A General Framework for Genome Rearrangement with Biological Constraints

Pijus Simonaitis[1], Annie Chateau[1,2], and Krister M. Swenson[1,2(✉)]

[1] LIRMM, CNRS – Université Montpellier, 161 rue Ada, 34392 Montpellier, France
swenson@lirmm.fr
[2] Institut de Biologie Computationnelle (IBC), Montpellier, France

Abstract. This paper generalizes previous studies on genome rearrangement under biological constraints, using double cut and join (DCJ). We propose a model for weighted DCJ, along with a family of optimization problems called φ-MCPS (Minimum Cost Parsimonious Scenario), that are based on edge labeled graphs. After embedding known results in our framework, we show how to compute solutions to general instances of φ-MCPS, given an algorithm to compute φ-MCPS on a circular genome with exactly one occurrence of each gene. These general instances can have an arbitrary number of circular and linear chromosomes, and arbitrary gene content. The practicality of the framework is displayed by generalizing the results of Bulteau, Fertin, and Tannier on the Sorting by wDCJs and indels in intergenes problem, and by generalizing previous results on the Minimum Local Parsimonious Scenario problem.

Keywords: Double Cut and Join (DCJ)
Weighted genome rearrangement · Parsimonious scenario
Breakpoint graph · Maximum alternating cycle decomposition

1 Introduction

1.1 Context

The practical study of genome rearrangement between evolutionarily distant species has been limited by a lack of mathematical models capable of incorporating biological constraints. Without such constraints the number of *parsimonious* (shortest length) rearrangement scenarios between two gene orders grows exponentially with respect to the minimum number of rearrangements between genomes [11]. A natural way to mitigate this problem is to develop models that weight rearrangements according to their likelihood of occurring; a breakpoint may be more likely to occur in some intergenic regions than others.

To this end, the study of length-weighted reversals was started in the late nineties by Blanchette, Kunisawa, and Sankoff [9]. Baudet, Dias, and Dias

M. Blanchette and A. Ouangraoua (Eds.): RECOMB-CG 2018, LNBI 11183, pp. 49–71, 2018.
https://doi.org/10.1007/978-3-030-00834-5_3

present a summary of work done in this area, along with work on reversals centered around the origin of replication [2]. Recently, Tannier has published a series of papers focused on weighting intergenic regions by their length in nucleotides. In [7], Biller, Guéguen, Knibbe, and Tannier pointed out that, according to the Nadeau-Taylor model of uniform random breakage [17,18], a breakpoint is more likely to occur in a longer intergenic region. Subsequent papers by Fertin, Jean, and Tannier [15], and Bulteau, Fertin, and Tannier [12] present algorithmic results for models that take into account the length of intergenic regions. Using Hi-C data [16], Veron et *al.* along with our own study, have pointed out the importance of weighting pairs of breakpoints according to how close they tend to be in physical space [19,24]. In order to use this physical constraint, we partitioned intergenic regions into co-localized areas, and developed algorithms for computing distances that minimize the number of rearrangements that operate on breakpoints between different areas [22,23].

Much of this work is based on the mathematically clean model for genome rearrangement called *Double Cut and Join*, or *DCJ* [4,25]. Genomes are partitioned into n orthologous syntenic blocks that we will simply call *genes*. Each gene is represented by two extremities, and each chromosome is represented by an ordering of these extremities. Those extremities that are adjacent in this ordering are paired, and transformations of these pairs occur by swapping extremities of two pairs. DCJ can naturally be interpreted as a graph edit model with the use of the *breakpoint graph*, where there is an edge between gene extremities a and b for each adjacent pair. A DCJ operation replaces an edge pair $\{\{a,b\},\{c,d\}\}$ of the graph by $\{\{a,c\},\{b,d\}\}$ or $\{\{a,d\},\{b,c\}\}$. This edge edit operation on a graph is called a *2-break*.

This paper establishes a general framework for weighting rearrangements. The results are based on the problem of transforming one edge-labeled graph into another through a scenario of operations, each weighted by an arbitrary function φ. The problem, called φ-Minimum Cost Parsimonious Scenario (or φ-MCPS), asks for a scenario with a minimum number of 2-breaks, such that the sum of the costs for the operations is minimized.

1.2 Applications of Our Framework

While our framework is general, we use it to render two previous studies more practical. The first study is our work relating the likelihood of rearrangement breakpoints to the physical proximity in the nucleus [23]. This work is based on the hypothesis that two breakpoints could be confused when they are physically close. The model in this study labels the breakpoint graph edges (corresponding to intergenic regions) with fixed "colors", and the cost of a DCJ has a weight of one if the labels are different and a weight of zero if they are the same. Using that cost function, we colored intergenic regions by grouping them according to their physical proximity, as inferred by Hi-C data. Although this technique of grouping proved to make biological sense [19,22], it is far from ideal since much of the information given by the Hi-C data is lost in the labeling, and it is not immediately clear how to best compute the grouping. Our results here bypass

the complexity of grouping by allowing each DCJ to be weighted by the values taken directly from the Hi-C contact maps. We give an algorithm for φ-MCPS on a breakpoint graph with an arbitrary φ and fixed edge labels, that runs in $O(n^5)$ time in the worst case but has better parameterized complexity in practice (see Example 1). We give in Sect. 10.1 other reasons why the running times for this algorithm should remain practical.

The second study that we improve is that of Bulteau, Fertin, and Tannier [12]. Their biological constraint is based on the number of nucleotides in the intergenic regions containing breakpoints; they compute parsimonious scenarios that minimize the number of nucleotides inserted and deleted in intergenic regions. Their algorithm is restricted to instances where the breakpoint graph has only cycles (and no paths — sometimes referred to as *co-tailed* genomes). Using their $O(n \log n)$ algorithm, our framework gives an $O(n^3)$ algorithm on any breakpoint graph (see Example 3).

This is an example of how our framework simplifies algorithm design on weighted DCJs. For a weight function adhering to our general criteria of Sect. 4, future algorithm designers now need only to concentrate on developing an efficient algorithm that works on a single cycle of a breakpoint graph. Thanks to Theorem 4, they will get a polynomial time algorithm that works on a general instance for free. Section 8 shows that the same is true for approximation algorithms.

This paper is based on general results we obtain on weighted transformations of edge-labeled multi-graphs. The permitted transformations can change the connectivity of the graph through a 2-break, or change the edge labels, or both. This model not only proves to be powerful enough to subsume the previously mentioned results, but also offers other advantages. It is flexible enough so that DCJ costs can be based on the labels of edges in the breakpoint graph, or on the vertices, or a combination of both. Also, since single-gene insertions and deletions can be represented as "ghost" adjacencies [20], all of this paper applies to genomes where genes could be missing in one genome or the other. Most results can be applied to genomes with duplicate genes (as depicted in Fig. 1).

1.3 Our Model and General Results

The foundation of this paper is a renewed understanding of scenarios of 2-breaks on Eulerian graphs, a subject that has been studied not only in a restricted setting for genome rearrangement [1,4], but also in the more general settings of network design [5,6]. Although our results are about the transformation of one arbitrary Eulerian multi-graph G into another one H having the same vertex set, we find it convenient to reason in an equivalent but different setting. In the alternative setting we are given an Eulerian 2-edge-colored multi-graph with black and gray edges, the black edges being from G and the gray from H. We transform the connectivity of the black edges into the connectivity of the gray edges. Therefore, whenever we use the word *graph*, *path* (resp. *cycle*), we are referring to an Eulerian 2-edge-colored multi-graph, a path (resp. cycle) that alternates between black and gray edges. Naturally, a *cycle decomposition* of a

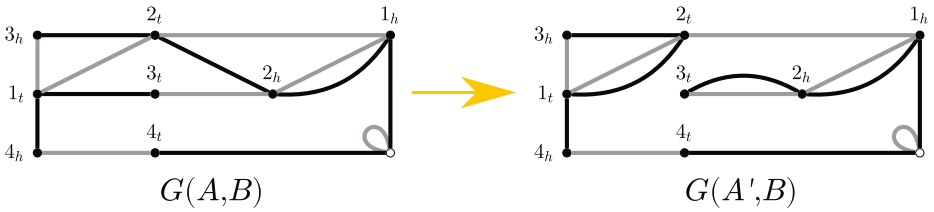

Fig. 1. Eulerian 2-edge-color multi-graphs for genomes $A = (\{3_t, 1_t\}, \{1_h, 2_h\}, \{2_t, 3_h\}), (\{4_t\}, \{4_h, 1_t\}, \{1_h\})$, $B = (\{1_h, 2_h\}, \{2_t, 1_t\}), (\{3_t, 2_h\}, \{2_t, 1_h\}, \{1_t, 3_h\})$, and $A' = (\{3_t, 2_h\}, \{2_t, 1_t\}, \{1_h, 2_h\}, \{2_t, 3_h\}), (\{4_t\}, \{4_h, 1_t\}, \{1_h\})$. Edges adjacent to a special vertex $\circ$ represent the endpoints of linear chromosomes (*e.g.* black edges $\{1_h, \circ\}$ and $\{4_t, \circ\}$). Extra edges are added for the missing genes (*e.g.* the black edge $\{2_t, 2_h\}$ and the gray edge $\{4_h, 4_t\}$), called *ghost adjacencies* in [20]. In the genomes A and A', gene 1 is repeated twice, and the operation transforming A into A' is an insertion of a gene 2, corresponding to the 2-break $G(A, B) \to G(A', B)$. A DCJ scenario transforming A' into the linear genome B includes a deletion of a gene 4.

graph is a partition of the edges of an Eulerian 2-edge-colored multi-graph into a set of alternating cycles. A *breakpoint graph* is a graph with a vertex for each gene extremity — each incident to exactly one gray and one black vertex — along with one chromosome endpoint vertex $\circ$ that could have degree as high as $2n$ (see Fig. 2). Section 2 introduces the breakpoint graph in detail, and defines the Double Cut and Join (DCJ) model.

Our model for weighting operations is primarily based on a labeling $\mathcal{L}$ of the edges, a set $\mathcal{O}$ of valid operations, and a weight function $\varphi : \mathcal{O} \to \mathbb{R}_+$. Roughly speaking, a labeled input graph can be transformed through a series of operations in $\mathcal{O}$, where an operation can change the connectivity of the black edges of the graph, and/or change the labels of the edges. Any weight function φ defines an optimization problem φ-MCPS, which asks for a scenario that minimizes the total weight of the operations. This model subsumes many previously studied weighted DCJ models, as described in Sect. 4.1.

The spine of our results is built from successive theorems that speak to the decomposability into subproblems of a φ-MCPS instance. Theorem 1 shows that a parsimonious scenario of 2-breaks transforming the black edges into the gray implies a Maximum Alternating Edge-disjoint Cycle Decomposition (or MAECD) [13]. Theorem 2 says that an optimal solution to φ-MCPS can be found using solutions to the MAECD problem, so that if φ-MCPS can be solved on a simple alternating cycle, then it can be solved on any instance. Theorem 3 says that an optimal solution to φ-MCPS on a simple alternating cycle can be found using a solution to the φ-MCPS problem on what we call a *circle*, that is, an alternating cycle that does not visit the same vertex twice (see Fig. 4).

Under the common genome model, where each gene occurs exactly once in each genome, a relationship exists between parsimonious DCJ scenarios and solutions to MAECD on a breakpoint graph [4,10]. We exploit this link in Sect. 7. Theorem 4 ties everything together; an amortized analysis shows that,

given an $O(r^t)$ algorithm for computing φ-MCPS on a circle with r edges, φ-MCPS can be calculated on a breakpoint graph in $O(n^{t+1})$ time.

Under a more general genome model, that allows for changes in copy numbers of genes (*e.g.* insertions, deletions, and duplications), the spine of our results still holds due to the convenient representation of missing genes as *ghost adjacencies* in an Eulerian 2-edge-colored multi-graph [20] (See Fig. 1). All of our results hold for pairs of genomes with non-duplicated genes, but unequal gene content. Indeed, a breakpoint graph (*i.e.* graph with limited degree for most nodes) can still represent the pair of genomes in this case.

Caprara proved that MAECD is NP-Hard for Eulerian 2-edge-colored multi-graphs where each vertex is incident to at most two gray and two black edges (which is the case when there are two copies of each gene) [13]. We present a simple integer linear program (or ILP) that solves φ-MCPS for these types of graphs, given a method to solve φ-MCPS on a circle. This ILP is likely to be unwieldy in general, since the number of variables is exponential in the number of simple alternating cycles. In the case of breakpoint graphs on specific genomes, this may not always be intractable, as the number of duplicate genes may be limited. See Sect. 10.1 for a discussion of these practical matters.

2 DCJ Scenarios for Genomes and Breakpoint Graphs

A *genome* consists of *chromosomes* that are linear or circular orders of genes separated by potential *breakpoint* regions. In Fig. 2 the tail of an arrow represents the *tail extremity*, and the head of an arrow represents the *head extremity* of a gene. We can represent a genome by a set of *adjacencies* between the gene extremities. An *adjacency* is either *internal*: an unordered pair of the extremities that are adjacent on a chromosome, or *external*: a single extremity adjacent to one of the two ends of a linear chromosome. In what follows we will suppose that two genomes A and B are partitioned into n genes each occurring exactly once in each genome, and our goal will be to transform A into B using a sequence of DCJs.

Fig. 2. Genomes A and B with their respective sets of adjacencies $\{\{1_t\},\{1_h,2_t\},\{2_h,3_h\},\{3_t\}\}$ and $\{\{1_t\},\{1_h,2_h\},\{2_t,3_h\},\{3_t\}\}$. A DCJ $\{1_h,2_t\},\{2_h,3_h\} \rightarrow \{1_h,2_h\},\{2_t,3_h\}$ transforms A into B. The transformation $G(A,B) \rightarrow G(B,B)$ is a 2-break and $G(B,B)$ is a terminal graph.

Definition 1 (double cut and join). *A DCJ cuts one or two breakpoint regions and joins the resulting ends of the chromosomes back in one of the four following ways:* $\{a,b\},\{c,d\} \rightarrow \{a,c\},\{b,d\}$*;* $\{a,b\},\{c\} \rightarrow \{a,c\},\{b\}$*;* $\{a,b\} \rightarrow \{a\},\{b\}$*; and* $\{a\},\{b\} \rightarrow \{a,b\}$*.*

We represent the pairs of the genomes with a help of a breakpoint graph [1,25].

Definition 2 (breakpoint graph). *$G(A,B)$ for genomes A and B is a 2-edge-colored Eulerian undirected multi-graph. V consists of $2n$ gene extremities and an additional vertex $\circ$. For every internal adjacency $\{a,b\} \in A$ (resp. $\{a,b\} \in B$) there is a black (resp. gray) edge $\{a,b\}$ in $G(A,B)$ and for every external adjacency $\{a\} \in A$ (resp. $\{a\} \in B$) there is a black (resp. gray) edge $\{a,\circ\}$ in $G(A,B)$. There is a number of black and gray loops $\{\circ,\circ\}$ ensuring that $d^b(G(A,B),\circ) = d^g(G(A,B),\circ) = 2n$.*

3 2-break Scenarios for 2-edge-colored Graphs

In this paper a *graph* is an Eulerian 2-edge-colored undirected multi-graph with edges colored black or gray as in Fig. 1. A graph with equal multi-sets of black and gray edges is called *terminal*, and our goal is to transform a given graph into a terminal one using 2-breaks.

Definition 3 (2-break scenario). *A 2-break replaces two black edges $\{x_1,x_2\}$ and $\{x_3,x_4\}$ by either $\{x_1,x_3\}$ and $\{x_2,x_4\}$ or $\{x_1,x_4\}$ and $\{x_2,x_3\}$. A 2-break* scenario *of* length *m is a sequence of m 2-breaks transforming a graph into a terminal one.*

Definition 4 (Eulerian graph and alternating cycle). *G is* Eulerian *if every vertex has equal black and gray degrees. A cycle is* alternating *if it is Eulerian. All use of the word* cycle *in this paper will be synonymous with alternating cycle.*

Define a Maximum Alternating Edge-disjoint Cycle Decomposition (MAECD) of a graph G as a decomposition of G into a maximum number of edge-disjoint alternating cycles. Denote the size of a MAECD of G by $c(G)$ and the number of its black edges by $e(G)$. We make a distinction between simple cycles and circles (look at Fig. 4 to see a simple cycle that is not a circle).

Definition 5 (simple cycle and circle). *A graph G is a* simple cycle *if the size of a* MAECD*, $c(G) = 1$. If in addition to that $deg^b(G,v) = deg^g(G,v) = 1$ for every vertex v, then G is called a* circle*.*

3.1 Parsimonious 2-break Scenarios

The problem of finding a minimum length (or *parsimonious*) 2-break scenario was treated in several unrelated settings using different terminology. Lemma 1, proven in the appendix, was treated in [6] where the authors also showed that finding a minimum length 2-break scenario is NP-hard due to the NP-hardness of finding a MAECD of a graph. A variant of the problem for Eulerian digraphs where all the gray edges are loops was solved in [8].

Lemma 1 (Bienstock et al. in [6]). *The minimum length of a 2-break scenario on a graph G is $d_{2b}(G) = e(G) - c(G)$.*

Since finding a MAECD for a breakpoint graph is easy, Lemma 1 leads to a linear time algorithm for finding a parsimonious DCJ scenario [25]. The algorithm is based on Lemma 2, proven in the appendix.

Lemma 2 (Yancopoulos et al. in [25]). *The minimum length of a DCJ scenario transforming genome A into B is equal to $d_{2b}(G(A,B)) = e(G(A,B)) - c(G(A,B))$.*

3.2 Decomposition of a 2-break Scenario

In this section we will show how a 2-break scenario ρ of length m can be partitioned into subscenarios $\rho^1, \ldots, \rho^k$ and G can be decomposed into edge-disjoint Eulerian subgraphs $H^1, \ldots, H^k$ where ρ^i is a scenario for H^i, and $k \geq e(G) - m$. We will use this decomposition in Sect. 5 to show that φ-MCPS on a graph can be solved by solving φ-MCPS on its simple cycles.

For a graph G and a 2-break scenario ρ we define a directed 1-edge-colored edge-labeled graph $\mathcal{D}(G, \rho)$, akin to the *trajectory graph* introduced by Shao, Lin, and Moret [21]. Denote the sequence of the first l 2-breaks of ρ by ρ_l and the graph obtained from G after these 2-breaks by G_l. Define $\mathcal{D}(G, \rho_0)$ in the following way: for each black edge e of G we have two new vertices connected by a directed edge labeled by e (see Fig. 3). For the l-th 2-break of ρ, $\{x_1, x_2\}, \{x_3, x_4\} \to \{x_1, x_3\}, \{x_2, x_4\}$, merge the endpoints of the edges labeled $\{x_1, x_2\}$ and $\{x_3, x_4\}$ in $\mathcal{D}(G, \rho_{l-1})$. Proceed by adding two new vertices to $\mathcal{D}(G, \rho_{l-1})$ and two edges labeled $\{x_1, x_3\}$ and $\{x_2, x_4\}$ from the merged vertex to the newly added ones to obtain $\mathcal{D}(G, \rho_l)$. Continue until $\mathcal{D}(G, \rho_m)$ is obtained, where m is the length of ρ, and denote it by $\mathcal{D}(G, \rho)$.

Shao, Lin, and Moret [21] characterize the connected components of a trajectory graph for a parsimonious scenario. Using similar techniques we prove the following theorem in the appendix.

Theorem 1. *If $\mathcal{D}(G, \rho)$ has k connected components then ρ can be partitioned into k subscenarios ρ^i and G can be partitioned into k edge-disjoint Eulerian subgraphs H^i in such a way that ρ^i is a scenario for H^i for every $i \in \{1, \ldots, k\}$. If ρ is parsimonious, then $k = c(G)$ and $C(\rho) = \{H^1, \ldots, H^k\}$ is a* MAECD *of G.*

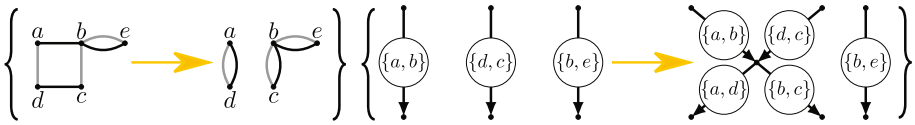

Fig. 3. A 2-break $\{a, b\}, \{d, c\} \rightarrow \{a, d\}, \{b, c\}$ transforming a graph G into a terminal one is depicted on the left. A directed graph $\mathcal{D}(G, \rho)$ is obtained from $\mathcal{D}(G, \rho_0)$ on the right for this scenario ρ of length 1. The endpoints of the edges labeled $\{a, b\}$ and $\{d, c\}$ are merged and two new edges labeled $\{a, d\}$ and $\{b, c\}$ are introduced. $\mathcal{D}(G, \rho)$ has 2 connected components that correspond to the 2 simple cycles of G.

4 Labeled 2-breaks and Their Costs

In this section we outline our model for assigning costs to 2-breaks on a graph G. We associate labels to edges, and then describe a set of valid operations $\mathcal{O}$ where each operation may transform the connectivity of G, the labeling of G, or both. Our cost function is defined on $\mathcal{O}$. This model is general enough to treat the edge labeled DCJ problems of [12] and [23].

For a set of vertices V and a set of labels $\mathcal{L}$ a *labeled edge* is an unordered pair of vertices plus a label, denoted $(\{a, b\}, x)$ for $a, b \in V$ and $x \in \mathcal{L}$. A *label change* $(\{a, b\}, x) \rightarrow (\{a, b\}, y)$ changes the label of an edge. A *labeled 2-break* $(\{a, b\}, x), (\{c, d\}, y) \rightarrow (\{a, c\}, z), (\{b, d\}, t)$ is a 2-break that replaces two labeled edges. Take a set $\mathcal{O}$ containing labeled 2-breaks and label changes, and a graph G with a labeling of its edges $\lambda : E \rightarrow \mathcal{L}$. An *$\mathcal{O}$-scenario* $\rho_{\mathcal{O}}$ for (G, λ), is a sequence of operations in $\mathcal{O}$ transforming (G, λ) into $(\bar{G}, \bar{\lambda})$ such that $\bar{G}$ is terminal, and the multi-sets of black and gray labeled edges of $\bar{G}$ are equal. The number of 2-breaks in $\rho_{\mathcal{O}}$ will be called the *2-break-length* of the scenario. If a $\rho_{\mathcal{O}}$ exists for (G, λ), then $d_{2b\mathcal{O}}(G, \lambda)$ denotes the minimum 2-break-length of an $\mathcal{O}$-scenario.

An $\mathcal{O}$-scenario does not necessarily exist for a given (G, λ), however if it exists, then the inequality $d_{2b\mathcal{O}}(G, \lambda) \geq d_{2b}(G)$ holds, where $d_{2b}(G)$ is the minimum length of a 2-break scenario on a graph G. In this paper we deal with the sets $\mathcal{O}$ that have the necessary operations to parsimoniously transform (G, λ) into $(\bar{G}, \bar{\lambda})$.

Definition 6 (p-sufficient $\mathcal{O}$ for (G, λ)). *A set $\mathcal{O}$ is* parsimonious-sufficient *or* p-sufficient *for (G, λ) if we have $d_{2b\mathcal{O}}(G, \lambda) = d_{2b}(G)$.*

The cost function that we consider is $\varphi : \mathcal{O} \rightarrow \mathbb{R}_+$. The cost of an $\mathcal{O}$-scenario is the sum of the costs of its constituent operations. If $\mathcal{O}$ is p-sufficient for (G, λ), then $\text{MCPS}_{\varphi}(G, \lambda)$ is the minimum cost of an $\mathcal{O}$-scenario of the 2-break-length equal to $d_{2b}(G)$, otherwise $\text{MCPS}_{\varphi}(G, \lambda)$ is ∞. We consider the following problem:

*Problem 1 (φ-*Minimum Cost Parsimonious Scenario *or φ-*MCPS*).*

INPUT : A graph G, and a labeling of its edges λ.
OUTPUT : $\text{MCPS}_{\varphi}(G, \lambda)$.

4.1 Examples of the Weighted DCJ Problems in the Literature

*Example 1 (*Minimum Local Parsimonious Scenario*).* In [23] we suppose the adjacencies of genome A to be partitioned into spatial regions represented by different colors. We then develop a polynomial time algorithm for finding a parsimonious DCJ scenario minimizing the number of rearrangements whose breakpoints appear in different regions. The problem, as it is stated in [23], differs slightly from φ-MCPS as in that study we do not have colors for the adjacencies of genome B. However, we can bridge this gap as follows.

Take a set of labels $\mathcal{L} = \mathcal{L}_c \cup \{\tau\}$ consisting of the colors $\mathcal{L}_c$ representing the different spatial regions of a genome and an additional terminal label τ. Define $\mathcal{O}$ as containing the labeled 2-breaks $(\{a,b\},x),(\{c,d\},y) \to (\{a,c\},x),(\{b,d\},y)$ for $a,b,c,d \in V$ and $x,y \in \mathcal{L}_c$, and a label change $(\{a,b\},x) \to (\{a,b\},\tau)$ for $a,b \in V$ and $x \in \mathcal{L}_c$. The cost φ_c of a labeled 2-break in $\mathcal{O}$ is 0 if the labels of the edges being replaced are equal and 1 otherwise. The cost of a label change is 0.

In [23] we presented an $O(n^4)$ time algorithm solving φ_c-MCPS for a labeled breakpoint graph with the gray edges labeled by τ. In [22] we demonstrated that finding a minimum cost $\mathcal{O}$ scenario for such a breakpoint graph, when the parsimonious criteria is disregarded, is NP-hard, and proposed an algorithm that is exponential in the number of colors but not in the number of genes.

In Sect. 9 we use the same $\mathcal{O}$. We fix a symmetric function $\Phi : \mathcal{L}^2 \to \mathbb{R}_+$ and define $\varphi_f((\{a,b\},x),(\{c,d\},y) \to (\{a,c\},x),(\{b,d\},y)) = \Phi(x,y)$. This drastically enriches the model introduced in [23]. In Sect. 7 we provide an $O(n^5)$ time algorithm solving φ_f-MCPS for a labeled breakpoint graph.

*Example 2 (*DCJ weighted by Hi-C*).* In [19] we weight each DCJ by the value taken directly from the Hi-C contact map. In this model every intergenic region of genome A gets assigned an interval corresponding to its genomic coordinates on a chromosome. The *weight* of a DCJ acting on two intergenic regions is then equal to the average Hi-C value for their corresponding intervals. In [19] we provide an algorithm greedily maximizing the weight of a parsimonious scenario and find that the obtained weight is significantly higher than the weight of a random parsimonious scenario.

Take a set of labels consisting of the genomic intervals corresponding to the intergenic regions of a genome A plus an additional terminal label. Keep the same $\mathcal{O}$ as in Example 1. Define $\Phi_{HiC}(x,y)$ on two genomic intervals to be their average Hi-C value. The problem that maximizes Hi-C values can be easily transformed into a minimization problem by setting the cost of a labeled 2-break $(\{a,b\},x),(\{c,d\},y) \to (\{a,c\},x),(\{b,d\},y)$ to $\Phi_{\max} - \Phi_{HiC}(x,y)$, where $\Phi_{\max}$ is the maximum $\Phi_{HiC}(x,y)$ over all x,y.

In [19] the optimality of the proposed greedy algorithm is not discussed, but our work presented in Sect. 9 of this paper provides us with a polynomial time algorithm for solving this problem exactly.

*Example 3 (*Sorting by wDCJs and indels in intergenes*).* Bulteau, Fertin, and Tannier [12] introduce a problem where adjacencies of genomes are

labeled with their genetic length (number of nucleotides). A *wDCJ* is a labeled DCJ that preserves the sum of the genetic lengths of the adjacencies and an *indel* δ is a label change that increases or decreases the genetic length of an adjacency by δ. The cost of a wDCJ is 0 and the cost of an indel δ is $|\delta|$. A scenario of wDCJs and indels for (G, λ) is said to be *valid* if its wDCJ-length is $d_{2b}(G)$. The paper presents an $O(n \log n)$ algorithm for finding a minimum cost scenario among the *valid* ones, for the genomes with circular chromosomes and n genes.

Translating this into our formalism yields the following φ-MCPS problem. The labels $\mathcal{L}$ would be the natural numbers, while $\mathcal{O}$ contains labeled 2-breaks $(\{a, b\}, w_1), (\{c, d\}, w_2) \rightarrow (\{a, c\}, w_3), (\{b, d\}, w_4)$ for every $a, b, c, d \in V$, and $w_i \in \mathcal{L}$ satisfies $w_1 + w_2 = w_3 + w_4$. $\mathcal{O}$ also contains label changes $(\{a, b\}, w_1) \rightarrow (\{a, b\}, w_2)$ for every $a, b \in V$ and $w_i \in \mathcal{L}$. $\mathcal{O}$ is p-sufficient for any (G, λ) since G can be first transformed into a terminal graph using any parsimonious 2-break scenario and then its labels can be adjusted. The cost φ_l of a labeled 2-break is 0 and the cost φ_l of a label change $(\{a, b\}, w_1) \rightarrow (\{a, b\}, w_2)$ is $|w_1 - w_2|$.

In [12] the authors present an $O(r \log r)$ time algorithm for solving φ_l-MCPS on a circle with r vertices. Combining this algorithm with our results from Sect. 7 gives an algorithm solving φ_l-MCPS in $O(n^3)$ time for a labeled breakpoint graph. The ILP defined in Sect. 5 solves φ_l-MCPS for any labeled graph.

*Example 4 (*wDCJ-dist*).* Fertin, Jean, and Tannier [15] treat a problem wDCJ-dist where wDCJs without indels are allowed, and the sums of the genetic lengths of the adjacencies of two genomes are equal.

In this case we keep the same $\mathcal{L}$ and $\mathcal{O}$ as in Example 3 except that the label changes are excluded from $\mathcal{O}$. A labeled graph is said to be *balanced* if the sums of the labels of black and gray edges are equal. wDCJ-dist is the problem of finding $d_{2b\mathcal{O}}$ for a balanced graph whose connected components are circles. The authors show that wDCJ-dist is strongly NP-complete. However they also prove that $d_{2b\mathcal{O}}(O, \lambda) = d_{2b}(O)$ for a balanced circle and that $\mathcal{O}$ is p-sufficient for a graph whose connected components are balanced circles.

Example 5. Although ignored in the previous examples, the weighting of operations based on the vertices is also possible under our framework. For example, take $\mathcal{L} = \{\tau\}$, $\mathcal{O}$ containing labeled 2-breaks $(\{a, b\}, \tau), (\{c, d\}, \tau) \rightarrow (\{a, c\}, \tau), (\{b, d\}, \tau)$ and any cost function $\varphi_v : \mathcal{O} \rightarrow \mathbb{R}_+$. The costs of the labeled 2-breaks in $\mathcal{O}$ could be a function of the genomic coordinates of the participating gene extremities.

Note that the set $\mathcal{O}$ is implicit, rather than explicit. In Example 3, $\mathcal{O}$ would be too large to represent explicitly since every pair of genetic lengths for every pair of edges would exist.

5 φ-MCPS for a Graph

Theorem 2. *Denote the* φ-cost *of a* MAECD *as the sum of the* MCPS_φ *on its cycles.* MCPS_φ *for a graph is equal to the minimum* φ*-cost of its* MAECD.

Proof. For a cycle S of a labeled graph (G, λ), λ^S denotes the labeling of the edges of S according to λ. We suppose that $min(\emptyset) = \infty$ and prove the following:

$$\mathrm{MCPS}_\varphi(G, \lambda) = min\Big\{ \sum_{S \in C} \mathrm{MCPS}_\varphi(S, \lambda^S) \;\Big|\; C \text{ is a MAECD of } G \Big\}.$$

Suppose that there exists a MAECD C of G consisting of the simple cycles for which $\mathcal{O}$ is p-sufficient. For every $S \in C$ take an $\mathcal{O}$-scenario $\rho^S_\mathcal{O}$ of cost $\mathrm{MCPS}_\varphi(S, \lambda^S)$ and 2-break-length $d_{2b}(S)$. By performing these scenarios one after another we obtain an $\mathcal{O}$-scenario $\rho_\mathcal{O}$ for (G, λ) of 2-break-length $\sum_{S \in C} d_{2b}(S) = d_{2b}(G)$ and of cost $\sum_{S \in C} \mathrm{MCPS}_\varphi(S, \lambda^S)$. This yields a scenario such that $\mathrm{MCPS}_\varphi(G, \lambda) \leq \sum_{S \in C} \mathrm{MCPS}_\varphi(S, \lambda^S)$.

On the other hand, suppose that $\mathcal{O}$ is p-sufficient for (G, λ) and take an $\mathcal{O}$-scenario $\rho_\mathcal{O}$ for (G, λ) of length $d_{2b}(G)$. For ρ, a 2-break scenario obtained from $\rho_\mathcal{O}$ when the labels of the edges are neglected, a decomposition $C(\rho)$ corresponding to ρ is a MAECD of G due to Theorem 1. A subsequence $\rho^S_\mathcal{O}$ of $\rho_\mathcal{O}$, consisting of the operations acting on the edges of a cycle $S \in C(\rho)$, is an $\mathcal{O}$-scenario for (S, λ^S) of 2-break-length $d_{2b}(S)$. A sequence of operations $\hat{\rho}_\mathcal{O}$ obtained by performing the subsequences $\rho^S_\mathcal{O}$ one after another for each $S \in C(\rho)$ is an $\mathcal{O}$-scenario for (G, λ). By construction the 2-break-length of $\hat{\rho}_\mathcal{O}$ is equal to the 2-break-length of $\rho_\mathcal{O}$. The costs of $\rho_\mathcal{O}$ and $\hat{\rho}_\mathcal{O}$ are also equal, as they consist of exactly the same operations that are performed in different orders, thus the cost of $\rho_\mathcal{O}$ is greater or equal to $\sum_{S \in C(\rho)} \mathrm{MCPS}_\varphi(S, \lambda^S) \geq min\Big\{ \sum_{S \in C} \mathrm{MCPS}_\varphi(S, \lambda^S) \;\Big|\; C \text{ is a MAECD of } G \Big\}$. □

Take the set $\mathcal{S}$ of simple labeled cycles of (G, λ). If one can solve φ-MCPS for every $S \in \mathcal{S}$, then Theorem 2 provides a straightforward way to solve φ-MCPS for (G, λ) as a set packing problem. First compute $c(G)$ by solving the ILP in the left column. Then proceed by solving the other ILP to compute $\mathrm{MCPS}_\varphi(G, \lambda)$.

$$\begin{array}{l} \text{Maximize } \sum_{S \in \mathcal{S}} x_S \\ \text{Subject to } \sum_{S : e \in S} x_S \leq 1 \text{ for each edge } e \text{ of } G \\ \quad \text{and } x_S \in \{0, 1\} \text{ for simple cycle } S \in \mathcal{S}. \end{array} \qquad \begin{array}{l} \text{Minimize } \sum_{S \in \mathcal{S}} x_S \mathrm{MCPS}_\varphi(S, \lambda^S) \\ \text{Subject to } \sum_{S : e \in S} x_S \leq 1 \text{ for each edge } e \text{ of } G, \\ \quad \sum_{S \in \mathcal{S}} x_S = c(G) \\ \quad \text{and } x_S \in \{0, 1\} \text{ for simple cycle } S \in \mathcal{S}. \end{array}$$

The size of $\mathcal{S}$ may be exponential in the size of G, which might make these ILPs intractable in general. For graphs representing genomes with duplicate genes, the number of simple cycles can grow exponentially as a function of the number of duplicate genes. For breakpoint graphs, the number grows quadratically.

6 φ-MCPS for a Simple Cycle

The decomposition theorem of Sect. 5 reduces the computation of φ-MCPS on a graph to the computation of φ-MCPS on a simple alternating cycle. In this

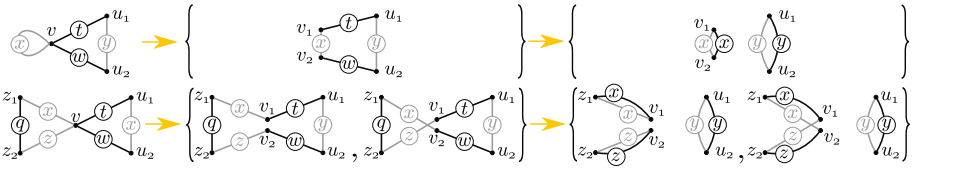

Fig. 4. Two simple cycles having a vertex v of degree two are depicted in the first column. Their sets of the corresponding circles obtained by splitting v into v_1 and v_2 are depicted in the second column. This set is of size 1 for the upper simple cycle containing the gray loop $\{v, v\}$, and of size 2 for the lower simple cycle. An $\mathcal{O}$-scenario for a simple cycle provides a scenario of the same cost and length transforming the graphs in the second column to the ones that become terminal once v_1 and v_2 are merged. One possible outcome of such a scenario is presented in the third column.

section we further decompose the problem into simpler versions of cycles, called circles, which are alternating cycles that contain a vertex only once.

Denote $deg_2(G)$ for a graph G as the number of vertices with black and gray degree equal to two. It is easy to check that $deg^b(S, v) = deg^g(S, v) \leq 2$ for any vertex v of a simple cycle S. If $deg_2(S) = 0$, then S is a circle. See the first column of Fig. 4 for examples of simple cycles that are not circles.

Take a simple cycle S on vertices V, a labeling of its edges λ and denote S_0 as $\{(S, \lambda)\}$. Choose a vertex of degree two in S. If it is incident to a gray loop, then split it into two vertices, as depicted on the top row of Fig. 4, to obtain a set S_1 consisting of a single simple cycle. Otherwise split it into two vertices, as depicted on the bottom row of Fig. 4, to obtain a set S_1 consisting of two simple cycles. The simple cycles in S_1, by construction, share the same set of vertices, that we denote $\hat{V}$, and the same multi-set of labeled black edges. $\mathcal{O}$ and a cost function φ defined for vertices V can be extended in a natural way to $\hat{\mathcal{O}}$ and $\hat{\varphi}$ defined for vertices $\hat{V}$. For example if a vertex v was split into v_1 and v_2, then $\hat{\varphi}((\{v_1, u\}, x) \to (\{v_1, u\}, y)) = \varphi((\{v, u\}, x) \to (\{v, u\}, y))$ for $u \in V \cap \hat{V}$ and labels x, y. In the appendix we prove the following lemma.

Lemma 3. $\mathrm{MCPS}_{\varphi}(S, \lambda) = min\{\mathrm{MCPS}_{\hat{\varphi}}(\hat{S}, \hat{\lambda}) |\ (\hat{S}, \hat{\lambda}) \in\ S_1\}$

Simple cycles in S_1 share the same set of vertices of degree two. Choose such a vertex and split it simultaneously in all the cycles in S_1 as previously to obtain a set S_2 of at most 4 simple cycles sharing the same set of vertices and the same multi-set of labeled black edges. Continue this procedure until the set $circ(S, \lambda) = S_{deg_2(S)}$ of the labeled circles is obtained. We denote $\overline{V}$ as the set of vertices of these circles. $\mathcal{O}$ and a cost function φ defined for vertices V can be extended in a natural way to $\overline{\mathcal{O}}$ and $\overline{\varphi}$ defined for vertices $\overline{V}$.

Theorem 3. MCPS_{φ} *for a simple cycle* (S, λ) *is equal to the minimum of the* MCPS_{φ} *among the circles in* $circ(S, \lambda)$.

Proof. We prove $\mathrm{MCPS}_{\varphi}(S, \lambda) = min\{\mathrm{MCPS}_{\bar{\varphi}}(O, \lambda^O) |\ (O, \lambda^O) \in\ circ(S, \lambda)\}$, which is clearly true for $deg_2(S) = 0$. We suppose it to be true for $deg_2(S) < t$

and prove it for $deg_2(S) = t$ by induction. By Lemma 3 we get $\text{MCPS}_\varphi(S, \lambda) = min\{\text{MCPS}_{\hat{\varphi}}(\hat{S}, \hat{\lambda}) | \ (\hat{S}, \hat{\lambda}) \in \ S_1\}$. Since, for a simple cycle $(\hat{S}, \hat{\lambda}) \in S_1$ we have $deg_2(\hat{S}) = t - 1$, we use the inductive hypothesis to obtain $\text{MCPS}_{\hat{\varphi}}(\hat{S}, \hat{\lambda}) = min\{\text{MCPS}_{\bar{\varphi}}(O, \lambda^O) | \ (O, \lambda^O) \in \ circ(\hat{S}, \hat{\lambda})\}$. Further, we know that $circ(S, \lambda) = \cup_{(\hat{S},\hat{\lambda})\in S_1} circ(\hat{S}, \hat{\lambda})$ by construction. Combining these results we obtain that the theorem is true for $deg_2(S) = t$. □

7 φ-MCPS for a Breakpoint Graph

In this section we suppose that there exists an algorithm for computing MCPS_φ on a labeled circle (*e.g.* the algorithm of Sect. 9). Using this algorithm as a subroutine we will construct an algorithm for finding MCPS_φ for a labeled breakpoint graph. This is a generalization of the work first presented in [23].

Take genomes A and B partitioned into n genes where each gene occurs exactly once in each genome, and a labeling λ of the edges of $G(A, B)$. For all the vertices $v \neq \circ$ we have $deg^g(G(A, B), v) = deg^b(G(A, B), v) = 1$. Thus, if there is a circle in $G(A, B)$ containing an edge then this circle is the only simple cycle containing this edge. This means that every MAECD of $G(A, B)$ includes all of its circles. These set aside we are left with $G(A, B)'$, which is a union of alternating paths starting and ending at $\circ$ with end edges of the same color. If this color is black we call the path AA, and BB otherwise.

We proceed by constructing a complete weighted bipartite graph H having the AA and BB paths of $G(A, B)'$ as vertices. Every simple cycle of $G(A, B)'$ is a union of an AA path and a BB path. An edge joining these paths in H will have the weight equal to MCPS_φ for a union of these paths. A MAECD of $G(A, B)'$ provides us with a maximum matching of H and every such matching provides a MAECD of $G(A, B)'$. Denote λ' as the labeling of the edges of $G(A, B)'$ according to λ. Using Theorem 2 we obtain that $\text{MCPS}_\varphi(G(A, B)', \lambda')$ is equal to the minimum weight of a maximum matching of H. There is an equal number p of AA and BB paths. Let P denote the total number of edges in $G(A, B)'$. Using this notation we obtain the following lemma proven in the appendix.

Lemma 4. *For a function f and an $O(f(r))$ time algorithm for φ-*MCPS *on a labeled circle on r vertices, there exists an $O(p^2 f(P) + p^3 + f(n))$ time algorithm for φ-*MCPS *on a labeled breakpoint graph. If $f(r) = O(r^t)$ for some constant $t \geq 1$, then φ-*MCPS *on a labeled breakpoint graph can be solved in $O(pP^t + p^3 + n^t)$ time.*

Both p and P are $O(n)$, thus Lemma 4 leads to the following theorem.

Theorem 4. *Given a constant $t \geq 2$ and an $O(r^t)$ time algorithm for φ-*MCPS *on a labeled circle on r vertices, φ-*MCPS *on a labeled breakpoint graph can be solved in $O(n^{t+1})$ time.*

Corollary 1. *Using the $O(r^4)$ algorithm from Sect. 9 we obtain an $O(n^5)$ algorithm for solving φ_f-*MCPS *on a labeled breakpoint graph with fixed labels.*

Corollary 2. *Using the $O(r \log r)$ algorithm from [12] for the* Sorting by wDCJs and indels in intergenes *problem on a circle (see Example 3), we obtain an $O(n^3)$ algorithm for solving the problem on a breakpoint graph.*

8 α-approximation for φ-MCPS

Theorems 2 and 3 demonstrate how φ-MCPS for any labeled graph can be solved if one is able to solve φ-MCPS for a labeled circle. This is exploited in Theorem 4 to solve φ-MCPS for a breakpoint graph. Analogous results hold if instead of an exact algorithm one has an α-approximation for φ-MCPS for a labeled circle. This is illustrated with the following theorem proven in the appendix.

Theorem 5. *For a constant $t \geq 2$ and an $O(r^t)$ time α-approximation algorithm for* φ-MCPS *on a labeled circle on r vertices, there exists an $O(n^{t+1})$ time α-approximation algorithm for* φ-MCPS *on a labeled breakpoint graph.*

9 φ_f-MCPS for a Circle with Fixed Labels

Here we define φ_f-MCPS, a particular instance of a φ-MCPS problem, and solve it for a circle. φ_f-MCPS generalizes our previous work presented in Example 1 and 2.

For a set V of vertices and a set $\mathcal{L} \cup \{\tau\}$ of labels, define a set $\mathcal{O}$ consisting of labeled 2-breaks $(\{a,b\},x),(\{c,d\},y) \to (\{a,c\},x),(\{b,d\},y)$ for $a,b,c,d \in V$ and $x,y \in \mathcal{L}$, and label changes $(\{a,b\},x) \to (\{a,b\},\tau)$ for $a,b \in V$ and $x \in \mathcal{L}$. Fix a symmetric function $\Phi : \mathcal{L}^2 \to \mathbb{R}_+$ and define a cost function $\varphi_f((\{a,b\},x),(\{c,d\},y) \to (\{a,c\},x),(\{b,d\},y)) = \Phi(x,y)$ and $\varphi_f((\{a,b\},x) \to (\{a,b\},\tau)) = 0$.

We will provide a polynomial time algorithm for φ_f-MCPS on a labeled circle with the gray edges labeled by a terminal label τ. Without loss of generality we can suppose that all of the black edges of a circle have different labels; if two edges are labeled with the same label x, then we simply replace one of these labels with a new label $\hat{x}$ and set $\hat{\Phi}(\hat{x},y) = \Phi(x,y)$ and $\hat{\Phi}(y,z) = \Phi(y,z)$ for $y,z \in \mathcal{L}$.

For a labeled circle having r black edges, define a set $V_{\mathcal{L}}$ of r vertices corresponding to their labels. For an $\mathcal{O}$-scenario $\rho_{\mathcal{O}}$ we define a 1-edge-colored undirected graph $\mathcal{T}(\rho_{\mathcal{O}})$ with vertices $V_{\mathcal{L}}$ and an edge $\{x,y\}$ for every labeled 2-break in $\rho_{\mathcal{O}}$ replacing the edges labeled with x and y (See Fig. 5). The *cost* of an edge $\{x,y\}$ is defined to be $\Phi(x,y)$ and the cost of a $\mathcal{T}(\rho_{\mathcal{O}})$ is the sum of the costs of its edges. The costs of $\rho_{\mathcal{O}}$ and $\mathcal{T}(\rho_{\mathcal{O}})$ are equal by construction.

Fix a circular embedding of $V_{\mathcal{L}}$ respecting the order of the black edges on the labeled circle (See Fig. 5). A graph with vertices $V_{\mathcal{L}}$ is said to be *planar on the circle* if none of its edges cross in this embedding. In the appendix we prove Lemma 5 linking planar trees and parsimonious scenarios.

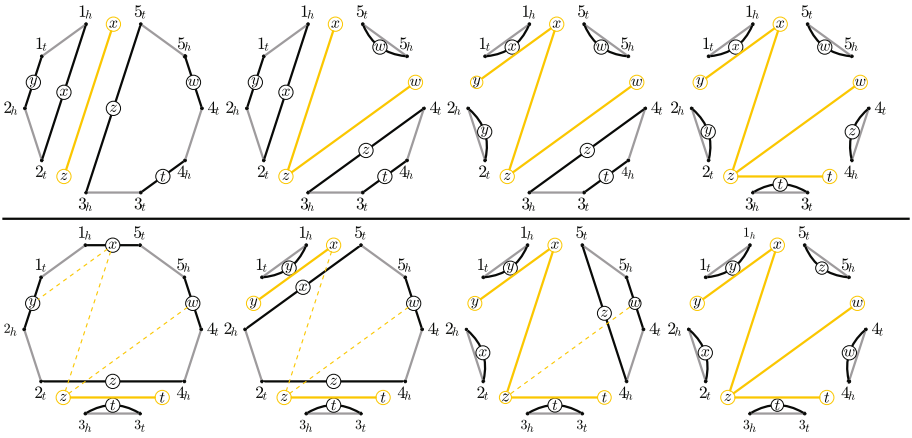

Fig. 5. On the top: 4 steps of a parsimonious $\mathcal{O}$-scenario for a circle are depicted together with each $\mathcal{T}$ corresponding to the scenario at that point colored in yellow. Vertices of $\mathcal{T}$ are superimposed on the corresponding edges of a circle providing their circular embedding. All of the $\mathcal{T}$ are planar trees. **On the bottom:** For a given planar tree $\mathcal{T}$ (dashed yellow) we provide a scenario $\rho_{\mathcal{O}}$ such that $\mathcal{T}(\rho_{\mathcal{O}}) = \mathcal{T}$.

Lemma 5. *If $\rho_{\mathcal{O}}$ is a minimum 2-break-length $\mathcal{O}$-scenario for a labeled circle (O, λ), then $\mathcal{T}(\rho_{\mathcal{O}})$ is a planar tree on (O, λ). In addition to that, for a planar tree $\mathcal{T}$ on (O, λ) there exists an $\mathcal{O}$-scenario $\rho_{\mathcal{O}}$ such that $\mathcal{T}(\rho_{\mathcal{O}}) = \mathcal{T}$.*

Farnoud and Milenkovic in [14] pose the problem of sorting permutations by cost-constrained mathematical transpositions and provide a dynamic programming algorithm for finding a minimum cost planar tree on a circle. In the appendix we provide their proof for a following lemma which, together with Lemma 6, leads to Theorem 6.

Lemma 6 (Farnoud et al. in [14]). *A minimum cost planar tree on a circle can be found in $O(r^4)$ time, where r is the number of vertices of a tree.*

Theorem 6. *φ_f-*MCPS *for a labeled circle on r vertices can be solved in $O(r^4)$ time.*

10 Conclusions and Future Directions

10.1 Practical Matters

Our algorithm for φ_f-MCPS on a breakpoint graph with fixed labels has a running time of $O(n^5)$ in the worst case. Note that the running time is dominated, however, by the maximum bipartite matching step in Sect. 7. The size of this graph is determined by the number of AA paths which is bounded by the number of chromosomes, so in practice it can be treated as a constant. Thus, the

algorithm scales like $O(n^4)$ on real data. Further, since n is the number of syntenic blocks — and not literally the genes as we call them — there are few blocks. Our analyses of *Drosophila* genomes yield no AA paths, and less than 100 blocks [19]. Although about 13,000 blocks between human and mouse are reported in the files associated to Baudet et *al.*, many of them can be merged because they are co-linear in the two species [3]. The effective number of blocks for this pair is closer to 600.

For graphs with higher degree nodes, like those graphs that represent genomes with duplicated genes, the number of simple cycles can grow rapidly. Although this relationship is not immediately evident, we expect that fixed parameter algorithms could be developed to handle biological data in the future.

10.2 Future Direction

Our cost framework is liberal, and in our examples we have explored only a small portion of its capacities. Edges can be labeled by complex objects such as vectors or trees that encode the biological information of extant genomes and its modification throughout a scenario. Costs can be a function of a combination of the edge labels and vertices. We hope that a closer study of the graph $\mathcal{D}(G, \rho)$ from Sect. 3.2 will lead to polynomial time algorithms for φ-MCPS on circles for a large family of problems.

While all of our results apply to genomes with insertions or deletions of single genes, further study is required in order to increase efficiency on genomes with duplicate genes. Other improvements to our work could consider non-parsimonious 2-break scenarios.

Acknowledgments. This work is partially supported by the IBC (Institut de Biologie Computationnelle) (ANR-11-BINF-0002), by the Labex NUMEV flagship project GEM, and by the CNRS project Osez l'Interdisciplinarité.

A Proofs

A.1 Lemma 1

Lemma. *The minimum length of a 2-break scenario on a graph G is $d_{2b}(G) = e(G) - c(G)$.*

Proof. A 2-break can increase the size of a MAECD by at most 1 and the size of a MAECD of a terminal graph is $e(G)$. This leads to an inequality $d_{2b}(G) \geq e(G) - c(G)$.

In this paragraph the *length* of a cycle will be its number of black edges. For any cycle c of length $l > 1$ there is a 2-break transforming c into a union of length 1 and length $l-1$ cycles. This way we obtain a scenario of length $l-1$ for c, and can transform every cycle of a MAECD of G independently, obtaining a 2-break scenario of length $e(G) - c(G)$. Thus, $d_{2b}(G) \leq e(G) - c(G)$. □

A.2 Lemma 2

Lemma. *The minimum length of a DCJ scenario transforming genome A into B is equal to $d_{2b}(G(A,B)) = e(G(A,B)) - c(G(A,B))$.*

Proof. $G(A,B)$ is constructed in such a way that for every DCJ $A \to A'$ the transformation $G(A,B) \to G(A',B)$ is a 2-break. Notably, a DCJ $\{a,b\} \to \{a\},\{b\}$ results in a transformation $\{a,b\},\{\circ,\circ\} \to \{a,\circ\},\{b,\circ\}$, as the construction of a breakpoint graph guarantees that there are enough black loops $\{\circ,\circ\}$ to realize such a 2-break. For any 2-break $G(A,B) \to G'$ with $G' \neq G(A,B)$ there exists a DCJ $A \to A'$ such that $G(A',B) = G'$. Since $G(B,B)$ is terminal, it follows that the minimum length of a scenario transforming A into B is $d_{2b}(G(A,B))$ and we conclude using Lemma 1. □

A.3 Theorem 1

Theorem. *If $\mathcal{D}(G,\rho)$ has k connected components then ρ can be partitioned into k subscenarios ρ^i and G can be partitioned into k edge-disjoint Eulerian subgraphs H^i in such a way that ρ^i is a scenario for H^i for every $i \in \{1,\ldots,k\}$. If ρ is parsimonious, then $k = c(G)$ and $C(\rho) = \{H^1,\ldots,H^k\}$ is a* MAECD *of G.*

Proof. Take a connected component C of $\mathcal{D}(G,\rho)$. It has an equal number of vertices of indegree 0 and vertices of outdegree 0. Its edges incident to the vertices of indegree 0 are labeled with the black edges of G and its edges incident to the vertices of outdegree 0 are labeled with the gray edges of G. Together these labels define a subgraph H of G that we will prove to be Eulerian.

Define C_l to be a subgraph of $\mathcal{D}(G,\rho_l)$ consisting of its connected components containing the vertices of indegree 0 of C. This way $C_m = C$. Define H_l to be a subgraph of G_l containing the gray edges of H and the black edges of G_l labeling the edges of C_l incident to the vertices of outdegree 0. This way $H_0 = H$ and H_m is a terminal graph.

We prove that H is Eulerian by induction. H_m is Eulerian as it is terminal. Suppose that H_l is Eulerian. By construction the two edges of G_l replaced by the l-th 2-break of ρ either both belong to H_{l-1} or both are outside of H_{l-1}. In the first case, H_l is obtained from H_{l-1} via a 2-break and as H_l is Eulerian this means that H_{l-1} is also Eulerian. In the second case, $H_l = H_{l-1}$, thus the latter stays Eulerian. Thus $H = H_0$ is Eulerian and we obtain a subsequence of ρ that is a scenario for H.

$\mathcal{D}(G,\rho_0)$ has $e(G)$ connected components. The l-th 2-break of ρ merges two vertices of $\mathcal{D}(G,\rho_{l-1})$, thus reduces the number of the connected components by at most 1. This means that the number k of the connected components of $\mathcal{D}(G,\rho)$ is greater or equal to $e(G) - m$.

If ρ is parsimonious, then its length m is $e(G) - c(G)$ using Lemma 1. This means that $k \geq c(G)$ and G can be partitioned into k edge-disjoint Eulerian subgraphs. Due to the maximality of $c(G)$, we have that $k = c(G)$ and all of the obtained edge-disjoint Eulerian subgraphs of G are simple cycles. □

A.4 Lemma 3

Lemma. $\mathrm{MCPS}_{\varphi}(S,\lambda) = min\{\mathrm{MCPS}_{\hat{\varphi}}(\hat{S},\hat{\lambda}) |\ (\hat{S},\hat{\lambda}) \in\ S_1\}$

Proof. For a labeled graph (H,μ) on vertices $\hat{V}$ we denote $r(H,\mu)$ as the labeled graph obtained from (H,μ) by merging the two vertices that were split in S. For $(\hat{S},\hat{\lambda}) \in\ S_1$ we have $r(\hat{S},\hat{\lambda}) = (S,\lambda)$ by construction. An operation in $\hat{\mathcal{O}}$ transforms $(\hat{S},\hat{\lambda})$ into such $(\hat{S}',\hat{\lambda}')$ that there exists unique operation in $\mathcal{O}$ of the same cost transforming (S,λ) into $r(\hat{S}',\hat{\lambda}')$. This leads to an observation that for an $\hat{\mathcal{O}}$-scenario for $(\hat{S},\hat{\lambda})$ there exists an $\mathcal{O}$-scenario of the same cost and the same 2-break-length for (S,λ).

On the other hand, for an operation in $\mathcal{O}$ transforming (S,λ) into (S',λ') there exists an operation in $\hat{\mathcal{O}}$ of the same cost transforming every $(\hat{S},\hat{\lambda}) \in S_1$ into $(\hat{S}',\hat{\lambda}')$ such that $r(\hat{S}',\hat{\lambda}'_S) = (S',\lambda')$. This leads to an observation that an $\mathcal{O}$-scenario for (S,λ) provides us with a sequence $\hat{\rho}_{\hat{\mathcal{O}}}$ of $\hat{\mathcal{O}}$ operations of the same cost and 2-break-length transforming every $(\hat{S},\hat{\lambda}) \in S_1$ into such $(\overline{\hat{S}},\overline{\hat{\lambda}})$ for which $r(\overline{\hat{S}},\overline{\hat{\lambda}})$ is a terminal graph with equal multi-sets of labeled gray and black edges. As the later graph is obtained by merging two vertices of degree one of the former, we know that its structure is as well fairly simple. We can check all the possible cases by hand and show that there is $(\hat{S},\hat{\lambda}) \in S_1$ such that $(\overline{\hat{S}},\overline{\hat{\lambda}})$ is itself a terminal graph with equal multi-sets of labeled gray and black edges.

If S_1 is of size 1, then there is a single choice for $(\overline{\hat{S}},\overline{\hat{\lambda}})$ such that $r(\overline{\hat{S}},\overline{\hat{\lambda}})$ is a terminal graph with equal multi-sets of labeled gray and black edges (see the right upper corner of Fig. 4). If S_1 is of size 2, then there are more cases, but they are all easy to check and one of them is given in the right bottom corner of Fig. 4. □

A.5 Lemma 4

Lemma. *For a function f and an $O(f(r))$ time algorithm for φ-*MCPS *on a labeled circle on r vertices, there exists an $O(p^2 f(P) + p^3 + f(n))$ time algorithm for φ-*MCPS *on a labeled breakpoint graph. If $f(r) = O(r^t)$ for some constant $t \geq 1$, then φ-*MCPS *on a labeled breakpoint graph can be solved in $O(pP^t + p^3 + n^t)$ time.*

Proof. The p^2 edges of a bipartite graph H can be weighted in $O(p^2 f(P))$ time due to Theorem 3 and the fact that the simple cycles of $G(A,B)$ have at most 1 vertex of degree 2. A minimum weight maximum matching of H can be found in $O(p^3)$ time using the Hungarian algorithm. Finally, MCPS_{φ} for the labeled circles in $G(A,B)$ can be computed in $O(f(n))$ time. Combining these results we obtain an $O(p^2 f(P) + p^3 + f(n))$ time algorithm for computing $\mathrm{MCPS}_{\varphi}(G(A,B),\lambda)$.

Now suppose that $f(r) = O(r^t)$ for some constant $t \geq 1$. Let $a_1,\ldots,a_p$ and $b_1,\ldots,b_p$ denote the number of edges in AA and BB paths with $\sum_{i=0}^{p} a_i = P_A$, $\sum_{j=0}^{p} b_j = P_B$ and $P = P_A + P_B$.

MCPS_φ for a union of an AA path and a BB path having a and b edges respectively can be obtained by computing MCPS_φ for at most two circles on $a+b$ vertices due to Theorem 3. This can be done in less than $c(a+b)^t$ steps for some constant c using the $O(r^t)$ time algorithm for computing MCPS_φ for a circle. MCPS_φ for every pair of AA and BB paths of $G(A,B)'$ can be computed in a number of steps bounded by:

$$\sum_{i=0}^{p}\sum_{j=0}^{p} c(a_i+b_j)^t = c\sum_{i=0}^{p}\sum_{j=0}^{p}\sum_{l=0}^{t}\binom{t}{l}a_i^l b_j^{t-l} = c\sum_{l=0}^{t}\binom{t}{l}\sum_{i=.For0}^{p}\sum_{j=0}^{p} a_i^l b_j^{t-l}$$

$$= c\sum_{j=0}^{p}\sum_{i=0}^{p} b_j^t + c\sum_{i=0}^{p}\sum_{j=0}^{p} a_i^t + c\sum_{l=1}^{t-1}\binom{t}{l}\sum_{i=0}^{p} a_i^l \sum_{j=0}^{p} b_j^{t-l}$$

$$= cp\sum_{j=0}^{p} b_j^t + cp\sum_{i=0}^{p} a_i^t + c\sum_{l=1}^{t-1}\binom{t}{l}\sum_{i=0}^{p} a_i^l \sum_{j=0}^{p} b_j^{t-l}$$

$$\leq cp(\sum_{j=0}^{p} b_j)^t + cp(\sum_{i=0}^{p} a_i)^t + c\sum_{l=1}^{t-1}\binom{t}{l}(\sum_{i=0}^{p} a_i)^l(\sum_{j=0}^{p} b_j)^{t-l}$$

$$\leq c(pP_B^t + pP_A^t) + pc\sum_{l=1}^{t-1}\binom{t}{l}P_B^{t-l}P_A^l = cp(P_B+P_A)^t = cpP^t$$

Thus, the weighting of H can be performed in $O(pP^t)$ time. This provides us with an $O(pP^t + p^3 + n^t)$ time algorithm for computing $\text{MCPS}_\varphi(G(A,B),\lambda)$.□

A.6 Theorem 5

Theorem. *For a constant $t \geq 2$ and an $O(r^t)$ time α-approximation algorithm for φ-*MCPS *on a labeled circle on r vertices, there exists an $O(n^{t+1})$ time α-approximation algorithm for φ-*MCPS *on a labeled breakpoint graph.*

Proof. In Theorem 3, MCPS_φ on a simple cycle is expressed as the minimum of the MCPS_φ for a set of corresponding circles. In Theorem 2, MCPS_φ on a graph is expressed as the minimum of the sums of the MCPS_φ for the simple cycles. We prove an auxiliary lemma establishing the following:

1. An α-approximation for MCPS_φ on a simple cycle can be obtained by taking the minimum of the α-approximations for the corresponding circles.
2. An α-approximation for MCPS_φ on a graph can be obtained by taking the minimum of the sums of the α-approximations for MCPS_φ on the simple cycles.

Lemma. *Take $k \in \mathbb{N}$ and two sets of positive numbers $\{q_1^*,\ldots,q_k^*\}$ and $\{q_1,\ldots,q_k\}$ with $q_i \leq \alpha q_i^*$ for every i. The following inequalities hold:*

1. $min\{q_i | i \in \{1,\ldots,k\}\} \leq \alpha min\{q_i^* | i \in \{1,\ldots,k\}\}$
2. $\sum_{i=0}^{k} q_i \leq \alpha \sum_{i=0}^{k} q_i^*$

Proof. Take u and v such that $q_u^* = min\{q_i^* | i \in \{1, \ldots, k\}\}$ and $q_v = min\{q_i | i \in \{1, \ldots, k\}\}$. By construction $q_v \leq q_u \leq \alpha q_u^*$ which proves the first inequality. For the second inequality it suffice to observe that $\sum_{i=0}^{k} q_i \leq \sum_{i=0}^{k} \alpha q_i^* = \alpha \sum_{i=0}^{k} q_i^*$ □

A simple cycle of a breakpoint graph has at most one vertex of degree 2. This means that it has at most two corresponding circles (see Theorem 6). Taking the minimum of the α-approximations for MCPS_φ on these circles provides us with an α-approximation for the simple cycle due to Theorem 6 and the first part of the lemma above. This way we obtain an α-approximation algorithm for φ-MCPS on a simple cycle of a breakpoint graph that runs in $O(r^t)$ time where r is the number of the vertices in the simple cycle.

We can reuse the structure of a bipartite graph H presented in Sect. 7 with the weights of the edges now being the α-approximations for the MCPS_φ on the corresponding simple cycles. Following the same reasoning as in Sect. 7, we know that the minimum cost maximum matching of H leads to a MAECD of a breakpoint graph minimizing the sum of the α-approximations for the MCPS_φ on its simple cycles. Combining Theorem 2, both parts of the lemma above, and the proof of Lemma 4, we obtain an $O(n^{t+1})$ time α-approximation algorithm for φ-MCPS on a breakpoint graph. □

A.7 Lemma 5

Lemma. *If $\rho_\mathcal{O}$ is a minimum 2-break-length $\mathcal{O}$-scenario for a labeled circle (O, λ), then $\mathcal{T}(\rho_\mathcal{O})$ is a planar tree on (O, λ). In addition to that, for a planar tree $\mathcal{T}$ on (O, λ) there exists an $\mathcal{O}$-scenario $\rho_\mathcal{O}$ such that $\mathcal{T}(\rho_\mathcal{O}) = \mathcal{T}$.*

Proof. We prove the first statement by induction. It is trivially true if O has 2 vertices. We suppose it to be true for all the circles having less than $2l$ vertices and prove it for a circle having $2l$ vertices. Fix a minimum 2-break-length scenario $\rho_\mathcal{O}$. Its length is $l-1$ due to Lemma 1. The first labeled 2-break of $\rho_\mathcal{O}$ transforms (O, λ) into two vertex disjoint labeled circles (O_1, λ_1) and (O_2, λ_2) both having less vertices than O. The rest of the scenario $\rho_\mathcal{O}$ can be partitioned into $\rho_\mathcal{O}^1$ acting on the edges of O_1 and $\rho_\mathcal{O}^2$ acting on the edges of O_2. As $\rho_\mathcal{O}$ is a minimum 2-break-length scenario, $\rho_\mathcal{O}^1$ and $\rho_\mathcal{O}^2$ must also be minimum 2-break-length scenarios. By the inductive hypothesis, $\mathcal{T}(\rho_\mathcal{O}^1)$ and $\mathcal{T}(\rho_\mathcal{O}^2)$ are planar trees on (O_1, λ_1) and (O_2, λ_2) respectively. $\mathcal{T}(\rho_\mathcal{O})$ can be easily obtained from $\mathcal{T}(\rho_\mathcal{O}^1)$ and $\mathcal{T}(\rho_\mathcal{O}^2)$ by taking the union of their edges and adding an edge corresponding to the first 2-break of $\rho_\mathcal{O}$. This way we obtain a planar tree $\mathcal{T}(\rho_\mathcal{O})$ on (O, λ) proving the first statement of the lemma.

Now define the *distance* of an edge $\{x, y\}$ in $\mathcal{T}$ as the minimum number of vertices between x and y in the fixed circular embedding of $\mathcal{T}$. For example, in the rightmost tree on the top of Fig. 5 the distance of the edge $\{w, z\}$ is one, because t is in between w and z, while the distance of the edge $\{x, y\}$ is 0. An edge is said to be *short* if its distance is 0. We prove an auxiliary lemma.

Lemma. *A planar tree $\mathcal{T}$ on (O, λ) has a short edge incident to a leaf.*

Proof. Choose a leaf x in $\mathcal{T}$ incident to an edge of the minimum distance d. If $d \neq 0$, then in between the leaf and the vertex that it is adjacent to, there are d other vertices. Since $\mathcal{T}$ is planar on (O, λ), it is easy to see that there is at least one other leaf among these d vertices, which contradicts the minimality of x. □

Now take a short edge $\{x, y\}$ incident to a leaf x in $\mathcal{T}$. Take the black edges $\{a, b\}$ and $\{c, d\}$ in (O, λ) labeled with x and y respectively and separated by a gray edge $\{b, c\}$. Perform a labeled 2-break $(\{b, a\}, x), (\{c, d\}, y) \rightarrow (\{b, c\}, x), (\{a, d\}, y)$. This 2-break results in two labeled circles. One of them is a terminal graph having two edges $\{b, c\}$ with the black one labeled with x. Remove the edge $\{x, y\}$ from $\mathcal{T}$. This way we have reduced the size of the problem. The number of the vertices in the circle was reduced by two and the number of the edges in the tree was reduced by 1. We iterate this procedure to construct a required scenario. See the bottom part of Fig. 5 for an example. □

A.8 Lemma 6

Lemma. *A minimum cost planar tree on a circle can be found in $O(r^4)$ time, where r is the number of vertices of a tree.*

Proof. Farnoud and Milenkovic pose the problem of sorting permutations by cost-constrained mathematical transpositions (a sorting scenario is called a *decomposition*) [14]. They define a cost function on the set of transpositions and treat the problem, called MIN-COST-MLD, of finding a minimum cost decomposition among the minimum length transposition decompositions of a permutation. They reduce this problem to finding a minimum cost planar tree on a circle, and propose the following $O(r^4)$ time dynamic programming algorithm for a tree having r vertices.

Enumerate the vertices 1 to r while respecting their order on the circle. Define $cost(i, j)$ as the minimum cost of a planar tree on the vertices $\{i, \ldots, j\}$ for $1 \leq i < j \leq r$ and set $cost(i, i) = 0$ for $1 \leq i \leq r$.

Take a planar tree $\mathcal{T}$ on the vertices $\{1, \ldots, r\}$. If $deg(1) = 1$ and 1 is on the edge $\{1, q\}$, then the cost of $\mathcal{T}$ is equal to $\Phi(1, q)$ plus the costs of the subgraphs of $\mathcal{T}$ induced by the vertices $\{2, \ldots, q\}$ and $\{q + 1, \ldots, r\}$. If $deg(1) > 1$, then take $q = \max(\{u | \{1, u\}$ belongs to $\mathcal{T}\})$ and $s = \max(\{u |$ there is a path in $\mathcal{T}$ joining 1 and u but not visiting $q\})$. The cost of $\mathcal{T}$ is equal to $\Phi(1, q)$ plus the costs of the subgraphs of $\mathcal{T}$ induced by the vertices $\{1, \ldots, s\}$, $\{s+1, \ldots, q\}$ and $\{q, \ldots, r\}$. This observation provides us with the following equality:

$$cost(i, j) = \max(cost(i, s) + cost(s + 1, q) + cost(q, j) + \Phi(i, q) |\ i \leq s < q \leq j)$$

for $1 \leq i < j \leq r$, that leads to an $O(r^4)$ time dynamic programming algorithm for finding $cost(1, r)$. □

References

1. Bafna, V., Pevzner, P.A.: Genome rearrangements and sorting by reversals. SIAM J. Comput. **25**(2), 272–289 (1996)
2. Baudet, C., Dias, U., Dias, Z.: Sorting by weighted inversions considering length and symmetry. BMC Bioinform. **16**(19), S3 (2015)
3. Baudet, C., Lemaitre, C., Dias, Z., Gautier, C., Tannier, E., Sagot, M.-F.: Cassis: detection of genomic rearrangement breakpoints. Bioinformatics **26**(15), 1897–1898 (2010)
4. Bergeron, A., Mixtacki, J., Stoye, J.: A unifying view of genome rearrangements. In: Bücher, P., Moret, B.M.E. (eds.) WABI 2006. LNCS, vol. 4175, pp. 163–173. Springer, Heidelberg (2006). https://doi.org/10.1007/11851561_16
5. Bhuiyan, H., Chen, J., Khan, M., Marathe, M.: Fast parallel algorithms for edge-switching to achieve a target visit rate in heterogeneous graphs. In: 43rd International Conference on Parallel Processing (ICPP), pp. 60–69. IEEE (2014)
6. Bienstock, D., Günlük, O.: A degree sequence problem related to network design. Networks **24**(4), 195–205 (1994)
7. Biller, P., Knibbe, C., Guéguen, L., Tannier, E.: Breaking good: accounting for the diversity of fragile regions for estimating rearrangement distances. Genome Biol. Evol. **8**, 1427–39 (2016)
8. Bitner, J.R.: An asymptotically optimal algorithm for the dutch national flag problem. SIAM J. Comput. **11**(2), 243–262 (1982)
9. Blanchette, M., Kunisawa, T., Sankoff, D.: Parametric genome rearrangement. Gene **172**(1), 11–17 (1996)
10. Braga, M.D.V., Sagot, M.-F., Scornavacca, C., Tannier, E.: The solution space of sorting by reversals. In: Măndoiu, I., Zelikovsky, A. (eds.) ISBRA 2007. LNCS, vol. 4463, pp. 293–304. Springer, Heidelberg (2007). https://doi.org/10.1007/978-3-540-72031-7_27
11. Braga, M.D.V., Stoye, J.: The solution space of sorting by DCJ. J. Comput. Biol. **17**(9), 1145–1165 (2010)
12. Bulteau, L., Fertin, G., Tannier, E.: Genome rearrangements with indels in intergenes restrict the scenario space. BMC Bioinform. **17**(14), 426 (2016)
13. Caprara, A.: Sorting by reversals is difficult. In Proceedings of the First Annual International Conference on Computational Molecular Biology, pp. 75–83. ACM (1997)
14. Farnoud, F., Milenkovic, O.: Sorting of permutations by cost-constrained transpositions. IEEE Trans. Inf. Theory **58**(1), 3–23 (2012)
15. Fertin, G., Jean, G., Tannier, E.: Algorithms for computing the double cut and join distance on both gene order and intergenic sizes. Algorithms Mol. Biol. **12**(1), 16 (2017)
16. Lieberman-Aiden, E., Van Berkum, N.L., Williams, L., Imakaev, M., Ragoczy, T., Telling, A., Amit, I., Lajoie, B.R., Sabo, P.J., Dorschner, M.O., et al.: Comprehensive mapping of long-range interactions reveals folding principles of the human genome. science **326**(5950), 289–293 (2009)
17. Nadeau, J.H., Taylor, B.A.: Lengths of chromosomal segments conserved since divergence of man and mouse. Proc. Natl. Acad. Sci. **81**(3), 814–818 (1984)
18. Ohno, S.: Evolution by Gene Duplication. Springer, Heidelberg (1970)
19. Pulicani, S., Simonaitis, P., Rivals, E., Swenson, K.M.: Rearrangement scenarios guided by chromatin structure. In: Meidanis, J., Nakhleh, L. (eds.) Comparative Genomics. RECOMB-CG 2017. LNCS, vol. 10562, pp. 141–155. Springer, Cham (2017)

20. Shao, M., Lin, Y.: Approximating the edit distance for genomes with duplicate genes under DCJ, insertion and deletion. In: BMC bioinformatics, vol. 13, p. S13. BioMed Central (2012)
21. Shao, M., Lin, Y., Moret, B.M.E.: Sorting genomes with rearrangements and segmental duplications through trajectory graphs. In: BMC bioinformatics, vol. 14, p. S9. BioMed Central (2013)
22. Simonaitis, P., Swenson, K.M.: Finding local genome rearrangements. Algorithms Mol. Biol. **13**(1), 9 (2018)
23. Swenson, K.M., Simonaitis, P., Blanchette, M.: Models and algorithms for genome rearrangement with positional constraints. Algorithms Mol. Biol. **11**(1), 13 (2016)
24. Veron, A., Lemaitre, C., Gautier, C., Lacroix, V., Sagot, M.-F.: Close 3D proximity of evolutionary breakpoints argues for the notion of spatial synteny. BMC Genomics **12**(1), 303 (2011)
25. Yancopoulos, S., Attie, O., Friedberg, R.: Efficient sorting of genomic permutations by translocation, inversion and block interchange. Bioinformatics **21**(16), 3340–3346 (2005)