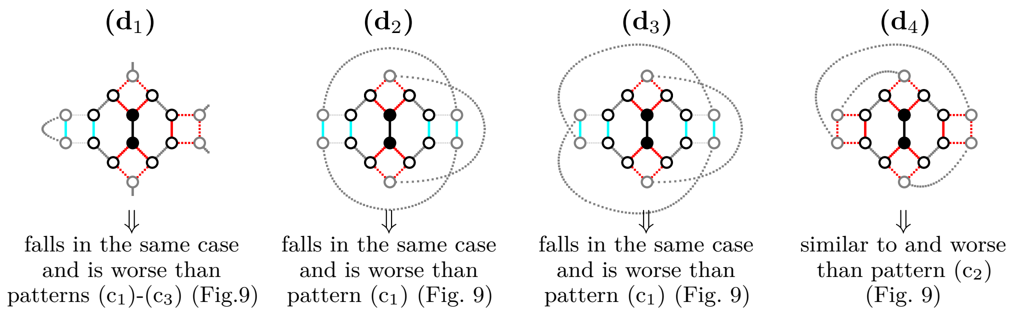
Figure 8 Exhaustive exploration of the pattern from Figure 7 (b). The only possibility to be explored is by connecting the top to the bottom vertex, and the smallest third cycle including edge uv in this scenario is an 8-cycle. (A symmetric case was omitted.)

In Figures 11, 12 and 13 we give examples of running Algorithm 1 and computing a straight disambiguation for three distinct ambiguous components.



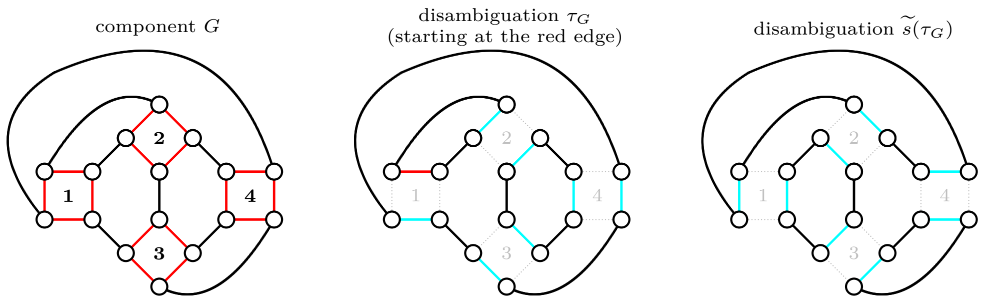
(c$_1$)
⇒ falls in the same case and is worse than pattern (b) (Fig. 8)
(c$_2$)
⇒
(c$_3$)
⇒ the edge uv cannot reach any vertex that is not part of the already considered pair with (intersecting) 4- and 6-cycle

Figure 9 Exhaustive exploration of the pattern from Figure 7 (c) with the three possibilities of connecting the four open ends. Only (c$_2$) needs to be further explored and no alternative gives a third {4..6}-cycle. (Symmetric cases were omitted.)



(d$_1$)
(d$_2$)
(d$_3$)
(d$_4$)
falls in the same case and is worse than patterns (c$_1$)-(c$_3$) (Fig.9)
falls in the same case and is worse than pattern (c$_1$) (Fig. 9)
falls in the same case and is worse than pattern (c$_1$) (Fig. 9)
similar to and worse than pattern (c$_2$) (Fig. 9)

Figure 10 Exhaustive exploration of the pattern from Figure 7 (d). Part (d$_1$) falls in the same case and is worse than pattern (c). Parts (d$_2$) – (d$_4$) display the other possibilities of connecting the open ends and are worse than smaller scenarios already explored. (Symmetric cases were omitted.)



component G
disambiguation τ_G (starting at the red edge)
disambiguation $\widetilde{s}(\tau_G)$
1
2
3
4

Figure 11 Example of the two best candidate disambiguations of a component of a {6}-pruned ambiguous breakpoint graph. Both candidates are optimal, inducing one 4- and two 6-cycles.

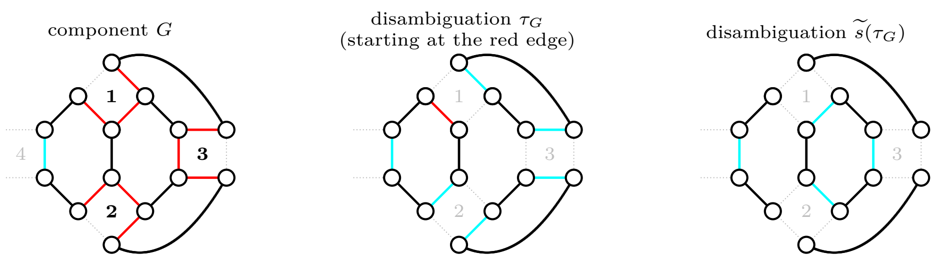

component G
1
2
3
4
disambiguation τ_G
(starting at the red edge)
1
2
3
disambiguation $\widetilde{s}(\tau_G)$
1
2
3

Figure 12 Example of the two best candidate disambiguations of a component G of a $\{6\}$-pruned ambiguous breakpoint graph. While τ_G is optimal and induces two 4-cycles and one 6-cycle, the alternative $\widetilde{s}(\tau_G)$ induces only a single 6-cycle.

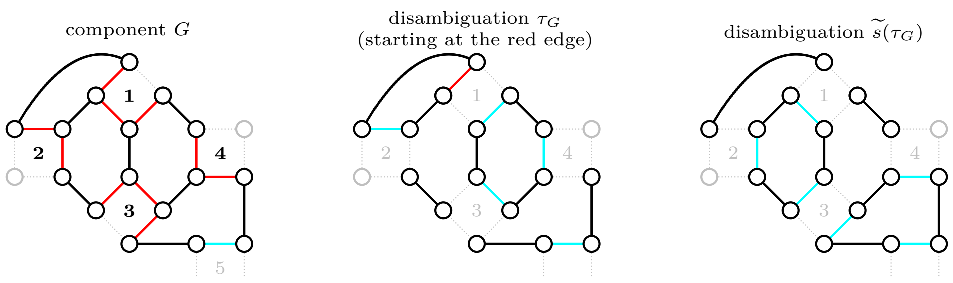

component G
1
2
3
4
5
disambiguation τ_G
(starting at the red edge)
1
2
3
4
disambiguation $\widetilde{s}(\tau_G)$
1
2
3
4

Figure 13 Example of the two best candidate disambiguations of a component G of a $\{6\}$-pruned ambiguous breakpoint graph. Both candidates are optimal: τ_G induces one 4- and one 6-cycle, while the alternative $\widetilde{s}(\tau_G)$ induces two 6-cycles.