Genome Halving and Double Distance with Losses

Olivier Tremblay Savard[1⋆], Yves Gagnon[2], Denis Bertrand[3], and Nadia El-Mabrouk[4]

[1] DIRO, Université de Montréal, H3C 3J7, Canada, olivier.tremblay-savard@umontreal.ca
[2] DIRO, y.gagnon@umontreal.ca
[3] DIRO, bertrden@iro.umontreal.ca
[4] DIRO, mabrouk@iro.umontreal.ca

Abstract. Given a phylogenetic tree involving Whole Genome Duplication events, we contribute to solving the problem of computing the rearrangement and DCJ distances on a branch of the tree linking a duplication node d to a speciation node or a leaf s. In the case of a genome G at s containing exactly two copies of each gene, the *genome halving problem* is to find a perfectly duplicated genome D at d minimizing the rearrangement distance with G. We generalize the existing exact linear-time algorithm for genome halving to the case of a genome G with missing gene copies. In the case of a known ancestral duplicated genome D, we develop a greedy approach for computing the distance between G and D, called the *double distance*. Two algorithms are developed in both cases of a genome G containing exactly two copies of each gene, or at most two copies of each gene (with missing gene copies). These algorithms are shown time-efficient and very accurate for both the rearrangement and DCJ distances.

Keywords: Genome Duplication, Genome Halving, Distance, Rearrangement, DCJ, Losses.

1 Introduction

Whole genome duplication (WGD), that has the effect of simultaneously doubling all the chromosomes of a genome, is probably the most spectacular evolutionary event leading to the creation of multiple gene copies. Right after the WGD event, a genome D_{predup} is transformed into a *perfectly duplicated genome* $D = (D_{predup} \oplus D_{predup})$ containing a complete set of duplicated chromosomes. However, subsequent evolutionary events such as rearrangements, losses and local duplications blur this initial perfect duplicate status. Usually, a hypothesis that a given species has been subject to a WGD event during its evolution is based on the fact that syntenies found in pairs (exactly two paralogous regions) cover a high proportion of the genome. Such evidence has shown up across the whole eukaryote spectrum, from the protist *Giardia* to the yeast species (Gordon *et al.*, 2009), including most plant lineages, several insects, fishes, amphibians, and even mammalian species (P. Dehal, 2005). In plant lineages, the angiosperm genomes that have been completely sequenced to date all show evidence of WGD events: three ancient polyploidy events have been revealed by the *Arabidopsis thaliana* genome (Blanc *et al.*, 2003; Bowers *et al.*, 2003), one by the rice genome that might characterize all monocots (in the grass family, maize reveals an additional WGD) (Salse *et al.*, 2008), and others by the poplar, grape and papaya genomes (Soltis *et al.*, 2009).

Being able to reconstruct the ancestral pre-duplicated genome is essential from both genome and organismal evolutionary standpoints. In particular, it allows to trace back the last evolutionary events that have occurred *en route* from this ancestor to the present-day genomes, understand the specificity of a given lineage by looking at the differences that separate it from its closest evolutionary neighbour, study the variation in rearrangement

⋆ The two first authors contributed equally to this work

and loss rates among the different branches of a phylogenetic tree, and the consequences of such variations on the species. In most cases, analyzing the duplication status of syntenies in extant species allows to position the WGD events on the corresponding phylogenetic tree with confidence, leading to a tree with additional *WGD nodes* that, in contrast to the speciation nodes, each has a single child. In addition, simple assumptions can be used to infer the content of ancestral genomes from that of extant species. However, inferring ancestral gene orders is far from being a simple task.

In the case of genomes with single gene copies, many algorithms have been developed to solve the *rearrangement phylogeny problem*, which consists in inferring gene orders at the internal nodes of a tree so that the sum of distances among all branches is minimized. The most natural distance between two gene orders is the minimum number of rearrangement events required to transform one gene order into the other. The rearrangements that have been most studied by the genome rearrangement community are inversions and reciprocal translocations (including fusion and fission). More recently, another distance that has been extensively studied is the Double Cut-and-Join (DCJ) distance which represents a greater repertoire of rearrangement events while giving rise to simpler formal results (Bergeron *et al.*, 2006, 2009; Yancopoulos *et al.*, 2005).

A prerequisite for applying any of the algorithms developed for solving the rearrangement phylogeny problem to a phylogeny with WGD nodes is to be able to compute the distance on a branch of the phylogeny. However, this is far from being straightforward, as the orthology relationship between duplicated genes is not set. In particular, computing the distance between a rearranged duplicated genome G (a genome with exactly two copies of each gene but in any order) and a perfectly duplicated genome D, called the *double distance* in (Gavranović and Tannier, 2010; Tannier *et al.*, 2009), has been shown to be NP-hard for the DCJ distance (Tannier *et al.*, 2009). When the ancestral genome D is unknown, the *genome halving problem* seeks for a perfectly duplicated genome D minimizing the rearrangement distance between G and D. In 2003, we have presented the first formal result related to genome duplication, which is an exact linear-time algorithm for solving the genome halving problem (El-Mabrouk and Sankoff, 2003). Our results have been reformulated by Alekseyev and Pevzner (Alekseyev and Pevzner, 2007a) using an alternative representation of the breakpoint graph. Subsequently, Sankoff and colleagues (Zheng *et al.*, 2008b,a), and more recently Gavranović and Tannier (Gavranović and Tannier, 2010), used variations of the genome halving strategy (*Guided Genome Halving* or GGH) to find the preduplicated ancestor of a doubled genome in the presence of a non-duplicated outgroup (Zheng *et al.*, 2008a,b).

In this paper, we contribute to solving a number of problems related to the computation of the rearrangement and DCJ distances on a branch of a phylogenetic tree connecting a first WGD node to a speciation node or a leaf, in both cases of a known and unknown preduplicated genome (label of the WGD node). In the case of an unknown ancestral genome, our result is a generalization of the genome halving algorithm to a genome G with missing gene copies. In the case of a known ancestral genome D, we develop two greedy algorithms for both cases of a genome G containing exactly two copies of each gene, or at most two copies of each gene (with missing gene copies). These algorithms are shown time-efficient and very accurate for both the rearrangement and DCJ distances.

2 Preliminaries

Let Σ be a set of n genes. A *string* is a sequence of genes from Σ, where each gene is signed (+ or −) depending on its transcriptional orientation. The *reverse* of a string $X = x_1 x_2 \dots x_r$ is the string $-X = -x_r\ -x_{r-1} \dots -x_1$. A *chromosome* is a string, and a *genome* is a collection of chromosomes. A *unichromosomal* genome has a single chromosome, and a *multichromosomal* genome has at least two nonnull chromosomes $C_1, C_2, \dots C_N$. A *circular chromosome* is a string $x_1 \dots x_r$, where x_1 is considered to follow x_r. A chromosome that is not circular is *linear*. To represent its endpoints, we add an "artificial gene", denoted O, at each extremity. In other words, a linear chromosome is a string of the form $Ox_1 \dots x_r O$.

In this paper, we consider both uni- and multichromosomal genomes. As most unichromosomal genomes are formed by a circular chromosome, and most multichromosomal genomes are formed by linear chromosomes, only circular unichromosomal genomes, and linear multichromosomal genomes are considered here.

2.1 Evolutionary events and genomic distances

All the following evolutionary events apply to both uni- and multichromosomal genomes, except translocations that are only relevant for multichromosomal genomes.

- A *reversal* (or *inversion*) is an operation that replaces some proper substring of a chromosome by its reverse.
- A *translocation* between two chromosomes $X = X_1X_2$ and $Y = Y_1Y_2$ is an event transforming them into the two chromosomes X_1Y_2 and Y_1X_2, or into $X_1(-Y_1)$ and $(-Y_2)X_2$. Two special cases of reciprocal translocations are *fusions* (if one of the two chromosomes generated by the translocation is an empty string) and *fissions* (if one of the two input chromosomes is the empty string).
- A *Whole Genome Duplication* (WGD) is an event resulting in the duplication of the genomic content. More precisely, in the case of a multichromosomal genome $G = \{C_1, C_2, \dots C_N\}$, a WGD transforms G into a multichromosomal genome $D = \{C_1, C'_1, C_2, C'_2, \dots C_N, C'_N\}$ containing $2N$ chromosomes where, for each $1 \leq i \leq N$, $C_i = C'_i$. In the case of a circular genome G represented by the string $x_1 x_2 \dots x_r$, a WGD transforms G into the circular genome D represented by the string $x_1 x_2 \dots x_r\ x'_1 x'_2 \dots x'_r$, where, for each $1 \leq j \leq r$, $x_j = x'_j$.
- Finally, a *loss* is an operation removing a proper substring from a chromosome.

A *rearrangement event* will refer to an inversion or a translocation event. The *rearrangement distance* between two genomes G and H (with the same gene content or not), denoted $d_R(G, H)$, is the minimum number of rearrangement events in a scenario transforming G into H. In the case of genomes with single gene copies, computing the inversion and/or translocation distance has been shown to be a polynomial-time problem, and the best developed method runs in linear time (Bader *et al.*, 2001; Bergeron *et al.*, 2004).

Another distance that has been extensively studied in the last years is the DCJ distance (Bergeron *et al.*, 2006, 2009; Yancopoulos *et al.*, 2005). A *Double-Cut-and-Join (DCJ)* is an operation that "cuts" two adjacencies pq and rs in a genome, and replaces them by

either pr and qs, or ps and qr. The repertoire of DCJ operations include inversions, reciprocal translocations, fusions and fissions, but also other "artificial" rearrangement operations such as the creation of "intermediate" circular chromosomes. Using such a circular intermediate, a transposition can actually be mimicked by two DCJ operations. Computing the DCJ distance between two signed permutations is a linear-time problem.

2.2 Genome definitions

In what follows, we consider G to be a genome defined on a set Σ of genes, i.e. g is in G iff $g \in \Sigma$.

- G is an *extension* of a genome H, iff the gene content of H is a subset of the gene content of G, and there is a sequence of gene insertions transforming H into G.
- G is a *singleton genome* iff each gene is present exactly once in G.
- G is a *Rearranged Duplicated genome (RD genome)* iff each gene is present exactly twice in G.
- G is a *perfectly duplicated genome* (or *duplicated genome* for short) iff:
  - *The multichromosomal case:* G is an RD genome containing an even number $2N$ of chromosomes, with two identical copies of each chromosome. If D is the set of the N different chromosomes, then we write $G = (D \oplus D)$.
  - *The circular case:* G is an RD genome and there is a string D such that G is exactly D followed by D. We also write $G = (D \oplus D)$.
- G is a *Rearranged Duplicated genome with Losses (RDL genome)* iff each gene is present at least once and at most twice in G.
- G is a *Duplicated genome with Losses (DL genome)* iff each gene is present in one or two copies in G, and if a duplicated genome D can be obtained from G by an appropriate insertion of an additional copy of each *singleton* (gene present in one copy in G). In other words, there is an extension of G that is a duplicated genome.

Let G be an RD genome and H be an RDL genome. We define the evolutionary cost $\mathcal{E}(G, H)$ as the minimum number of inversions, translocations and losses required to transform G into H.

2.3 The breakpoint graph

In a series of papers (Hannenhalli, 1995; Hannenhalli and Pevzner, 1999, 1995), Hannenhalli and Pevzner (hereafter HP) developed polynomial-time algorithms for computing the rearrangement distance (inversion only, translocation only, or inversion+translocation) between two singleton genomes G and H on Σ. The algorithms all depend on a bicolored graph $\mathcal{B}(G, H)$, called the *breakpoint graph*, constructed from G and H as follows (Tesler's formalism (Tesler, 2002)).

Graph $\mathcal{B}(G,H)$: If a gene x of Σ has a positive sign, replace it by the pair $x^t x^h$, and if it is negative, replace it by $x^h x^t$. Then the set V of vertices of $\mathcal{B}(G,H)$ is the set of x^t and x^h for all x in Σ. Any two vertices of V that are adjacent in some chromosome of G, other than x^t and x^h deriving from the same x, are connected by a black edge, and any two that are adjacent in H are connected by a gray edge. Notice that adjacencies to O are not represented.

In the case of circular chromosomes, each vertex in V is incident to exactly one black and one gray edge, and thus the graph uniquely decomposes into $c(G,H)$ disjoint cycles of alternating edge colors (*alternating cycles* for short).

In the case of G and H being multichromosomal genomes, let an *endpoint vertex of G* (resp. of H) be a vertex of V adjacent to O in G (resp. in H). Consider the *degree of a vertex* as being the number of edges incident to this vertex. Then any vertex has degree zero if it is an endpoint in both G and H, one if it is an endpoint in exactly one of the two genomes, and two otherwise. Thus, the graph decomposes into $c(G,H)$ cycles and $p(G,H)$ paths of alternating edge color. Notice that a path may contain only one vertex and no edges. We denote by p_{GG} (resp. p_{HH}) the number of paths linking two endpoints of G (resp. of H). If G and H have the same number of chromosomes, then $p_{GG} = p_{HH}$. Otherwise, suppose w.l.o.g. that G has more chromosomes than H, then $p_{HH} \leq p_{GG}$.

The rearrangement distance: Although somehow different algorithms are required for sorting by translocation only, inversion only or inversion+translocation, all results in (Hannenhalli, 1995; Hannenhalli and Pevzner, 1999, 1995), revisited by Tesler (Tesler, 2002) for multichromosomal genomes, can be summarized by a unique formula given bellow:

$$\mathsf{HP}: \quad d_R(G,H) = n + N - C(G,H) + h(G,H)$$

where n is the number of genes, N is the number of chromosomes of G, and $C(G,H) = c(G,H) + p(G,H) - p_{GG}$. In the case of circular genomes, $N = p(G,H) = p_{GG} = 0$. As for $h(G,H)$, it is a correction parameter that has a different value depending on the considered model. In all cases, $h(G,H)$ is related to the decomposition of $\mathcal{B}(G,H)$ into components, where a *component* is a maximal set of crossing cycles and paths. A component is termed *good* if it can be transformed into a set of cycles of size 1 by increasing the number of cycles at each step, and *bad* otherwise. The parameter $h(G,H)$ reflects the number of bad components of the graph. As the probability for a component to be bad is low, the value of $h(G,H)$ is usually low compared to the dominating parameter $C(G,H)$.

The DCJ distance: Based on the breakpoint graph, the DCJ distance between G and H can be expressed as follows (Bergeron *et al.*, 2006; Tannier *et al.*, 2009):

$$\mathsf{DCJ}: \quad d_{DCJ}(G,H) = n - \left(c(G,H) + \frac{p_{even}}{2} \right)$$

where p_{even} is the number of paths with an even number (≥ 0) of edges.

3 Genome Halving with Losses

Given an RD genome G, the *genome halving problem* is to find a duplicated genome D minimizing the rearrangement distance with G. In other words, let $d_R(G)$ be the minimum

rearrangement distance between G and any duplicated genome D. Then the problem is to find a duplicated genome D such that $d_R(G) = d_R(G, D)$.

In (El-Mabrouk and Sankoff, 2003) we have developed a linear-time algorithm, called Algorithm Dedouble, for the reversals-only version of the problem (in the case of unichromosomal genomes), the translocations-only version, and the version with both reversals and translocations. The approach was to start from a *partial breakpoint graph* $\mathcal{B}(G)$, i.e. the breakpoint graph with the set of edges restricted to the black edges representing G (see Figure 2.(b) for an example), and to complete this graph with a set of "valid" gray edges, i.e. gray edges representing a duplicated genome D (thin edges in Figure 2.(c)), in a way maximizing the number of cycles and paths (parameters $c(G, D)$ and $p(G, D)$ in the HP formula). The second step was then to perform modifications on the obtained graph in order to remove bad components that can be avoided, and obtain a duplicated genome D minimizing the rearrangement distance with G (i.e. minimizing the HP formula).

M_1: G	$\xleftarrow{Inv,\ Trans,\ Loss}$	$D = (D_{predup} \oplus D_{predup})$	$\xleftarrow{WGD}$	D_{predup}
RDL genome		Duplicated genome		Singleton genome
M_2: G	$\xleftarrow{Inv,\ Trans}$ A $\xleftarrow{Loss}$	$D = (D_{predup} \oplus D_{predup})$	$\xleftarrow{WGD}$	D_{predup}
RDL genome	DL genome	Duplicated genome		Singleton genome
M_3: G	$\xleftarrow{Loss}$ G' $\xleftarrow{Inv,\ Trans}$	$D = (D_{predup} \oplus D_{predup})$	$\xleftarrow{WGD}$	D_{predup}
RDL genome	RD genome	Duplicated genome		Singleton genome

Fig. 1. Evolutionary models M_1, M_2 and M_3, considered for a present-day rearranged duplicated genome with losses G. Direction of evolution is represented by arrows orientation.

Here, we seek to generalize Algorithm Dedouble to a present-day genome G containing both duplicated genes and singletons, i.e. to an RDL genome. Let G be a present-day RDL genome. We assume that G has evolved from an ancestral singleton genome through a WGD, and a sequence of inversions, translocations and loss events. We are then interested in finding such a pre-duplicated singleton genome D_{predup} minimizing the number of rearrangements needed to obtain G (see model M_1 in Figure 1). Note that we do not attempt to minimize the number of losses.

The following theorem allows to reduce the evolutionary model to a simpler one (model M_2 in Figure 1), where all losses occur first, followed by all rearrangement events.

Theorem 1. *Let G be an RDL genome and D be a duplicated genome. Then there exists a DL genome A with the same gene content as G, such that $d_R(G, A) = d_R(G, D)$.*

Proof: By induction on $n = \mathcal{E}(G, D)$
1. The property is trivially verified for $n = 0$ and $n = 1$.
2. Suppose the induction hypothesis is verified for a given $n \geq 1$. Now suppose that $\mathcal{E}(G, D) = n+1$, and let $\mathcal{E} = E_1, E_2, \ldots E_n, E_{n+1}$ be a sequence of $n+1$ events transforming a duplicated genome D into an RDL genome G. Let G' be the genome obtained after performing the

sequence of n events $\mathcal{E}' = E_1, E_2, \dots E_n$ on D. Then $n = \mathcal{E}(G', D)$ as otherwise (if n is not the minimum number of events transforming D into G') $n+1$ would not be the minimum number of events transforming D into G. Moreover, by the induction hypothesis, there exists a DL genome A' for which $d_R(G', A') = d_R(G', D)$.

If E_{n+1} is a rearrangement event, the DL genome A with the same gene content as G is equal to A'. Then, we have $d_R(G, A) = d_R(G, A') = d_R(G', A') + 1$ and $d_R(G, D) = d_R(G', D) + 1$. Therefore, $d_R(G, A) = d_R(G, D)$.

Otherwise, E_{n+1} is a loss event. Let A be the DL genome obtained from A' by removing the genes that are removed by the loss operation E_{n+1}. Then, it is easy to see that a minimum sequence of k rearrangement events transforming A' into G' can be converted into a sequence of k rearrangement events transforming A into G (just by removing the lost genes from the inverted or translocated segments). Therefore $d_R(G, A) \leq d_R(G', A')$. Similarly, a minimum sequence of k rearrangement events transforming A into G can be converted into a sequence of k rearrangement events transforming A' into G'. Therefore, $d_R(G, A) = d_R(G', A') = d_R(G, D)$ □

Corollary 1. *Let G be an RDL genome, and A be a DL genome with the same gene content as G, minimizing the cost $d_R(G, A)$. If D is the duplicated genome that is an extension of A, then $d_R(G) = d_R(G, D)$.*

Proof: Let A be a DL genome with the same gene content as G minimizing the cost $d_R(G, A)$, and D be the duplicated genome that is an extension of A. Then we have $d_R(G, D) = d_R(G, A)$. Suppose $d_R(G) \neq d_R(G, A)$, i.e. $d_R(G, A) > d_R(G)$. Let D' be a duplicated genome such that $d_R(G, D') = d_R(G)$. Then, from Theorem 1, there is a DL genome A' such that $d_R(G, A') = d_R(G, D') = d_R(G)$. And thus $d_R(G, A') < d_R(G, A)$, which is a contradiction with the fact that A minimizes the rearrangement cost □

Therefore, finding a duplicated genome D such that $d_R(G) = d_R(G, D)$ can be reduced to the problem of finding a DL genome A with the gene content of G such that $d_R(G, A)$ is minimal over all DL genomes with the gene content of G. In other words, loss events can be ignored.

To find such a DL genome A, we use a generalization of Algorithm Dedouble, called Algorithm Dedouble-RDL(G), that proceeds as follows:

1. Consider the RD genome G' obtained from G by "gluing" singletons to an adjacent gene. More precisely, consider a given orientation for chromosomes. Then, for each maximum sequence S of singletons in G: (1) if S is a chromosome, then just remove this chromosome; (2) otherwise, if S is connected to a left extremity of a chromosome, then replace its successor x (the gene representing the right adjacency of S in G) by the artificial gene $x' = Sx$; (3) otherwise, if S is not connected to a left extremity of a chromosome, then replace its predecessor x (possibly already updated in step (2)) by a new artificial gene x' representing the sequence xS.
2. Use Algorithm Dedouble to infer a duplicated genome A' from G'.
3. Recover a DL genome A from A' by replacing each of its artificial gene by its corresponding sequence of singletons, and by adding all removed chromosomes of G (formed exclusively of singletons).

The following theorem immediately follows from the fact that Algorithm Dedouble outputs a doubled genome A' minimizing the distance to G', and that singletons are preserved in the same order in G and A.

Theorem 2. *Let G be an RDL genome and A be the DL genome resulting from* Algorithm Dedouble-RDL(G). *Then $d_R(G, A) = d_R(G)$.*

4 An algorithm for the Double Distance

Let G be an RD genome and $D = (D_{predup} \oplus D_{predup})$ be a duplicated genome. The problem of computing the DCJ distance between G and D has already been shown to be an NP-hard problem (Tannier *et al.*, 2009), contrary to the polynomial-time complexity of computing the distance between two singleton genomes. This difference in complexity is the result of the missing one-to-one orthology relationship between the gene copies. In other words, given a labelling of the genes in G, the problem is to find a labelling of the genes in D leading to a minimum distance between G and D.

Consider a given beginning gene, in the case of a circular genome, or a given order and left-to-right orientation of chromosomes in the case of a multichromosomal genome G. Then, for each gene x (present in two copies in G and also in D), label the first occurrence of x in G as x_1 and the second as x_2. Let $\mathcal{B}(G)$ be the partial breakpoint graph for G. To complete this partial graph, each double adjacency (x^r, y^s) in D (where $r, s \in \{t, h\}$) should be represented in a completed graph $\mathcal{B}(G, D^L)$, where D^L is a labelling of genome D, by either one of the following pairs of gray edges: $\{(x_1^r, y_1^s), (x_2^r, y_2^s)\}$, or $\{(x_1^r, y_2^s), (x_2^r, y_1^s)\}$. Each of these two cases leads to a different labelling of the gene copies in D. The problem is then to choose the pairs of gray edges allowing to minimize the HP formula in the case of the rearrangement distance, or the DCJ formula in the case of minimizing the DCJ distance.

Here, we focus on maximizing the dominating value $C(G, D)$ in the HP formula. In the case of genome halving, this simplification has been called the Weak Genome Halving Problem (Alekseyev and Pevzner, 2007a). We similarly define our simplified problem as follows:

Weak Double Distance Problem. *For a given labelled RD genome G and an unlabelled duplicated genome D, find a labelling D^L of D such that the number of alternating cycles $C(G, D)$ of the breakpoint graph $\mathcal{B}(G, D^L)$ is maximized over all possible labellings of D.*

Notice that, in the case of a circular genome, a labelling of D maximizing the parameter $C(G, D)$ also maximizes the DCJ formula, as $C(G, D) = c(G, D)$ in this case. In the multichromosomal case, a labelling of D maximizing $C(G, D)$ is likely to also maximize the DCJ formula, though there is no guarantee for that.

Clearly, the "best" exhaustive approach trying all possible labellings for D has a worst running-time complexity in $O(n.2^n)$ for $n = |\Sigma|$. Indeed, D has 2^n possible labellings, and for each labelling, the most efficient approach for computing the rearrangement distance between G and D is linear.

4.1 Circular genomes

Let G be a circular RD genome and D be a circular duplicated genome. We consider the contracted breakpoint graph representation $\mathcal{CB}(D, G)$ defined as follows: the set of vertices of

$\mathcal{CB}(D,G)$ is $V = \{x^r, \text{for all}\, x \in \Sigma \,\text{and}\, r \in \{t,h\}\}$. Any two vertices which are adjacent in D (except the extremities of a same gene) are connected by two parallel gray edges, and any two adjacent in G (except the extremities of a same gene) are connected by a black edge (see Figure 2.(a)). Such representation has previously been used in the context of genome halving for circular (Alekseyev and Pevzner, 2007b) and multichromosomal genomes (Gavranović and Tannier, 2010), with the difference that each gray edge was represented exactly once. It follows that each vertex of $\mathcal{CB}(D,G)$ is adjacent to exactly two gray edges and two black edges.

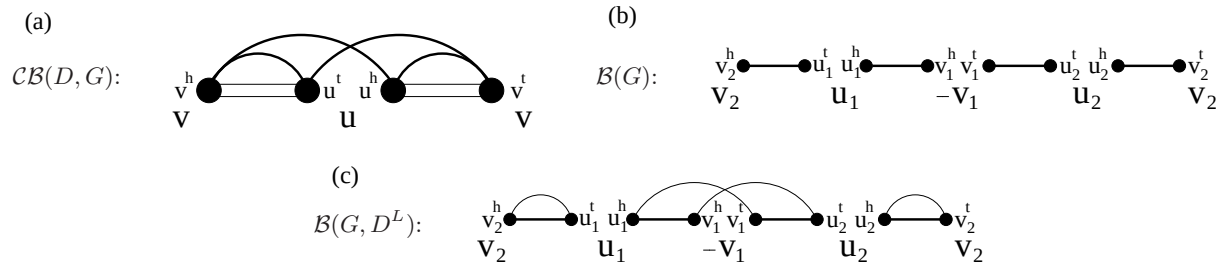

Fig. 2. (a) The contracted breakpoint graph $\mathcal{CB}(D,G)$ for the circular RD genome $G = (u\ -v\ u\ v)$ and the circular duplicated genome $D = (uv) \oplus (uv)$. Gray edges (thin lines) represent genome D and black edges (thick lines) represent genome G. (b) The partial breakpoint graph $\mathcal{B}(G)$. (c) The completed breakpoint graph $\mathcal{B}(G,D^L)$ corresponding to the labelling $G = (u_1\ -v_1\ u_2\ v_2)$ and $D^L = (u_1\ v_1) \oplus (u_2\ v_2)$. This labelling D^L, leading to 3 cycles, is optimal. The resulting rearrangement distance is 1.

Now consider the partial breakpoint graph $\mathcal{B}(G)$ of the labelled RD genome G (see Figure 2.(b)). Let C be an alternating edge-colour cycle in $\mathcal{CB}(D,G)$ with the set of vertices V, the set of black edges $b = \{b_1, \cdots b_m\}$, and the set of gray edges $g = \{g_1, \cdots g_m\}$. Then we can construct a corresponding cycle in $\mathcal{B}(G)$. More precisely, let $b^{\mathcal{B}} = \{b_1^{\mathcal{B}}, \cdots b_m^{\mathcal{B}}\}$ be a set of black edges in $\mathcal{B}(G)$ corresponding to $\{b_1, \cdots b_m\}$ (i.e., for each i, the two vertices adjacent to $b_i^{\mathcal{B}}$ are two labelled copies of the vertices adjacent to b_i), and let $V_b^{\mathcal{B}}$ be the set of all the vertices adjacent to $b_i^{\mathcal{B}}$, for all i. Then there is a set of gray edges $g^{\mathcal{B}} = \{g_1^{\mathcal{B}}, \cdots g_m^{\mathcal{B}}\}$ corresponding to $\{g_1, \cdots g_m\}$ and linking vertices of $V_b^{\mathcal{B}}$, allowing to form a single alternating cycle in $\mathcal{B}(G)$.

This observation leads to a greedy approach for labelling the genome D, or equivalently completing the partial graph $\mathcal{B}(G)$. Formally, a *completed graph* $\mathcal{B}(G,D^L)$ is a graph obtained from $\mathcal{B}(G)$ by adding gray edges such that each vertex of $\mathcal{B}(G,D^L)$ is adjacent to exactly 2 edges (one black and one gray) (Figure 2.(c)).

The general idea of Algorithm Double-Distance(G,D) given in Figure 3 is: at each step, pick an alternating cycle of minimum size from $\mathcal{CB}(D,G)$, construct the corresponding cycle in $\mathcal{B}(G)$, and then remove from $\mathcal{CB}(D,G)$ all used edges. The algorithm stops when the partial graph is completed.

Lemma 1. *Algorithm Double-Distance(G,D) terminates and results in a completed graph* $\mathcal{B}(G,D^L)$.

Algorithm Double-Distance(G,D)
Input: $\mathcal{CB}(D,G)$ and the partial graph $\mathcal{B}(G)$;
Output: The graph $\mathcal{B}(G)$ completed with gray edges (i.e. $\mathcal{B}(G,D^L)$);
1. **For** $CSize = 1$ to n **Do** ;
2. **For** $CVertex = b_1^l$ to b_n^l **Do**
3. **If** $\mathcal{CB}(D,G)$ is empty (i.e. no edges left)
4. **Return** ;
5. **If** there is an alternating cycle C of size $CSize$ beginning at $CVertex$ **Then**
Construct a corresponding cycle in $\mathcal{B}(G)$:
6. Let b be the set of black edges and g the set of gray edges of C;
7. Consider a set of black edges $b^{\mathcal{B}}$ corresponding to b in $\mathcal{B}(G)$ and not considered in a previous step;
8. Let $V_b^{\mathcal{B}}$ be the set of vertices of $b^{\mathcal{B}}$;
9. Construct a set of gray edges $g^{\mathcal{B}}$ corresponding to g and linking vertices of $V_b^{\mathcal{B}}$ in $\mathcal{B}(G)$;
10. Remove from $\mathcal{CB}(D,G)$ all edges of C;
11. **End If**
12. **End For**
13. **End For**

Fig. 3. A greedy approach for completing the partial graph $\mathcal{B}(G)$ with gray edges representing the genome D. Here, $n = |\Sigma|$ is the number of different genes, and $b_1, b_2, \ldots b_n$ is a left-to-right ordering of the black edges of $\mathcal{CB}(D,G)$. For each i, b_i^l is the vertex representing the left adjacency of b_i. The size of a cycle is the number of black (or equivalently gray) edges of the cycle.

Proof. We will show that, at each step, if the graph $\mathcal{CB}(D,G)$ is non-empty (it still contains edges), then it contains an alternating cycle. Notice first that if the graph $\mathcal{CB}(D,G)$ is a non-empty balanced graph (i.e. every vertex has the same number of incident gray and black edges), then it contains at least one alternating cyle. This is a consequence of the fact that a balanced graph contains an alternating eulerian cycle in every connected component (Kotzig, 1968; Pevzner, 1995).

Suppose that at a given step of the algorithm, $\mathcal{CB}(D,G)$ is a non-empty unbalanced graph. Clearly such a graph can not be the input of the algorithm, as at the beginning, each vertex has two adjacent edges of each color. On the other hand, it can not be the graph obtained after an iteration of the algorithm, as when edges adjacent to a vertex are removed, they are removed by bicolored pairs (i.e. one gray and one black edge). Therefore, if the graph $\mathcal{CB}(D,G)$ is non-empty, it is balanced and thus it contains at least one alternating cycle. Moreover, as only a finite number of alternating cycles can emerge from this graph, the algorithm is guaranteed to terminate (because at each step, at least one alternating cycle is removed).

Finally, since all gray edges of $\mathcal{CB}(D,G)$ are inserted in $\mathcal{B}(G)$, and each vertex of $\mathcal{B}(G)$ is considered only once as an adjacency of a new inserted gray edge (line 7), clearly when no edges remain in $\mathcal{CB}(D,G)$, the obtained graph $\mathcal{B}(G)$ is a completed graph □

Let $D^L = u_{1,\alpha_1} \ldots u_{r,\alpha_r} u_{1,\overline{\alpha_1}} \ldots u_{r,\overline{\alpha_r}}$ be a labelling for $D = u_1 \ldots u_r u_1 \ldots u_r$, where, for each $1 \leq i \leq r$, $\alpha_i \in \{1,2\}$, and $\overline{\alpha_i}$ denotes the complementary element of α_i in $\{1,2\}$. A bi-circular representation of D^L is the pair of circular chromosomes $\{u_{1,\alpha_1} \ldots u_{r,\alpha_r}, u_{1,\overline{\alpha_1}} \ldots u_{r,\overline{\alpha_r}}\}$.

Lemma 2. *The gray edges of the completed graph resulting from the execution of Algorithm Double-Distance(G,D) represent either a labelling of D, or a bi-circular representation of a labelling of D.*

Proof. At each execution of the internal For Loop (line 2), the algorithm selects an alternating cycle C in $\mathcal{CB}(D,G)$, and constructs a corresponding cycle in $\mathcal{B}(G)$. In other words, for each gray edge g in C linking two vertices x^r, y^s, for r and $s \in \{t,h\}$, we construct, in $\mathcal{B}(G)$, a corresponding edge $g^{\mathcal{B}}$ linking two labelled copies x^r_α and y^s_β of x and y respectively, for α and $\beta \in \{1,2\}$. Assume w.l.o.g. that $x^r_\alpha = x^h_1$ and that $y^s_\beta = y^t_1$. Creating this gray edge in $\mathcal{B}(G)$ is equivalent to labelling the genome D so that the two following adjacencies are created: (x^h_1, y^t_1) and (x^h_2, y^t_2) (the other possibility being (x^h_1, y^t_2) and (x^h_2, y^t_1)). In order to end up with a labelling of D, at each step the created adjacencies should not be in conflict with the ones already created. Suppose this is not the case. In other words, the newly created adjacencies (x^h_1, y^t_1) and (x^h_2, y^t_2) lead to a conflict. This can happen in one of the two following situations:

- We have already created, in a previous step of the algorithm, an adjacency (z^u, y^t), with $z^u \neq x^h$ and $u \in \{t,h\}$. This is impossible as (z^u, y^t) would have been an adjacency of D. But since D is a duplicated genome, it can not involve two different adjacencies for a given gene extremity y^t.
- We have already formed, following the previous steps of the algorithm, a segment of adjacencies resulting in y_1 being on the left-side of x_1, i.e. a segment of form $+y_1 \cdots + x_1$. This means that we also have created the homologous segment $+y_2 \cdots + x_2$. Then having to add the gray edge (x^h_1, y^t_1) means that D is a duplicated circular genome of form $+y \cdots + x + y \cdots + x$. Therefore, adding the last two remaining gray edges (x^h_1, y^t_1) and (x^h_2, y^t_2) results in a bi-circular representation of the labelling $+y_1 \cdots + x_1 + y_2 \cdots + x_2$ of D □

Following the proof of Lemma 2, it is immediate to see that, if the set of strings represented by the gray edges of the output completed graph of Algorithm Double-Distance(G,D) is not a labelling of D, then changing the last chosen set of gray edges results in a correct labelling of D. More precisely, instead of adding the last two gray edges (x^h_1, y^t_1) and (x^h_2, y^t_2) the right choice would have been to add the two gray edges (x^h_1, y^t_2) and (x^h_2, y^t_1).

Complexity: As each vertex is adjacent to two black edges, finding an alternating cycle of size k beginning at a given vertex of $\mathcal{CB}(D,G)$ (line 5) can be done in $O(2^k)$ time. Therefore, the algorithm has a worst running-time complexity bounded by $\Sigma^n_{k=1} n.2^k$, which is not better than the exhaustive approach in $O(n.2^n)$. However, as demonstrated in the experimental part of this paper, it is actually a much faster approach in practice. This is due to the edge removal step (line 7), which allows to reduce the graph quickly, and to stop the process after a small number of iterations.

4.2 Multichromosomal genomes

In the case of G and D being multichromosomal genomes, we define the contracted breakpoint graph $\mathcal{CB}(D,G)$ as before, except that it contains an additional vertex O such that any endpoint vertex in D is connected to O by two gray edges, and any endpoint vertex in G

is connected to O by a black edge. It follows that, except O that is adjacent to $2N_G$ black edges and $2N_D$ gray edges, N_G being the number of chromosomes of G, and N_D the number of chromosomes of D, each other vertex is adjacent to exactly two gray edges and two black edges.

Algorithm Double-Distance(G,D) can be used in the case of multichromosomal genomes if we replace line 3 with " **If** $\mathcal{CB}(D, G)$ is acyclic". At the end of the algorithm, $\mathcal{CB}(D, G)$ is acyclic and the only remaining paths connect two vertices that are both endpoints of G, or both endpoints of D (as a path connecting two endpoints of two different genomes would have been closed by Algorithm Double-Distance(G,D) to form a cycle). Then, to complete the graph $\mathcal{B}(G)$, it suffices to add the remaining paths of $\mathcal{CB}(D, G)$.

Due to the $2(N_G + N_D)$ edges incident to O, the worst-time complexity is the one for circular genomes multiplied by $N_G.N_D$, i.e. $O(n.N_G.N_D.2^n)$. Hopefully in practice, n is not a tight upper bound as exploration eventually stops for much smaller cycle sizes.

5 Double Distance with Losses

Similarly to genome halving, we aim to generalize the double distance to genomes with possibly missing gene copies. More precisely, given an RDL genome G and a duplicated genome $D = (D_{predup} \oplus D_{predup})$ on the same set of genes, how can we compute the distance between G and D? Notice first that a simple generalization of the strategy used for Algorithm Dedouble-RDL which would consist in (1) "gluing" the singletons of G to an adjacent gene and removing the corresponding copies from D, and (2) applying Algorithm Double-Distance to the obtained genomes G' and D', does not solve the problem. Indeed, the distance obtained after applying Algorithm Double-Distance(G',D') would only be a lower bound of the distance between genomes G and D. For example, suppose that an optimal sequence of rearrangements are performed on G' to transform it into D', and that the singletons are then unglued to form the DL genome A. The problem is that the perfectly duplicated genome D is not necessarily an extension of A.

As reducing the evolutionary model M_1 to M_2 does not help in this case, we instead reduce it to the symmetrical model M_3 (Figure 1.(c)), where all rearrangements occur first, followed by all losses. The next theorem justifies this reduction.

Theorem 3. *Let G be an RDL genome and D be a duplicated genome. Then there is an RD genome G' that is an extension of G, such that $d_R(G', D) = d_R(G, D)$.*

Proof. Proof by induction, very similar to the one of Theorem 1 □

The problem thus reduces to the one of finding an RD genome G' that is an extension of G, minimizing the weak double distance to D. Such a genome G' is called an *optimal extension of G*.

In the following section we focus on circular genomes. We subsequently explain, in section 5.2, the modifications that should be introduced in the case of multichromosomal genomes.

5.1 Circular genomes

The following lemma allows to limit the possible insertion positions of a missing gene in G.

Lemma 3. *Let x be a singleton in G, labelled x_1. Then, there is an optimal extension G' of G such that the inserted copy x_2 of x has a preserved (left or right) adjacency with D.*

Proof. Let G' be an optimal extension of G. More precisely, G' is an RD genome that is an extension of G and $d_R(G', D)$ is minimal over all RD genomes that are extensions of G. Let $\mathcal{R} = \{r_1, \cdots r_p\}$ be an optimal sequence of inversions transforming G' into D. If G' is a labelled genome, then applying $\mathcal{R}$ to G' results in a labelled genome D. Let L_2 and R_2 be respectively the left and right adjacencies of x_2 in D. If x_2 is left-adjacent to one copy of L or right-adjacent to one copy of R in G', then the lemma is verified. Otherwise, consider the genome G''_{glue} obtained from G' by removing the copy x_2, and gluing it to either L_2 or R_2. W.l.o.g., let's say that x_2 is glued to L_2, i.e. L_2 is replaced by $L''_2 = L_2\, x_2$. Then, for each $r_i \in \mathcal{R}$ acting on the segment X_i, consider the inversion r'_i acting on the segment X'_i obtained from X_i by replacing L_2 by L''_2, and removing x_2 from X_i, if applicable. Then, applying the sequence of inversions $\mathcal{R}' = \{r'_1, \cdots r'_p\}$ to G''_{glue} and then ungluing L''_2 gives rise to the genome D. Therefore, the genome G'' obtained by ungluing L''_2 in G''_{glue} is an optimal extension genome of G verifying the property that x_2 has a preserved adjacency with D □

In other words, x_2 can be inserted in G adjacent to one of the two copies of its left or right neighbour in D. The problem is to find the appropriate neighbour and copy number, as this would influence the number of operations required to place x_1 adjacent to the other copies of its neighbours. The idea of the algorithm will therefore be to create the adjacencies for x_1 before those for x_2.

We consider a tricolored graph $\mathcal{CB}(D, G)$ obtained by adding to the contracted breakpoint graph representation, introduced in the previous section, a new set of "dotted edges" defined as follows: for each vertex y^s representing an extremity of a singleton y of G (for $s \in \{t, h\}$) , construct a dotted edge linking y^s to its adjacent vertex in D (see Figure 5.(a) left). Algorithm Double-Distance-with-Loss(G,D), presented in Figure 4, takes as input the tricolored graph $\mathcal{CB}(D, G)$ and the partial breakpoint graph $\mathcal{B}(G)$, and completes $\mathcal{B}(G)$ with appropriate gray edges, but also additional black edges, corresponding to the missing singleton copies that have to be inserted in G to produce an extension G'.

In the following developments, as well as in Figure 4, alternating cycles only refer to cycles of black and gray alternating edge colors (i.e. dotted edges are not considered in the cycles). The algorithm proceeds as follows (see Figure 5 for an example):

1. The For Loop 2 - 9: Proceed as in Algorithm Double-Distance(G,D)), i.e. pick an alternating cycle of minimum size from $\mathcal{CB}(D, G)$, construct the corresponding cycle in $\mathcal{B}(G)$, and then remove from $\mathcal{CB}(D, G)$ all used edges. This step is performed as long as $\mathcal{CB}(D, G)$ contains an alternating cycle.
   It is easy to see that when we enter the For Loop with a graph $\mathcal{CB}(D, G)$ containing at least one cycle, and we leave this Loop with $\mathcal{CB}(D, G)$ being non-empty, then there is at least one single gray edge with a parallel dotted edge, which allows to enter the following For Loop.

2. The For Loop 13 - 31: For each singleton extremity y^s that has been considered in the previous step (i.e. the adjacency of y_1^s has been created in $\mathcal{B}(G)$), add the second copy of this singleton extremity (i.e. y_2^s) at the right place in $\mathcal{B}(G)$, and form the corresponding alternating cycle of size 1. Lines 14 - 17 allow to concatenate adjacent singletons into a single block, and lines 18 - 30 give all the details about the appropriate modifications (insertions and removals) of edges in $\mathcal{B}(G)$ and $\mathcal{CB}(D,G)$ resulting from the insertion of a singleton at the appropriate position in G.

Algorithm Double-Distance-with-Loss(G,D)
Input: $\mathcal{CB}(D,G)$ *and the partial graph* $\mathcal{B}(G)$;
Output: The partial graph $\mathcal{B}(G)$ *completed (i.e.* $\mathcal{B}(G,D^L)$*), and an RD genome* G' *that is an extension of* G;

1. **While** $\mathcal{CB}(D,G)$ is not empty (i.e. has edges left) **Do**
2. **For** $CSize = 1$ to n **Do** ;
3. **For** $CVertex = b_1^l$ to b_n^l **Do**
4. **If** there is a cycle C of size $CSize$ beginning at $CVertex$ **Then**
5. Construct a corresponding cycle in $\mathcal{B}(G)$ (instructions 6 to 9 of Algo. Double-Distance);
6. Remove from $\mathcal{CB}(D,G)$ all edges of C;
7. **End If**
8. **End For**
9. **End For**
10. **If** $\mathcal{CB}(D,G)$ does not contain any single gray edge **Then**
11. Choose an arbitrary gray edge (x^r, y^s) with a parallel dotted edge
12. **End If**
13. **For** each single gray edge (x^r, y^s) that has a parallel dotted edge *or* the chosen edge of Line 11 **Do**
14. **If** x and y are both singletons **Then**
15. Create the block B representing the adjacency (x^r, y^s)
16. Remove from $\mathcal{CB}(D,G)$ the two vertices x^r, y^s, and their adjacent edges;
17. Replace in both $\mathcal{CB}(D,G)$ and $\mathcal{B}(G)$ the vertex $x^{\overline{r}}$ by $B^{\overline{r}}$ and the vertex $y^{\overline{s}}$ by $B^{\overline{s}}$;
18. **Otherwise** {*Among x, y, only one is a singleton*}
19. Let y be the singleton vertex;
20. Let (x^r, z^u) be the remaining black edge of $\mathcal{CB}(D,G)$ adjacent to x^r;
21. In $\mathcal{B}(G)$:
22. Remove the black edge (x^r, z^u);
23. Add the black edges (x^r, y^s) and $(y^{\overline{s}}, z^u)$;
24. Add the gray edge (x^r, y^s);
25. In $\mathcal{CB}(D,G)$:
26. Remove the black edge (x^r, z^u);
27. Remove the gray and dotted edges (x^r, y^s);
28. Remove the dotted edge adjacent to $y^{\overline{s}}$;
29. Add the black edge $(y^{\overline{s}}, z^u)$;
30. **End If**
31. **End For**
32. **End While**
33. **Return** (The genome G' deduced from the black edges of $\mathcal{B}(G)$);

Fig. 4. The notation $\overline{s}$ for $s \in \{t,h\}$ refers to the complement of s in this set. More precisely, if $s = t$ then $\overline{s} = h$ and if $s = h$ then $\overline{s} = t$. A "single gray edge" is a gray edge that has no parallel gray edge.

Unfortunately, it happens (very rarely) that the graph $\mathcal{CB}(D,G)$ obtained as an output of the For Loop 13 - 31 is acyclic, which prevents any modification by the For Loop 2 - 9,

and does not allow then to re-enter the For Loop 13 - 31, as no single gray edge exists. This is the reason of Instructions 10 - 12, allowing to remove any gray edge with a parallel dotted edge, and then proceeding with the For Loop 13 - 31.

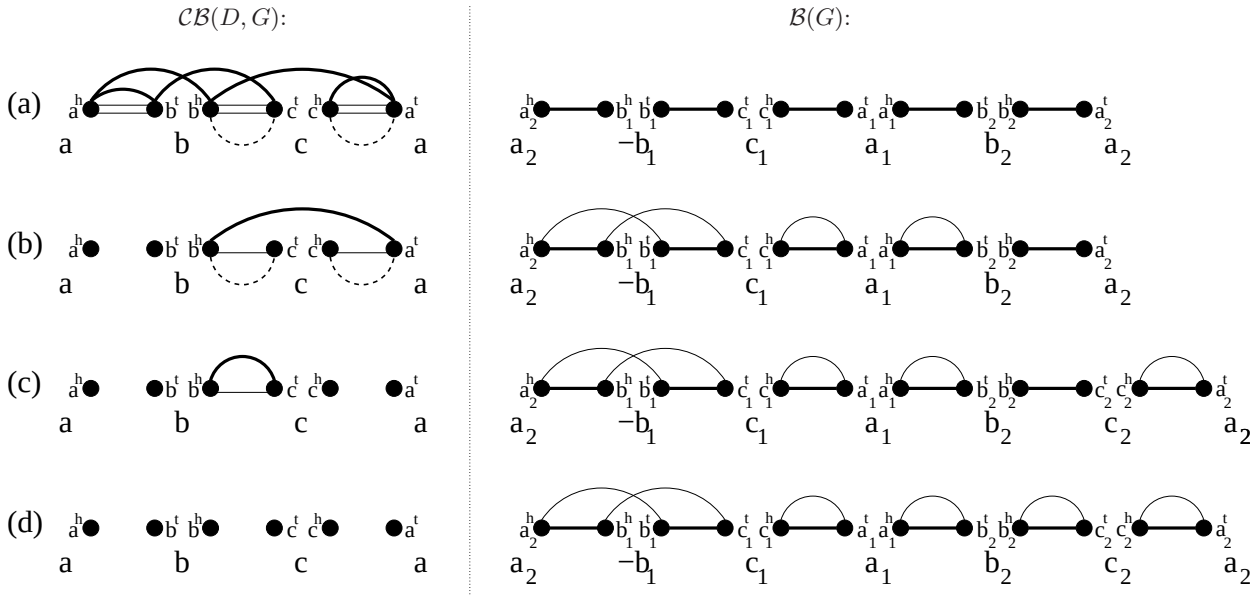

Fig. 5. An execution of Algorithm Double-Distance-with-Loss(G,D) with $D = (a\,b\,c) \oplus (a\,b\,c)$ and $G = (a\,-b\,c\,a\,b)$. The evolution of the contracted breakpoint graph $\mathcal{CB}(D,G)$ and the partial breakpoint graph $\mathcal{B}(G)$ are shown respectively on the left and right sides of the figure. (a) The initial graphs. Gene c is a singleton and, in $\mathcal{CB}(D,G)$, each of its extremities is connected by a dotted edge to its adjacent vertex in D. (b) The current graphs after executing the For Loop 2 - 9. No more alternating cycles are present in $\mathcal{CB}(D,G)$. (c) The current graphs after executing the For Loop 13 - 31. The second copy of c is inserted in $\mathcal{B}(G)$ and edges are updated in both graphs. (d) The current graphs after a second execution of the For Loop 2 - 9. As $\mathcal{CB}(D,G)$ is empty (no edges left), the algorithm stops.

Lemma 4. *The completed graph $\mathcal{B}(G, D^L)$ output by* Algorithm Double-Distance-with-Loss *satisfies:*

1. *Its set of black edges represent an extension of G.*
2. *Its set of gray edges represent, either a labelling of D or a bi-circular representation of a labelling of D.*

Proof. 1. Follows from the fact that the algorithm ends up with an empty graph $\mathcal{CB}(D,G)$ (no edges remain in $\mathcal{CB}(D,G)$). Therefore, at the end, for each singleton in G, two vertices and one black edge have been added in $\mathcal{B}(G)$, leading to a genome G' with all missing copies of G inserted.
2. Same proof as for Lemma 2 □

5.2 Multichromosomal genomes

In the case of G and D being multichromosomal genomes, the right and/or left neighbour in D of a singleton gene can be a chromosome end (represented by O in the contracted

breakpoint graph; see section 4.2). Since O is a special node that is adjacent to the endpoint genes of all the chromosomes, some modifications have to be made to Algorithm Double-Distance-with-Loss(G,D) for multichromosomal genomes.

The special node O is not considered as a singleton node in the contracted breakpoint graph, so the only part of Algorithm Double-Distance-with-Loss(G,D) that has to be changed is between lines 18 - 30. When working on the dotted edge (x^r, y^s), if $x^r = O$, then there are two possibilities: the singleton gene y is, in D, (1) adjacent to a gene and a chromosome end, or (2) adjacent to two chromosome ends (i.e. y is the only gene on a chromosome). In the first case, we can simply work on the dotted edge adjacent to $y^{\overline{s}}$, which is the dotted edge representing the other adjacency of y in D. In the second case, both y^s and $y^{\overline{s}}$ are connected to O by a dotted edge. Algorithm Double-Distance-with-Loss(G,D) can then be used on any of these two dotted edges if we skip lines 20, 22 and 26 and set $z^u = O$.

6 Results

Since the generalization of the genome halving problem to a present-day RDL genome has been proved to be an exact algorithm executing in linear time, we only test the performance of the proposed method to compute the double distance. We generated datasets through simulated evolutions between a duplicated genome D and an RD or RDL genome G for both circular and multichromosomal genomes, as follows.

Simulated datasets: We first determine n, the number of genes, and N, the number of chromosomes in D_{predup}. We also define l, the percentage of genes in D_{predup} that is lost after the WGD. Then, we generate D by applying a WGD, and a series of rearrangement and/or loss events are performed on D to obtain G. The rearrangements are simply the ones allowed by our model, namely inversions only in the case of circular genomes or inversions and translocations (including fusions and fissions) in the case of multichromosomal genomes. The number of rearrangement events, μ, is a parameter chosen prior to the data generation, and the size of each rearrangement is chosen randomly. As for the rates of rearrangement operations, we chose (Inv : Trans : Fus+Fiss) = (5 : 4 : 1) to follow the rates reported for a lineage where a WGD occurred (Gordon *et al.*, 2009).

In order to validate the distances obtained with our greedy approach, we use an exact algorithm described below.

Exact algorithm: If G is an RDL genome, we generate all possible RD genomes by reinserting the missing gene copies at all possible positions following Lemma 3. The following algorithm can be used with RD genomes. Let L (*resp.* L^*) be a complete (*resp.* partial) labelling of the gene copies of D, and $\mathcal{B}(G, D^{L^*})$ the breakpoint graph where the only defined gray edges are those adjacent to the genes of L^*. The idea is to compute a lower bound for $d_R(G, D)$ as we progressively construct L^*. More precisely, if at one step we have c cycles and p paths in $\mathcal{B}(G, D^{L^*})$, we know that the number of cycles in $\mathcal{B}(G, D^{L})$ will be at most equal to $c + p$. Thus it is possible to use the following lower bound in a branch and bound strategy: $d_R(G, D) \geq n - c - p$.

Due to the high running-time complexity of the exact method, validation with the exact distance can only be done for "simple" datasets obtained with a low number of genes, a low number of rearrangements, and a maximum of two singletons. For datasets that were too complex for the exact algorithm, we estimated the accuracy of our greedy algorithm for the double distance by comparing the inferred distance with the number of rearrangements performed between D and G in the simulated evolution.

6.1 Time efficiency

Since the running-time complexity is a function of n for the exact approach, we generated genomes containing different numbers of genes to evaluate the time efficiency of our greedy heuristic. For the exact method, n varies from 10 to 100, with an increment of 10. The parameters μ, N and l are arbitrarily fixed to 15, 4 and 0 respectively. For Algorithm Double-Distance-with-Loss(G,D), n varies from 100 to 1000 with an increment of 100 and we plotted the results for l equal to 0, 10 and 50%. With μ fixed to 15, the running-time of Algorithm Double-Distance-with-Loss(G,D) does not vary (below 0.001 seconds for all values of n). Thus, the number of rearrangements has been changed to $\mu = n$ in order to see a variation in the running-time. For each of those n values, multiple datasets were generated and the running time was averaged.

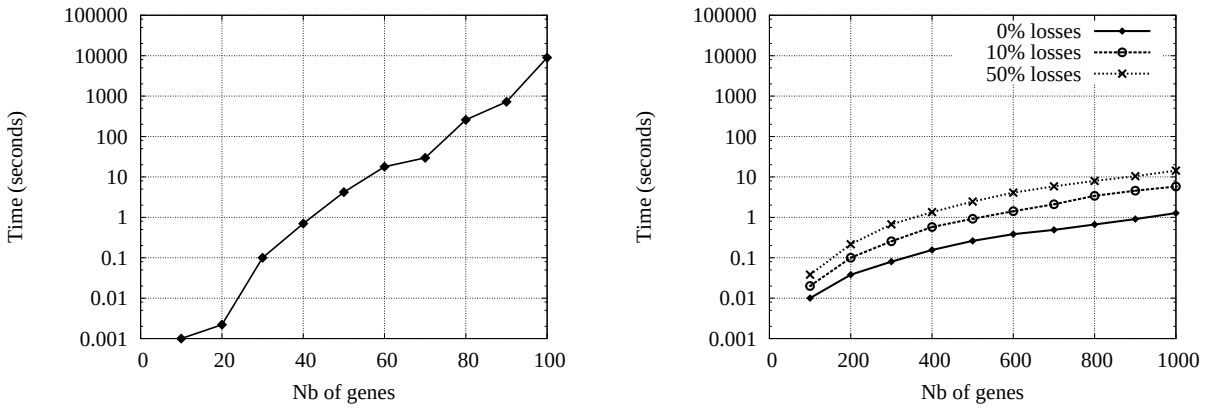

Fig. 6. Left: Running-time of the exact algorithm computing the double distance without losses between D and G with various number of genes and a fixed number of rearrangements (μ=15). Right: Running-time of Algorithm Double-Distance-with-Loss(G,D) to compute the double distance with various number of genes and rearrangements ($\mu = n$) and different gene loss percentages.

We can clearly observe the exponential running-time of the exact approach when the number of genes increases (see Figure 6 left). In contrast, Algorithm Double-Distance-with-Loss(G,D) is less limited by the genome size and more by the number of rearrangements. In Figure 6 right, we can see that even for datasets with a high number of rearrangements ($\mu = n$), the running-time, for 0% losses, remains under or close to 1 second. Obviously, the more losses there are in G, the slower is the algorithm, but the running-time remains less than the anticipated worst-time complexity.

6.2 Heuristic accuracy

Comparison with the exact approach. We now test whether Algorithm Double-Distance-with-Loss(G,D) infers an accurate rearrangement distance by comparing its results against those of the exact approach. Recall that because of the high running-time complexity of the exact approach, we can only perform this algorithm on simple datasets exhibiting low numbers of genes, rearrangements and losses. The genomes were generated with n fixed to 25, N to 4 for multichromosomal genomes, μ varying from 0 to 50 by increments of 5 and zero, one or two singletons. For each value of μ, 500 datasets were simulated. The error rate is the proportion of datasets for which the exact method found a more accurate distance than Algorithm Double-Distance-with-Loss(G,D). Results are averaged over all datasets showing a comparable number of rearrangement events.

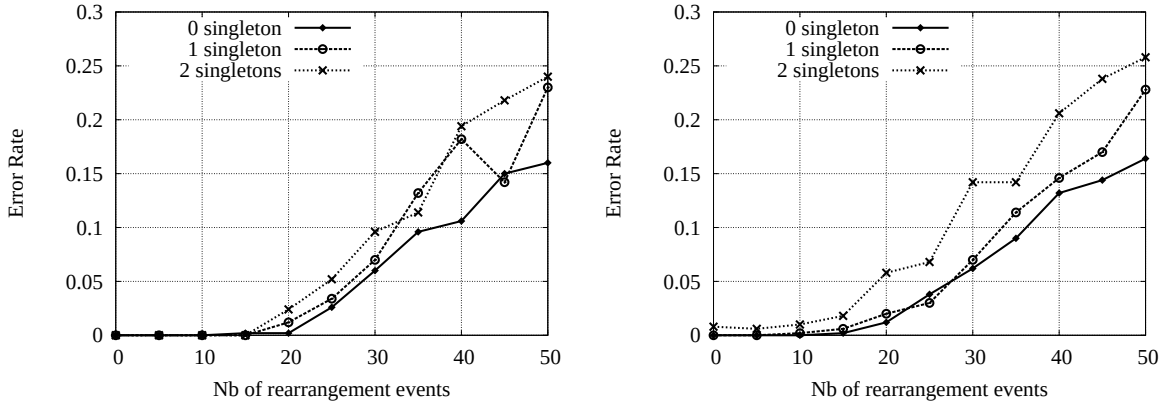

Fig. 7. Comparison of Algorithm Double-Distance-with-Loss(G,D) with the exact approach for genomes of size 50 right after the WGD, showing the error rate of the inferred rearrangement distance for circular (left) and multichromosomal genomes (right). Error rates were computed for genomes with zero, one and two singletons.

As observed in Figure 7, the error rate of Algorithm Double-Distance-with-Loss(G,D) is close to 0 when the number of rearrangements is less than 15. Moreover, the distance inferred by Algorithm Double-Distance-with-Loss(G,D) is in average really close to the optimal distance for both types of genomes (circular and multichromosomal). In fact, when the distance is not the same, it differs in average by 1 rearrangement and at most by 2 (which occurred only once in our simulations). Naturally, the error rate of Algorithm Double-Distance-with-Loss(G,D) is more apparent when the number of rearrangements and losses increases. This behavior is due to the fact that when a high number of rearrangements is performed, different cycles of equal size can be selected and a choice must be made affecting the remaining set of cycles. The presence of missing gene copies will also produce more errors because the reinsertion procedure introduces more choices. As stated before, in this experiment we seek to optimize the rearrangement distance, but we obtain similar results if we seek to optimize the DCJ distance (results not shown).

Complex datasets. As a final experiment, simulations were performed with $n = 1000$, $N = 8$ for multichromosomal genomes, μ varying from 0 to 1000, and l equal to 0, 25 and

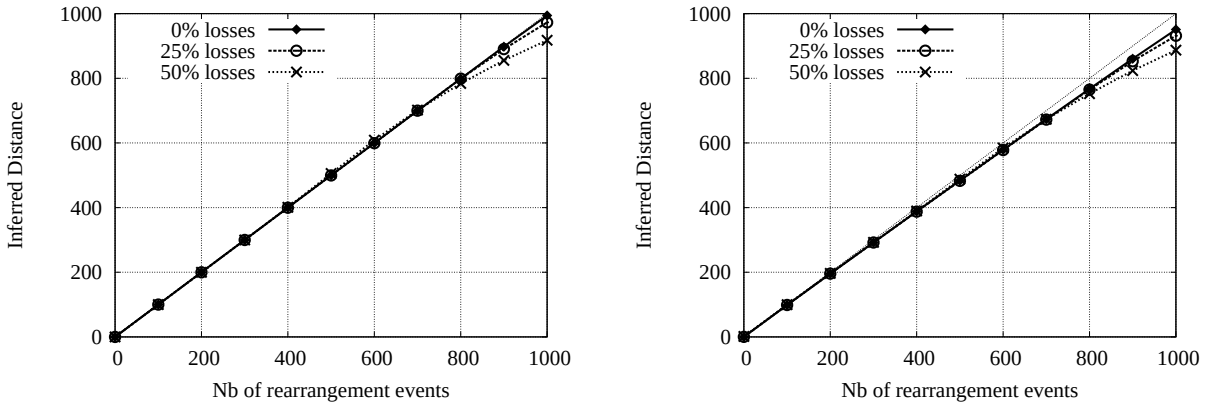

Fig. 8. Inferred rearrangement distances with complex datasets ($n = 1000$) and different gene loss percentages, for circular genomes (left) and multichromosomal genomes (right).

50%. The distances obtained with Algorithm Double-Distance-with-Loss(G,D) are compared with μ. Results shown in Figure 8 demonstrate that our method infers distances close to the number of rearrangement events performed on the original genome (for circular and multichromosomal genomes). However, when the number of rearrangement events increases, our approach underestimates that value. Notice that the more losses there are, the lesser the distance is because we reinsert the missing gene copies next to one of their adjacent genes in D. As in the comparison with the exact approach, the results are similar with the DCJ distance (not shown).

7 Conclusion

We presented a linear time algorithm to solve the genome halving problem for genomes with missing gene copies. We also presented a greedy heuristic (Algorithm Double-Distance(G,D)) to compute the distance between an RD genome G and a duplicated genome D for the rearrangement and DCJ distances. Finally, we generalized this algorithm so that genome G can be an RD or RDL genome (Algorithm Double-Distance-with-Loss(G,D)). Our experiments on simulated datasets showed that Algorithm Double-Distance-with-Loss(G,D) is time-efficient and accurate.

The proposed heuristic for the double distance could be adapted to genomes that have undergone more than one WGD, thus increasing the running-time complexity as the number of possible labellings for a gene would increase. Our algorithm could then be used for the rearrangement phylogeny problem with genomes that have evolved through one or more whole genome duplications. Indeed, this method would allow to compute distances efficiently on all branches of such a phylogeny and consequently, an algorithm for the median problem could be used on the tree.

Another interesting future work will concern the generalization of Algorithm Double-Distance-with-Loss(G,D) for genomes G and D both being RD or RDL genomes. The current approach, using the contracted breakpoint graph, can not be used directly when the genome D is not a perfectly duplicated genome.

Bibliography

Alekseyev, M. and Pevzner, P., 2007a. Colored de bruijn graphs and the genome halving problem. *IEEE/ACM Trans. Comput. Biol. Bioinformatics* 4, 98 – 107.

Alekseyev, M. and Pevzner, P., 2007b. Whole genome duplications, multi-break rearrangements, and genome halving problem. In *Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms*, 665 – 679.

Bader, D., Moret, B., and Yan, M., 2001. A linear-time algorithm for computing inversion distance between signed permutations with an experimental study. *Journal of Computational Biology* 8, 483 – 491.

Bergeron, A., Mixtacki, J., and Stoye, J., 2004. Reversal distance without hurdles and fortresses. In *Combinatorial Pattern Matching, LNCS*, volume 3109, 388 – 399.

Bergeron, A., Mixtacki, J., and Stoye, J., 2006. A unifying view of genome rearrangements. In *Algorithms in Bioinformatics, LNCS, WABI*, volume 4175, 163 – 173.

Bergeron, A., Mixtacki, J., and Stoye, J., 2009. A new linear time algorithm to compute the genomic distance via the double cut and join distance. *Theoretical Computer Science* 410, 5300 – 5316.

Blanc, G., Hokamp, K., and Wolfe, K., 2003. A recent polyploidy superimposed on older large-scale duplications in the Arabidopsis genome. *Genome Research* 13, 137 – 144.

Bowers, J., Chapman, B., Romg, J., and Paterson, A., 2003. Unravelling angiosperm genome evolution by phylogenetic analysis of chromosomal duplication events. *Nature* 422, 433 – 438.

El-Mabrouk, N. and Sankoff, D., 2003. The reconstruction of doubled genomes. *SIAM Journal on Computing* 32, 754 – 792.

Gavranović, H. and Tannier, E., 2010. Guided genome halving: probably optimal solutions provide good insights into the preduplication ancestral genome of Saccharomyces cerevisiae. In *Pacific Symposium on Biocomputing*, volume 15, 21 – 30.

Gordon, J., Byrne, K., and Wolfe, K., 2009. Additions, losses, and rearrangements on the evolutionary route from a reconstructed ancestor to the modern saccharomyces cerevisiae genome. *PloS Genetics* 5, e1000485.

Hannenhalli, S., 1995. Polynomial-time algorithm for computing translocation distance between genomes. In *LNCS*, volume 937, 162 – 176.

Hannenhalli, S. and Pevzner, P., 1995. Transforming men into mice. In *Proceedings of the IEEE 36th Annual Symposium on Foundations of Computer Science*, 581 – 592.

Hannenhalli, S. and Pevzner, P. A., 1999. Transforming cabbage into turnip (polynomial algorithm for sorting signed permutations by reversals). *JACM* 48, 1 – 27.

Kotzig, A., 1968. Moves without forbidden transitions in a graph. *Matematicky casopis* 18, 76 – 80.

P. Dehal, J. B., 2005. Two rounds of whole genome duplication in the ancestral vertebrate. *Plos Biology* 3, e314.

Pevzner, P., 1995. Dna physical mapping and alternating eulerian cycles in colored graphs. *Algorithmica* 13, 77 – 105.

Salse, J., Bolot, S., Throude, M., Jouffe, V., Piegu, B., Quraishi, U., Calcagno, T., Cooke, R., Delseny, M., and Feuillet, C., 2008. Identification and characterization of shared duplications between rice and wheat provide new insight into grass genome evolution. *The Plant Cell* 20, 11 – 24.

Soltis, D., Albert, V., Leebens-Mack, J., Bell, C., Paterson, A., Zheng, C., Sankoff, D., dePamphilis, C., Wall, P., and Soltis, P., 2009. Polyploidy and angiosperm diversification. *American Journal of Botany* 96, 336 – 348.

Tannier, E., Zheng, C., and Sankoff, D., 2009. Multichromosomal median and halving problems under different genomic distances. *BMC Bioinformatics* 10.

Tesler, G., 2002. Efficient algorithms for multichromosomal genome rearrangements. *Journal of Computer and System Sciences* 65, 587 – 609.

Yancopoulos, S., Attie, O., and Friedberg, R., 2005. Efficient sorting of genomic permutations by translocation, inversion and block interchange. *Bioinformatics* 21, 3340 – 3346.

Zheng, C., Zhu, Q., Adam, Z., and Sankoff, D., 2008a. Guided genome halving: hardness, heuristics and the history of the hemiascomycetes. ISMB, 96 – 104.

Zheng, C., Zhu, Q., and Sankoff, D., 2008b. Descendants of whole genome duplication within gene order phylogeny. *Journal of Computational Biology* 15, 947 – 964.