The Author(s) *BMC Bioinformatics* 2016, **17**(Suppl 14):413
DOI 10.1186/s12859-016-1261-9

BMC Bioinformatics

RESEARCH Open Access

Fast ancestral gene order reconstruction of genomes with unequal gene content

Pedro Feijão[1*] and Eloi Araujo[1,2]

From 14th Annual Research in Computational Molecular Biology (RECOMB) Comparative Genomics Satellite Workshop Montreal, Canada. 11-14 October 2016

Abstract

Background: During evolution, genomes are modified by large scale structural events, such as rearrangements, deletions or insertions of large blocks of DNA. Of particular interest, in order to better understand how this type of genomic evolution happens, is the reconstruction of ancestral genomes, given a phylogenetic tree with extant genomes at its leaves. One way of solving this problem is to assume a rearrangement model, such as Double Cut and Join (DCJ), and find a set of ancestral genomes that minimizes the number of events on the input tree. Since this problem is NP-hard for most rearrangement models, exact solutions are practical only for small instances, and heuristics have to be used for larger datasets. This type of approach can be called event-based. Another common approach is based on finding conserved structures between the input genomes, such as adjacencies between genes, possibly also assigning weights that indicate a measure of confidence or probability that this particular structure is present on each ancestral genome, and then finding a set of non conflicting adjacencies that optimize some given function, usually trying to maximize total weight and minimizing character changes in the tree. We call this type of methods homology-based.

Results: In previous work, we proposed an ancestral reconstruction method that combines homology- and event-based ideas, using the concept of intermediate genomes, that arise in DCJ rearrangement scenarios. This method showed better rate of correctly reconstructed adjacencies than other methods, while also being faster, since the use of intermediate genomes greatly reduces the search space. Here, we generalize the intermediate genome concept to genomes with unequal gene content, extending our method to account for gene insertions and deletions of any length. In many of the simulated datasets, our proposed method had better results than MLGO and MGRA, two state-of-the-art algorithms for ancestral reconstruction with unequal gene content, while running much faster, making it more scalable to larger datasets.

Conclusion: Studing ancestral reconstruction problems under a new light, using the concept of intermediate genomes, allows the design of very fast algorithms by greatly reducing the solution search space, while also giving very good results. The algorithms introduced in this paper were implemented in an open-source software called RINGO (ancestral Reconstruction with INtermediate GenOmes), available at https://github.com/pedrofeijao/RINGO.

Keywords: Ancestral reconstruction, Small parsimony problem, Genome rearrangement, Double-cut-and-join, InDels, Gene insertions and deletions

*Correspondence: pfeijao@cebitec.uni-bielefeld.de
[1]Technische Fakultät and CeBiTec, Universität Bielefeld, Universitätsstr. 25, 33615 Bielefeld, Germany
Full list of author information is available at the end of the article

Background

With the increased availability of assembled genomes, methods that can analyse whole genome data and reconstruct phylogenetic trees based on large sctructural variations become increasingly relevant. A problem of great interest is the reconstruction of ancestral genomes based on gene order data. This is a classical problem in the field of genome rearrangements, where a large amount of research has been devoted, and still poses many challenges. In this problem, we are given a phylogenetic tree with extant genomes at its leaves, and need to reconstruct the gene orders at the internal nodes of the tree, corresponding to ancestral genomes.

We can broadly divide approaches of solving this problem in two categories. The first is a parsimonious approach, called *event-* or *distance-based*, were a rearrangement distance is given and the aim is to find ancestral genomes that minimize the length of the tree, defined as the total number of rearrangement events on all edges of the tree. Since BPAnalysis [1], the first proposed method, which was based the breakpoint distance, many other distance-based methods were developed, with different distances, such as the reversal distance (GRAPPA [2] and MGR [3]), the double cut and join (DCJ) distance [4, 5] (PATHGROUPS [6], GASTS [7] and MGRA [8, 9]), and the single cut or join (SCJ) distance [10] (SCJ Small Phylogeny [11]), just to cite a few examples.

Another category can be called *homology-based*, where methods usually do not apply rearrangement models directly, but instead treat conserved structures between the input genomes, such as conserved adjacencies or gene clusters, as binary characters (presence and absence). These characters can also have weights that represent a confidence or probability measure, and ancestral genomes are found by optimizing an objective function that might combine factors such as maximization of weights or probabilities, and minimizing character changes in the tree. Notable examples include the pioneer InferCARs [12], as well as GapAdj [13], ANGES [14], PMAG+ [15, 16], ProCARs [17] and PhySca [18].

In our recent contribution to this field, we proposed a method that combines ideas from homology-based methods, namely adjacency weights, with the DCJ rearrangement model, by defining *intermediate genomes*, genomes that arise in optimal DCJ scenarios. We obtained promising results with this aproach, both in terms of running time and quality of the ancestral reconstruction [19].

Our previous approach, as well as most of the aforementioned methods (MGRA, GapAdj and PMAG+ are exceptions), assume that all the input genomes have the same gene content, with just one copy of each gene, which is of course not a very realistic assumption, but it does make the problem much less complicated. In recent years, the focus has been shifted to include also gene content operations, such as gene insertion and deletions. MGRA and PMAG+, for instance, are updates of previous methods that dealt only with same gene content genomes.

In this direction, in this paper we extend the intermediate genome definition to unequal gene content genomes, by using the DCJ indel model [20]. Using this model, we study theoretical in "Preliminaries", "Intermediate genomes" and "Ancestral reconstruction" sections and practical aspects in "Ancestral reconstruction algorithms" and "Results" sections. The complexity of the problem is unknown but we show that, depending on certain features of breakpoint graph we know how to solve the problem in polynomial time and in all other cases we have a FTP algorithms when we parameterize by the number c of the chromosomes. The ideas from this studying are partially used inspiring a description of a heuristic that has shown very good results regarding quality and time. In the last "Discussion" and "Conclusion" sections we discuss obtained results.

Preliminaries

Genes and genomes

A *gene* g is a sequence of two elements $g^t g^h$ or $g^h g^t$. So, $g^t g^h$ and $g^h g^t$ represent the same gene g with different orientation. We call g^h and g^t *extremities*, g^t is a *tail* and g^h is a *head* of g. Two different genes don't share extremities. If $\mathcal{G}$ is a set of genes, denote $\mathcal{G}^{\pm} = \cup_{g \in \mathcal{G}} \{g^t, g^h\}$. So, if $|\mathcal{G}| = n$, then $|\mathcal{G}^{\pm}| = 2n$.

A *chromosome* C is a sequence of genes that can be *linear* or *circular*. Denote by V_C the set of genes in C. If C is linear we represent it by adding a *telomere*, represented by the symbol $\circ$, at its endpoints. An *adjacency* in C is a pair $xy \equiv yx$ such that x and y are in $V_C^{\pm} \cup \{\circ\}$, implying that two genes are consecutive in C. If x or y is a telomere, this represents an extremity of a linear chromosome, and this type of adjacency is called a *telomeric adjacency*.

A *genome* is a set of chromosome and it is represented by the union of adjacency sets of their chromosomes. A genome is circular (linear) if all its chromosomes are circular (linear). For two genomes A and B, if $V_A = V_B$, we say that they have the *same gene content*. Conversely, if $V_A \neq V_B$, they have *unequal gene content*.

DCJ operation and the breakpoint graph

Let A be a genome, and $xy \neq vw$ two adjacencies in A. A *double cut and join operation* (DCJ) [4] on genome A is an operation that cuts two adjacencies of A and joins the free extremities in a different way. Many common rearrangement operations, like reversals and translocations, can be represented by a DCJ. Formally, a DCJ transforms A into genome $A - \{xy, vw\} \cup \{vy, xw\}$. There is also the special case of $A - \{xy\} \cup \{\circ x, \circ y\}$ and the reverse case $A - \{\circ x, \circ y\} \cup \{xy\}$, for $x, y \neq \circ$. For two genomes A and B

with same gene content, the *DCJ distance* between A and B is the minimum number $d_{\mathrm{DCJ}}(A,B)$ of DCJ operations that transforms A into B. The distance $d_{\mathrm{DCJ}}(A,B)$ can be found with the *breakpoint graph* of A and B, denoted by $\mathtt{BP}(A,B)$, which is an edge-colored graph $G = (V_A^{\pm}, A \cup B)$, that is, the vertices are the gene extremities, and edges the adjacencies of both genomes (ignoring telomeric adjacencies). Edges from A have one color and edges from B have a different color. By definition, the breakpoint graph is collection of color alternating cycles and paths. Figure 1 shows and example of a breakpoint graph.

The DCJ distance is given by

$$d_{\mathrm{DCJ}}(A,B) = n - c(A,B) + \frac{p_{even}(A,B)}{2}, \qquad (1)$$

where $n = |\mathcal{G}|$ is the number of genes, $c(A,B)$ and $p_{even}(A,B)$ are the number of cycles and the number of paths with even number of edges in $\mathtt{BP}(A,B)$ respectively, which can be found in linear time [5].

For genomes A and B with unequal gene content ($V_A \neq V_B$), extra operations are required for inserting and deleting genes in A in order to transform A into B. Genes in $V_B - V_A$ are called *unique genes* of B, and conversely $V_A - V_B$ is the set of unique genes of A. An *insertion* in A consists in inserting a contiguous sequence of genes of $V_B - V_A$ in A, and a *deletion* in A is the inverse operation, i.e, removing a contiguous sequence of genes of $V_A - V_B$ from A. An *indel* is a general expression meaning an insertion or a deletion. The *DCJ-indel distance* between A and B is the minimum number of DCJs and indels required to transform A into B, and it is denoted as $d_{\mathrm{DCJ}}^{\mathrm{ind}}(A,B)$. This distance can also be found in polynomial time, using two different approaches (Compeau [20] and Braga et al. [21]). Here, we use Compeau's approach, which is based creating *prosthetic chromosomes* [22] in each genome, formed by the unique genes of the other, creating two new genomes with the same gene content.

DCJ distance for unequal content genomes

For genomes A and B with unequal gene content, let $\mathcal{G} = V_A \cup V_B$ be the set of genes from both genomes. The breakpoint graph has a similar definition as before, changing only the vertex set, that is, $\mathtt{BP}(A,B) = (\mathcal{G}^{\pm}, A \cup B)$, which means that new types of vertices and paths will be present.

A vertex a in $\mathtt{BP}(A,B)$ is A-open if $a \notin V_A^{\pm}$, it is *B-open* if $a \notin V_B^{\pm}$ and it is *not-open* otherwise. As well as telomeres, a missing gene in A or B appears as a endpoint of a path as we can see in Fig. 2. For a path p in $\mathtt{BP}(A,B)$, we say that p is *even* if the number of edges of p is even and it is *odd* otherwise; p is *not-open* if its endpoints are both not-open; p is an *AA-path* (*BB-path*) if its endpoints are both A-open (B-open); p is an *AB-path* if it has one A-open and one B-open endpoint; p is an *A-path* (*B-path*) if it has one A-open (B-open) and one not-open endpoint. Define p_{AB} as the number of AB-path and p_A^o as the number of odd A-paths. Other notation for the number of odd/even-length paths (p_A^o, p_B^e and p_B^o) are defined analogously. When comparing two genomes A and B, a *singleton* is a circular chromosome C composed only by unique genes from one of the genomes, that is, $V_A \cap V_C = \emptyset$ or $V_B \cap V_C = \emptyset$. The number of singletons for A and B is denoted by $sing(A,B)$. Clearly, we can obtain $sing(A,B)$ in polynomial time.

A *completion* for A and B is a pair of genomes A' and B' obtained from A and B by adding *artificial singletons* (prosthetic chromosomes) in A and B in such way the $V_{A'} = V_{B'} = \mathcal{G}$.

Compeau [20] showed that the DCJ-indel distance is given by

$$d_{\mathrm{DCJ}}^{\mathrm{ind}}(A,B) = \min_{A',B'} \left\{ d_{\mathrm{DCJ}}(A',B') \right\} + sing(A,B). \qquad (2)$$

A completion A' and B' for A and B such that minimize $d_{\mathrm{DCJ}}(A',B')$ is called *optimal*.

In order to find optimal completions, consider the following definitions. For a set A, a *matching* M is a collection of disjoint subsets of A. M is a *perfect matching* of A or simply a perfect matching if the union of all sets in M is A. M is a *k-matching* if every set in M has k elements. A completion can then be seen as a perfect 2-matching of A-open vertices joined with a perfect 2-matching of B-open vertices in $\mathtt{BP}(A,B)$. In Fig. 3, we have an example of a breakpoint graph and a completion.

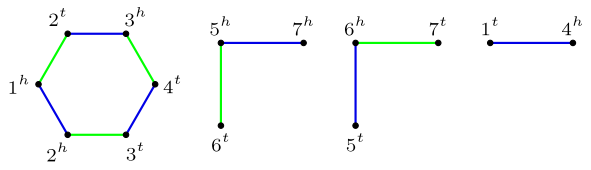

Fig. 1 Breakpoint graph $\mathtt{BP}(A,B)$ of genomes $A = \left\{\circ 1^t, 1^h 2^t, 2^h 3^t, 3^h 4^t, 4^h \circ, \circ 5^t, 5^h 6^t, 6^h 7^t, 7^h \circ\right\}$ and $B = \left\{1^h 2^h, 2^t 3^h, 3^t 4^t, 4^h 1^t, \circ 6^t, 6^h 5^t, 5^h 7^h, 7^t \circ\right\}$. Edges of A are *green*, of B are *blue*

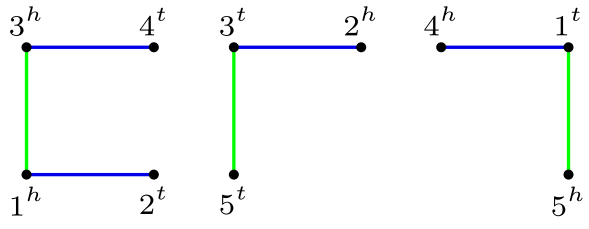

Fig. 2 Breakpoint graph $\mathtt{BP}(A,B)$ of circular genomes $G_A = (1,-3,5)$ and $G_B = (1,2,3,4)$, with adjacency sets $A = \left\{1^h 3^h, 3^t 5^t, 1^t 5^h\right\}$ and $B = \left\{1^h 2^t, 2^h 3^t, 3^h 4^t, 4^h 1^t\right\}$. Edges of A are *green*, and of B are *blue*. There is one *AA*-path and two *AB*-paths

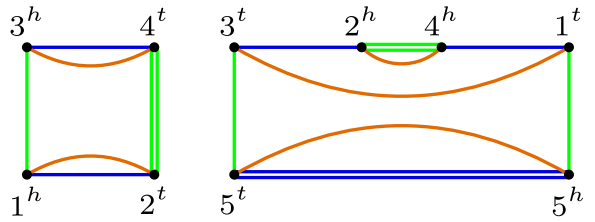

Fig. 3 The unique optimal completion C of $\mathtt{BP}(A,B)$ from Fig. 2, where A-open (B-open) vertices are joined by *green* (*blue*) double edges, closing the AA-path and linking both AB-paths, which makes $d_{\mathtt{DCJ}}^{\mathtt{ind}}(A,B) = n - c = 3$. The *orange* edges $M' = \{1^h2^t, 2^h4^h, 4^t3^h, 3^t1^t, 5^t5^h\}$ form a set of non-crossing chords covering all vertices of C. By Claim 2, M' leads to an intermediate genome. Notice that $S = \{5^t5^h\}$ is an artificial singleton, that is, a circular chromosome with only unique genes of A. Therefore, $M = M' - S = \{1^h2^t, 2^h4^h, 4^t3^h, 3^t1^t\}$, representing the circular chromosome $(1, 2, -4, -3)$, is an intermediate genome. M is present in the optimal scenario $\mathcal{S} = \{M_0 = A, M_1 = (1, -3), M_2 = (1, 2, -4, -3), M_3 = B\}$, composed by one deletion, one insertion, and one reversal (DCJ)

Let $\mathcal{C}$ be the set of all completions for A and B. If $n_A = |V_B - V_A|$ and $n_B = |V_A - V_B|$ are the number of unique genes in both genomes, then $\mathtt{BP}(A,B)$ has $2n_A$ A-open vertices and $2n_B$ B-open vertices. Since there are $(2n_A - 1)!!$ different 2-matchings for the A-open vertices and $(2n_B - 1)!!$ different 2-matchings for the B-open vertices, we have that

$$|\mathcal{C}| = (2n_A - 1)!! \cdot (2n_B - 1)!!, \tag{3}$$

which is exponential on the number of unique genes of A and B. However, an optimal completion can be found in polynomial time, which implies, since we can obtain $sing(A,B)$ in polynomial time, that (2) can also be computed in polynomial time [20].

Enumerating all optimal completions

The intuition behind finding an optimal completion is that Eq. (2) is minimized when the number of cycles and even paths of $\mathtt{BP}(A,B)$ is maximized. This guides the linking of components with A- and B-open vertices into creating as many cycles and even paths as possible. Therefore, AA-paths and BB-paths are always closed directly by linking their own A- or B-open vertices, since each becomes a cycle. AB-paths are usually linked in pairs, creating one cycle per pair. A-paths are also paired, ideally two paths with opposing parity, since this creates an even pair, and similarly for the B-paths. In many cases, this simple strategy is already enough to find optimal completions. Unfortunately, this can get more complicated when in some cases a triplet of components, specifically one A-path, one AB-path and one B-path can be linked in an optimal completion. In the following, we enumerate the space of all optimal completions, summarizing the results introduced by Compeau [20].

Let $\mathcal{C}^*$ be the space of all optimal completions for A and B. Using results from [20] we define a hypergraph $\mathcal{H}$ representing $\mathcal{C}^*$. The vertices represent components of the breakpoint graph, and hyperedges of $\mathcal{H}$ represent linked components that form a new component in a completion. In any completion, components without open vertices are not linked with other components. Also, AA-paths (BB-paths) become cycles by adding an edge between the two A-open (B-open) vertices in any optimal completion. Therefore, these components are not in $\mathcal{H}$.

In the following definitions, we use the notation of Cartesian product, but exclude pairs of identical elements, since a component can not be linked to itself. Let V be the set of vertices of $\mathcal{H}$. V is the union of the following sets, representing components of the $\mathtt{BP}(A,B)$: Λ^o, Λ^e, Υ, Γ^o and Γ^e, the set of odd A-paths, even A-paths, AB paths, odd B-paths and even B-paths respectively. Consider the set of hyperedges of $\mathcal{H}$ that is the union of sets $T_1 = \Lambda^o \times \Lambda^e$; $T_2 = \Gamma^o \times \Gamma^e$; $T_3 = \Upsilon \times \Upsilon$; $T_4 = \Lambda^o \times \Lambda^o$; $T_5 = \Lambda^e \times \Lambda^e$; $T_6 = \Gamma^o \times \Gamma^o$; $T_7 = \Gamma^e \times \Gamma^e$; $T_8 = \Lambda^o \times \Upsilon \times \Gamma^o$; $T_9 = \Lambda^o \times \Upsilon \times \Gamma^e$; $T_{10} = \Lambda^e \times \Upsilon \times \Gamma^o$; $T_{11} = \Lambda^e \times \Upsilon \times \Gamma^e$.

1. if p_{AB} is even, $p_A^o \leq p_A^e$ and $p_B^o \geq p_B^e$, an optimal completion is any perfect matching using hyperedges in $T_1 \cup T_2 \cup T_3 \cup T_5 \cup T_6$.
2. if p_{AB} is even, and $p_A^o \geq p_A^e$ and $p_B^o \leq p_B^e$, an optimal completion is any perfect matching using hyperedges in $T_1 \cup T_2 \cup T_3 \cup T_4 \cup T_7$.
3. if p_{AB} is odd, and $p_A^o \leq p_A^e$ and $p_B^o \geq p_B^e$, an optimal completion is any perfect matching using only one hyperedge in T_{10} and hyperedges $T_1 \cup T_2 \cup T_3 \cup T_5 \cup T_6$.
4. if p_{AB} is odd and $p_A^o \geq p_A^e$ and $p_B^o \leq p_B^e$, an optimal completion is any perfect matching using only one hyperedge in T_9 and hyperedges in $T_1 \cup T_2 \cup T_3 \cup T_4 \cup T_7$.
5. if $p_A^o < p_A^e$ and $p_B^o < p_B^e$, an optimal completion is any perfect matching using hyperedges in $T_1 \cup T_2 \cup T_3 \cup T_5 \cup T_7 \cup T_{11}$.
6. if $p_A^o > p_A^e$ and $p_B^o > p_B^e$, an optimal completion is any perfect matching using hyperedges in $T_1 \cup T_2 \cup T_3 \cup T_4 \cup T_6 \cup T_8$.

Claim 1 *Let $n = |\mathcal{G}|$ and c the sum of the number of chromosomes in A and B. Then, there are at most $((2c)!)^2 \cdot O(n^c)$ different ways to choose a 3-matching in an optimal solution in $\mathcal{H}$.*

Proof Each set with three components represents one A-path, one AB-path and one B-path. Since each A-path and B-path has one telomere each and we have

c chromosomes, there are $i \leq c$ triples in a solution. Considering that $i = 0, \ldots, c$, there are at most

$$\binom{n}{0} + \binom{n}{1} + \ldots + \binom{n}{c} = O(n^c)$$

different ways to choose a set of AB-path to obtain triples in a optimal completion.

Once chosen a set of AB-path and we have to choose no more than $2c$ A-path and no more than $2c$ B-path. So, we have a total of no more than $((2c)!)^2 \cdot O(n^c)$ different ways to choose a 3-matching in an optimal solution in $\mathcal{H}$. □

Methods

In our previous approach, we used the concept of intermediate genomes to propose a new ancestral reconstruction method, in the context of genomes with same gene content [19]. We extend this approach here to genomes with unequal gene content, by dealing with gene insertion and deletion events.

In the following sections, every key aspect of the proposed method will be explained. Basic properties of intermediate genomes are described, based on existing results, and new properties for the case of genomes with unequal gene content are shown. Then, we show how the classic problems of small phylogeny and genome median can be reformulated adding intermediate genome constraints, also proposing a new problem, the *Maximum Weight Intermediate Genome*, that is at the core of our method.

Practical aspects such as estimating tree branch lengths and finding adjacency weights at each internal node of the tree are described. Finally, we describe the main algorithm, that iteratively reconstructs ancestors at internal nodes in a bottom-up approach, by using intermediate genome properties and adjacency weights.

Intermediate genomes

In this section, we review some key combinatorial properties of intermediate genomes and extend the definition for genomes with unequal gene content, assuming that gene deletions and duplications have occurred.

Basic properties of intermediate genomes

An *optimal DCJ scenario* between two genomes A and B is an ordered list of genomes $\mathcal{S} = (M_0, M_1, \ldots, M_k)$ where $k = d_{\text{DCJ}}(A, B)$, $A = M_0$, $M_k = B$ and M_i can be obtained from M_{i-1} by applying a DCJ operation, for $i = 1, \ldots, k$. Any genome $M_i \in \mathcal{S}$ is called an *intermediate genome* of A and B.

Optimal DCJ scenarios can be found by dealing with each component in the breakpoint graph independently. A scenario that follows this strategy will be called *independent component scenario*. There are also optimal scenarios where a DCJ operations may act on two different components, specifically two even paths, but these are very rare [23]. Currently, we ignore recombination of even paths, in order to simplify the combinatorial analysis. In other context, a method was proposed to include this type of events [24], and we plan to add a similar extension to our framework as well.

Given breakpoint graph $\texttt{BP}(A, B)$, a *circular breakpoint graph* can be obtained by transforming the paths into cycles as follows: i) to for each even path, add a new vertex $\circ$ and connect both extremities of the path to this new vertex; ii) for each odd path, add two new vertices $\circ_1$ and $\circ_2$ with and edge connecting both, and connect each extremity of the path to a different new vertex. This circular version of the breakpoint graph is composed only of cycles and it preserves the DCJ distance equation given by Eq. (1), adjusting n to $n + k/2$ to account for the extra number of k artificial vertices added [19].

The main property of intermediate genomes on independent component scenarios is given by the following theorem:

Theorem 1 ([19]) *Given genomes A and B with the same set of genes, a genome M is an intermediate genome of A and B in an independent component scenario if and only if the edges of M are non-crossing chords in the cycles of the circular* $\texttt{BP}(A, B)$*, and M covers all vertices of* $\texttt{BP}(A, B)$.

In practice this makes it very easy to verify if a given genome is an intermediate genome, or even to create one given a choice of possible adjacencies, a key aspect of our ancestral reconstruction algorithm.

Intermediate genomes for DCJ InDel scenarios

The definition of intermediate genomes for genomes with unequal content is the same as the original one, just considering optimal *DCJ-indel* scenarios, instead of DCJ only scenarios.

It is somewhat straightforward to extend the definition of intermediate genomes, using the DCJ-indel model of Compeau [20] and the concept of optimal completions. Given an optimal completion C of a breakpoint graph $\texttt{BP}(A, B)$, we can create a *circular completion* by applying the operation of transforming all paths into cycles, similarly as done above to a breakpoint graph for genomes with the same gene content. After a circular completion is found, the resulting breakpoint graph is essentially the same as a breakpoint graph for genomes with same gene content. Therefore, we extend the results of Theorem 1 in the following claim.

Claim 2 *Given genomes A and B, a circular optimal completion C of* $\texttt{BP}(A, B)$*, and a set* M' *of non-crossing chords in the cycles of C, covering all vertices of C, the genome* $M = M' - S$*, where S is the set of the adjacencies of all singletons of* M' *in respect to A and B, is an*

intermediate genome of A and B. Conversely, if M is an intermediate genome of A and B, there exists a circular optimal completion C of $\texttt{BP}(A,B)$ *and a set of adjacencies S, where* $M' = M \cup S$ *is a set of non-crossing chords in the cycles of C, covering all of its vertices, and S forms the set of adjacencies of singletons of* M' *in respect to A and B.*

Note that this result is general, also applicable for the same gene content genomes, since in this case we can consider that the breakpoint graph is directly an unique and optimal completion, and the set of singletons is always an empty set. Figures 2 and 3 show an example of an optimal completion and an intermediate genome.

Ancestral reconstruction

In this section we explore how the concept of intermediate genomes can be used for ancestral reconstruction of gene orders.

In the context of rearrangement distance models, the ancestral reconstruction problem can be stated as: considering a measure distance $d(A,B)$ between genomes A and B, given a tree T with n extant genomes at the leaves, find a labeling of the internal nodes corresponding to ancestral genomes, such that the total length of the tree, defined as the sum of all distances $d(.)$ on the edges, is minimized. This is usually called the *small phylogeny problem*.

The simplest instance of this problem happens when only three genomes A, B and C are given, and we want to find a genome M minimizing $d(A,M)+d(B,M)+d(C,M)$, the *genome median problem*. Despite being NP-hard for DCJ and many other models, it is well studied and many exact and heuristic methods have been proposed [25, 26],

Here we investigate new definitions of both the median problem and the small phylogeny problem that include intermediate genomes, motivated by the fact that some studies show that purely minimizing the tree length (or finding median genomes) might not be the best option for ancestral reconstruction [27].

Let $\texttt{IG}(A,B)$ represent the set of intermediate genomes between A and B. For the median problem, we can use the fact that $d(A,M) + d(B,M) = d(A,B)$ if M is in $\texttt{IG}(A,B)$ to give the following definition.

Problem 1 (Intermediate Genome Median) *Given two genomes A and B, and an outgroup genome C, find an* $M \in \texttt{IG}(A,B)$ *minimizing* $d(C,M)$.

Problem 2 (Intermediate Genome Small Phylogeny) *Given a rooted binary tree T with n extant genomes at the leaves, find a labeling of the internal nodes such that the tree length is minimized, and each genome on an internal node is an intermediate genome of its children.*

Theorem 2 *The DCJ Intermediate Genome Median is NP-hard.*

Proof A *balanced bicoloured graph G* is a graph where each vertex has the same number of red and blue incident edges, all vertices have degree two or four, and there is no cycle formed by edges of the same colour. An *alternating cycle* in G is a cycle where red and blue edges are alternating. The *breakpoint graph decomposition problem* (BGD) is to find a maximum number of edge-disjoint alternating cycles of G. This problem is NP-hard [28].

A proof for this theorem can be derived directly from the original proof of NP-hardness of the DCJ median problem, where a reduction from BGD is performed [29]. In that proof, from an instance of the BGD with $G = (\mathcal{V}, \mathcal{B} \cup \mathcal{R})$, where $\mathcal{V}$ is a set of vertices and $\mathcal{B}$ and $\mathcal{R}$ are sets of blue and red edges, the genomes A, B and C on $\mathcal{G}$ are constructed. The set $\mathcal{G}$ contains one gene X for each degree 2 vertex and two genes X and $\bar{X}$ for each degree 4 vertex X of G. The set of adjacencies of A is $\{X^h X^t : X \in \mathcal{G}\}$. The set of adjacencies of B is $\{X^h\bar{X}^t, X^t\bar{X}^h : X \in \mathcal{V}$ and degree of X is $4\} \cup \{X^h X^t : X \in \mathcal{V}$ and degree of X is $2\}$. The set of adjacencies of C is defined adding to C an adjacency in $\{X^hY^h, X^h\bar{Y}^h, \bar{X}^hY^h, \bar{X}^h\bar{Y}^h\}$ for each $XY \in \mathcal{B}$, and an adjacency in $\{X^tY^t, X^t\bar{Y}^t, \bar{X}^tY^t, \bar{X}^t\bar{Y}^t\}$ for each $XY \in \mathcal{R}$. Figure 4 shows an example of the construction of genomes from a balanced bicoloured graph.

Defining A, B, C this way, there is a median $M \subseteq A \cup B$ that indicates the number of alternating cycles we have in a maximum edge-disjoint alternating cycle of G [29].

As a consequence of $M \subseteq A \cup B$, we have that $M \in \texttt{IG}(A,B)$ [19]. So, $M \in \texttt{IG}(A,B)$ and minimizes $d_{\text{DCJ}}(M,A) + d_{\text{DCJ}}(M,B) + d_{\text{DCJ}}(M,C)$ solving both the DCJ median for this specific instance and the BGD for the general case. It follows, since we can construct genomes

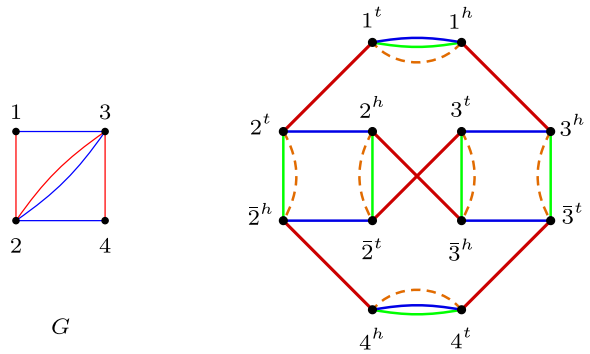

Fig. 4 Given a balanced bicoloured graph G (at *left*), a breakpoint graph is constructed (at *right*), with genomes $A = \{1^t1^h, 2^t2^h, \bar{2}^t\bar{2}^h, 3^t3^h, \bar{3}^t\bar{3}^h, 4^t4^h\}$ (in *blue*), $B = \{1^t1^h, 2^t\bar{2}^h, \bar{2}^t2^h, 3^t\bar{3}^h, \bar{3}^t3^h, 4^t4^h\}$ (in *green*) and $C = \{1^t2^t, \bar{2}^t3^t, \bar{3}^t4^t, 1^h3^h, 2^h\bar{3}^h, \bar{2}^h4^h\}$ (in *red*). In this example, $M = B \subseteq A \cup B$ is a median (in *dashed orange edges*)

A, B, C in polynomial time and BGD is NP-hard, that DCJ Intermediate Genome Median is also NP-hard. □

Since the median and consequently the small phylogeny problem are NP-hard also in their intermediate genomes formulation, we propose an approach that combines adjacency weighting methods that are common in adjacency-based algorithms, with the DCJ rearrangement model in the form of intermediate genomes, but without the need to explicitly consider searching for rearrangement events and/or scenarios, which makes the problem much more tractable.

Maximum weight intermediate genome

Problem 3 (Maximum Weight Intermediate Genome) *Given genomes A and B on set of genes $\mathcal{G}$ and a set of adjacency weights $W = \{w_{ij} \mid ij \in \mathcal{G}^{\pm} \times \mathcal{G}^{\pm}\}$, find a genome M such that*

$$M = \underset{M \in \mathtt{IG}(A,B)}{\arg\max} \sum \delta_{ij}(M) \cdot w_{ij}$$

where $\delta_{ij}(M) = 1$ if $ij \in M$, 0 otherwise.

If the genomes A and B have the same genes, this problem can be solved in polynomial time, since finding a maximum weight set of non-crossing chords in a cycle is equivalent to finding a maximum weight independent set on a circle graph (MWIS) [30]. Therefore, it is possible to find an optimal $M \in \mathtt{IG}(A, B)$ by solving a MWIS for each component of $\mathtt{BP}(A, B)$.

If A and B have different gene sets, the problem becomes much harder, since each completion of $\mathtt{BP}(A, B)$ will give rise to different components and therefore different solutions for the individual MWIS. The naive method of finding the maximum weight IG for all completions is impractical, since, according Eq. (3), there is an exponential number of completions.

A strategy to solve Problem 3 is to search a perfect matching in the graph $\mathcal{H}$ that represents all possible optimal completions in C^*, where the weight of each hyperedge is the weight obtained by solving the MWIS for the correspondent component.

Edmonds [31] shows that the maximum weighted perfect 2-matching problem can be solved in polynomial time. It follows directly from the $\mathcal{H}$ representation that

Claim 3 *Suppose that p_{AB} is even, and $p_A^o \leq p_A^e$ and $p_B^o \geq p_B^e$ or $p_A^o \geq p_A^e$ and $p_B^o \leq p_B^e$. Then, the Maximum Weight Intermediate Genome problem can be solved polynomially.*

Moreover, we have that

Claim 4 *Suppose that p_{AB} is odd, and $p_A^o \leq p_A^e$ and $p_B^o \geq p_B^e$ or $p_A^o \geq p_A^e$ and $p_B^o \leq p_B^e$. Then, the Maximum Weight Intermediate Genome problem can also be solved polynomially.*

Proof Since p_{AB} is odd, $p_A^o \leq p_A^e$ and $p_B^o \geq p_B^e$ or $p_A^o \geq p_A^e$ and $p_B^o \leq p_B^e$, there is exactly one hyperedge with 3 elements. The number of hyperedges with 3 elements in $\mathcal{H}$ is limited by $\binom{n}{3} = O(n^3)$. Once one hyperedge with 3 elements is removed, according to Claim 3, finding a perfect 2-matching in the remaining vertices of the graph is polynomial. Therefore, an optimal solution is found in polynomial time by repeating this for all $O(n^3)$ hyperedges with three elements and choosing the solution with maximum weight. □

Unfortunately, the cases where $p_A^o < p_A^e$ and $p_B^o < p_B^e$, or $p_A^o > p_A^e$ and $p_B^o > p_B^e$ are most likely NP-hard, due to the presence of up to c (number of chromosomes) triple-matchings in optimal completions, as opposed to just one. This means that the complexity of the Maximum Weight Intermediate Genome problem is still open for the general case. However, considering that the number of chromosomes is constant, we have the following interesting result from the theoretical point of view.

Theorem 3 *There is a polynomial time FPT algorithm for the Maximum Weight Intermediate Genome problem when it is parameterized by the number c of chromosomes.*

Proof Claim 3 and 4 guarantee that there is a polynomial time algorithm if $p_A^o \leq p_A^e$ and $p_B^o \geq p_B^e$ or $p_A^o \geq p_A^e$ and $p_B^o \leq p_B^e$. If $p_A^o < p_A^e$ and $p_B^o < p_B^e$, or $p_A^o > p_A^e$ and $p_B^o > p_B^e$, using a polynomial algorithm for maximum weighted perfect 2-matching and claim 1, we have a FTP algorithm with parameter c. □

Ancestral reconstruction algorithms

In this section we describe the practical algorithms that were used for our proposed ancestral reconstruction method. First, we discuss how adjacency weights can be obtained. Then, how these weights are used by a heuristic to find candidate intermediate genomes for the ancestral nodes of the input tree.

Finding adjacency weights

Adjacency weights were obtained using two methods. First, using the software DeClone [32], that randomly samples evolutionary scenarios and assign weights based on how often an adjacency is present on those scenarios. The parsimony score of a given scenario is determined by the number of gains/losses of adjacencies along the branches of the tree. DeClone samples scenarios depending on a parameter kT. When kT is close to zero, only

optimal scenarios (with minimal parsimony score) are sampled, and as kT increases, sub-optimal scenarios have a higher chance of being sampled. The weights for each adjacency at each internal node depend on how often this adjacency is observed at this internal node. Typical values include $kT = 0.1$ for sampling optimal scenarios almost exclusively, and $kT = 1$ for a more balanced distribution including non-optimal scenarios [18].

We also propose a second way of deriving adjacency weights, inspired by the weighting scheme used in InferCARs [12]. Given a rooted phylogenetic tree T, let $w_\alpha(ij)$ denote the weight of adjacency ij at a node α. Weights in all nodes are recursively defined by

$$w_\alpha(ij) = \frac{D_L \cdot w_R(ij) + D_R \cdot w_L(ij)}{D_L + D_R} \quad (4)$$

where D_L (D_R) is the distance to the left (right) child of α, and $w_L(ij)$ ($w_R(ij)$) is the weight of ij at the left (right) child of α. For leaf nodes, $w_\alpha(ij) = 1$ if the adjacency is present and $w_\alpha(ij) = 0$ otherwise.

To define the weights in our approach, we proceed as follows: for every internal node α, let γ be the the parent node of α, and create a new tree T' by removing from T the subtree defined by the node α. Then, remove the original root and reroot T' at the node γ and use the recurrence equation above to find $w_\gamma(ij)$ for all adjacencies ij. The adjacency weights for α are then $w_\alpha(ij) = w_\gamma(ij)$ for each ij. An example is shown on Fig. 5.

The motivation for using this weighting algorithm is that, while reconstructing a particular node α, the information from the leaves is given in the form of the breakpoint graph, while the weights that will guide the reconstruction of the intermediate genome should reflect information from the "other side" of the tree. The experimental results show, somewhat surprisingly, that this simple weighting scheme not only is faster than DeClone, but also increases the quality of the reconstruction.

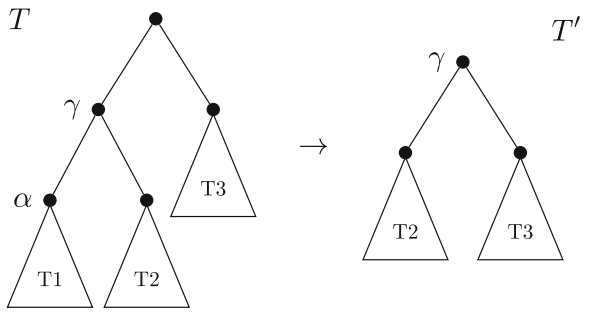

Fig. 5 To find adjacency weights for node α on a tree T, a new tree T' is created where α and its subtree $T1$ are removed, and T' is rerooted at γ, the parent node of α. Then, Eq. (4) is applied to find weights for γ, which are then assigned to node α on the original tree T

Estimating branch lengths

For the InferCARs weight algorithm, branch lenghts are needed. Since branch lengths are not always available, we tested how different estimation methods might impact the adjacency weights and consequentely the ancestral reconstruction. For this, we implemented two classic methods of branch length estimation, Minimum Evolution [33] and Fitch-Margoliash Least Squares [34], briefly described in the following.

Let T be an unrooted tree with k leafs and $n = 2k - 3$ edges, with edge lengths denoted by the vector $w = (w_1, \ldots, w_n)$. Let M be a $m \times n$ matrix, where $m = \binom{k}{2}$. Each column of M represents a branch length, and each row a pairwise comparison between two leafs of T. An element m_{ij} of M is 1 if the edge j is present in the tree path from the two leafs being compared, and $m_{ij} = 0$ otherwise. Let $d = (d_1, \ldots, d_m)$ be a vector where each element d_i stores the DCJ-Indel distance of the two genomes being compared on this row i. Therefore, for $k > 3$ leafs, we have $m > n$ and $Mw = d$ is an over-determined equation system. Then, as proposed by Fitch and Margoliash [34], a good candidate for the edge weights is the vector w^* that minimizes the least squares error, that is,

$$w^* = \arg\min_{w \in \mathbb{R}^n} \|Mw - d\|^2.$$

Another idea is to assume that the pairwise distances in d are a lower-bound for the tree traversal distances, and find edge lengths that satisfy this restriction and have minimum total sum. This method, called Minimum Evolution by Waterman et al. [33], is based on solving the following Linear Programming formulation:

$$\begin{aligned} \text{minimize} \quad & \sum_{i=1}^{n} w_i \\ \text{subject to} \quad & Mw \geq d \\ & w_i \geq 0, \quad i = 0, \ldots, n \end{aligned}$$

An algorithm for the IG-Indel small parsimony problem

Given a rooted phylogenetic tree with genomes at the leaves and a set of adjacency weights, our method works in a bottom-up fashion, by choosing two leaves with the same parent, reconstructing the ancestor at this parent node, and labeling this current node as a leaf, until the root of the tree is reconstructed.

At each node being reconstructed, given the two children genomes and a set of adjacency weights, a heuristic for the Maximum Weight Intermediate Genome (MWIG) problem is called, which tries to quickly find an optimal completion with high adjacency weight.

To do that, we build the hypergraph $\mathcal{H}$ representing all optimal completions $\mathcal{C}^*$, but ignore triple matchings, focusing only on 2-matchings present in optimal completions, as given by the sets T_i, $i = 1, \ldots, 7$. The weight

of an edge in $\mathcal{H}$ is given by the solution of a MWIS on the component correspoding to the given edge. If p_{AB} is even, there is a perfect matching in $\mathcal{H}$ corresponding to an optimal completion. We find a maximum weight perfect matching using `BlossomV` [35]. Then, from each MWIS solution for the matched components, we get adjacencies to build a genome G that is a high weight solution for the MWIG. If p_{AB} is odd, we could use Claim 4 strategy of removing every possible triplet of $\mathcal{H}$ and solving the even case as described, picking then the combination with highest weight. Since the number of triplets can be very high, we chose to solve this in a faster way by adding three dummy nodes v_a, v_b, and v_{ab} to $\mathcal{H}$, connected with zero weights to all vertices corresponding to A-, B- and AB-paths, respectively, artificially transforming $\mathcal{H}$ in a even p_{AB} case, and then finding a maximum weight perfect matching on $\mathcal{H}$. The three components that are matched to the dummy nodes are then combined, and a MWIS is solved for this triplet.

A pseudocode of the proposed method, which we call IG_SMALL_PHYLOGENY, is given at Algorithm 1.

Algorithm 1 The main function IG_SMALL_PHYLOGENY receives a tree T, an extant genomes list $\mathcal{G}$, and a set $\mathcal{W}$ of adjacency weights for each internal node T, and returns a new list with the reconstructed ancestral genomes added. In a bottom-up approach, it chooses the two closest sibling leaves and reconstructs the ancestral parent node calling MAX_WEIGHT_IG, that receives the two leaves and the ancestral adjacency weights, and returns a high weight intermediate genome. For cycles, paths, AA-paths and BB-paths, solving a MWIS finds a maximum weight set of non-crossing adjacencies. Then, for the other open components, we build the graph $\mathcal{H}$ by finding all pairs of each type of component AB-, A- and B-paths according to sets $T_1, \ldots, T_7$, with the weight of each matching given by solving a MWIS on the paired components. If p_{AB} is odd, we have to complete $\mathcal{H}$ with dummy vertices and edges to guarantee there exists a perfect matching. Here, Λ^+ and Λ^- represent the sets of odd and even A-paths, where Λ^+ is the higher cardinality set, and Λ^- the smaller. Γ^+ and Γ^- are defined similarly. A Maximum Weight Perfect Matching is then solved on $\mathcal{H}$. If p_{AB} is odd, we form the triplet given by the components that were matched with the dummy vertices and solve another MWIS. An intermediate genome is built by collecting all adjacencies from the MWIS solutions from the matched componenents, and then removing artificial singletons.

```
1:  function IG_SMALL_PHYLOGENY(T, 𝒢, 𝒲)
2:      while |T| > 2 do
3:          ℓ1, ℓ2 ← Find closest siblings of T
4:          p ← get parent node of (ℓ1, ℓ2)
5:          𝒢[p] ← MAX_WEIGHT_IG(𝒢[ℓ1], 𝒢[ℓ2], 𝒲[p])
6:          Remove n1 and n2 of T          ▷ p becomes a leaf.
7:      return 𝒢
8:  function MAX_WEIGHT_IG(A, B, W)
9:      bp ← BP(A, B)
10:     for (i, j) ∈ ∪_{i=1}^{7} T_i do          ▷ edges (i, j) of ℋ
11:         Find w_ij solving a MWIS on component i ∪ j
12:     if p_AB is odd then                      ▷ add dummy nodes
13:         Add edge (v_a, i) for i ∈ Λ+, with w_{v_a,i} = 0
14:         Add edge (v_b, j) for j ∈ Γ+, with w_{v_b,i} = 0
15:         Add edge (v_ab, k) for k ∈ ϒ, with w_{v_ab,i} = 0
16:     M ← Maximum Weight Perfect Matching on ℋ
17:     if p_AB is odd then
18:         Find (v_a, i*), (v_b, j*), (v_ab, k*) ∈ M
19:         Solve MWIS for i* ∪ k* ∪ j* and add to M
20:     From the MWIS solutions in M, build genome G
21:     Remove artificial singletons from G
22:     return G
```

Results

We implemented our algorithms in a software called RINGO (ancestral Reconstruction with INtermediate GenOmes), available at https://github.com/pedrofeijao/RINGO. We created several simulated datasets to test our proposed algorithms and compare with other existing approaches. RINGO was ran with DeClone weights for $kT = 0.1$, $kT = 0.4$ and $kT = 0.8$, and also our custom weight algorithm. For the custom weights, we used the branch lengths given from the simulations, and also tested with branch length estimates given by Minimum Evolution and Least Squares.

We compared RINGO with two other methods for ancestral reconstruction of unequal content genes, MGRA [9] and PMAG+ [15], implemented in the tool MLGO [16].

Simulated datasets

The simulated datasets were created using a similar procedure as in [19], with a few extra parameters to include indel events. A birth-death model with a birth rate of 0.001 and a death rate of 0 generates an ultrametric tree with $N = 12$ leaves, and the branch lengths are disturbed by multiplying by e^d, where d is a real number uniformly chosen from the interval $[-2, +2]$. The branch lengths are then rescaled so the tree has a diameter $D \in \{0.5n, 1n, 1.5n, 2n, 2.5n\}$, where $n = 1000$ is the number of genes, and the diameter is the maximum distance between two leaves.

The root node is labeled with an unichromosomal genome with 1000 genes, and evolution is simulated along the edges by performing a number of random events defined by the edge length. Events are chosen randomly between reversals, deletions and insertions, with probability $1 - P$, $P/2$ and $P/2$ respectively, with $P \in$

$\{0, 0.2, 0.4, 0.6\}$. The length of an indel is sampled uniformly from $[1, I]$, with $I \in \{1, 5\}$. Although the expected size of the leaf genomes is 1000, there is not guarantee that genomes will have the same size. For each combination of D, P and I, we generated 20 datasets.

Discussion

All algorithms were compared in terms of quality of the reconstruction, DCJ distance to the correct ancestral genomes, and running time.

The quality results of all simulations are summarized on Fig. 6. Each column represents the average results of RINGO, MLGO and MGRA on each dataset, showing the average number of true positives and false positives, when comparing the adjacencies of the simulated and the reconstructed genomes, in all internal nodes of a given tree. More detailed results are given on Table 1, that also shows all variations of the RINGO algorithms.

In datasets with small amount of evolution ($D = 0.5$ and $D = 1$), specially with unitary indels ($I = 1$), MGRA has a slightly better quality than the two others. But, as soon as the rearrangement rate increases, MGRA quality decreases rapidly, while RINGO and MLGO quality seems to decrease in a slower, somewhat linear rate.

At higher rates ($D > 1$), MLGO has a slightly higher number of true positives, but at the cost of a much higher number of false positives. RINGO is a more conservative method, with the smallest number of false positives in all datasets.

When comparing the datasets with $I = 1$ versus $I = 5$, we notice a decrease in quality for all algorithms for the larger indels, but MGRA has a slightly larger loss of quality, specially at higher rates of evolution. In fact, in most datasets with $I = 1$, increasing the indel probability P also increases the quality of MGRA, while the opposite happens for $I = 5$. We believe that this might be a

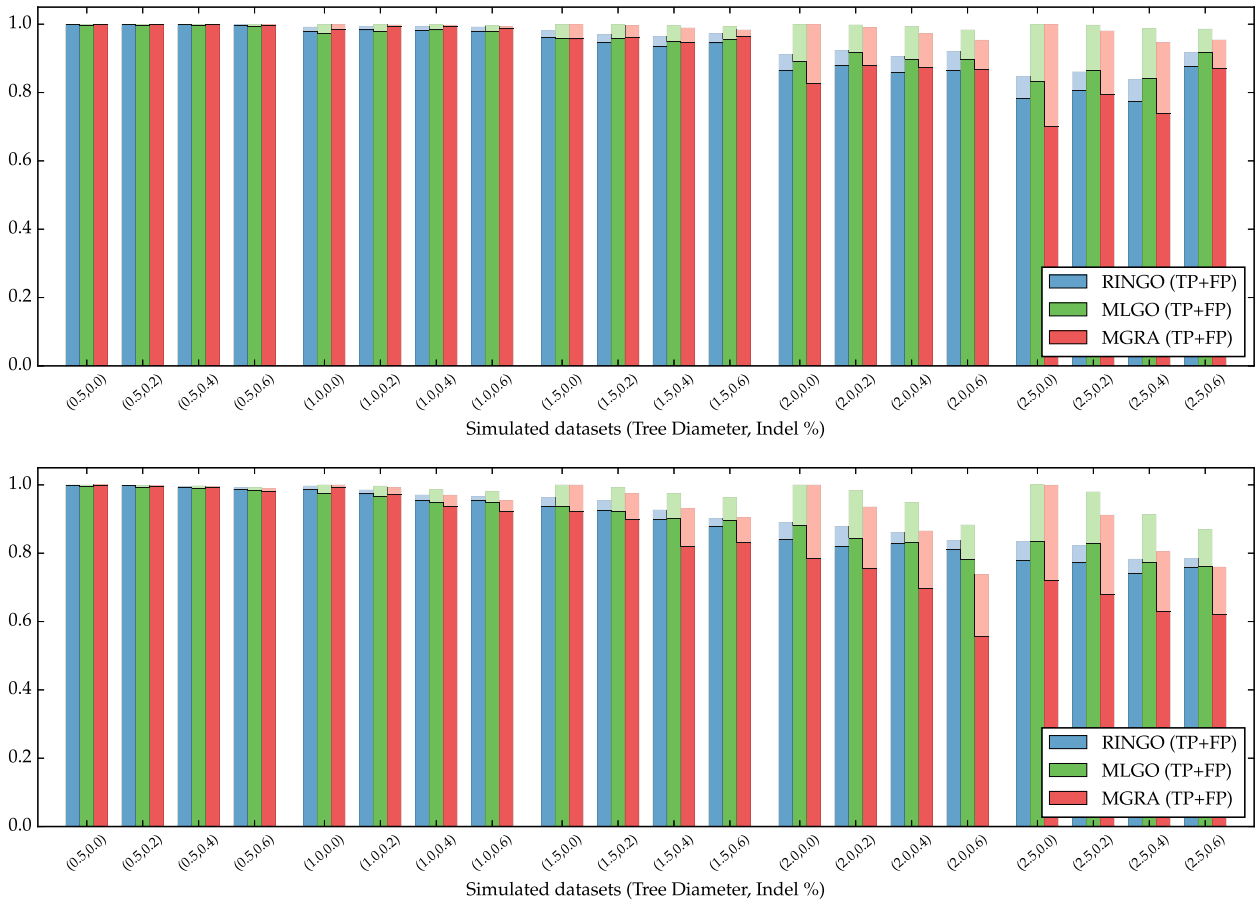

Fig. 6 Quality of the adjacency reconstruction for each dataset, with single gene indels ($I = 1$, *top plot*) and with indel size $\in [1, 5]$ ($I = 5$, *bottom plot*). Each column group represents the average results of RINGO (with custom adjacency weights and branch length estimation with ME), MLGO and MGRA on each dataset, with the percentage of true positives and false positives, when comparing the simulated adjacencies and the reconstructed adjacencies in all internal nodes of the simulated trees

Table 1 Average adjacency reconstruction, in terms of true positives (TP) and false positives (FP) for all tested algorithms, for all datasets grouped by tree diameter

Diameter (D)	0.5 *n*		1.0 *n*		1.5 *n*		2.0 *n*		2.5 *n*	
Adjacency results (%)	TP	FP	TP	FP	TP	FP	TP	FP	TP	FP
Unitary Indels										
RINGO – Sim. branch lengths	99.8	0.2	99.1	0.7	94.0	3.1	87.9	4.3	81.8	5.5
RINGO – Est. branch lengths with ME	99.8	0.2	99.0	0.7	93.7	3.0	87.7	4.1	80.6	5.6
RINGO – Est. branch lengths with LS	99.8	0.2	99.0	0.7	93.4	3.0	86.7	4.1	80.7	5.4
RINGO – DeClone weights, $kT = 0.1$	99.6	0.6	98.1	2.0	92.1	7.8	86.9	8.7	80.7	11.0
RINGO – DeClone weights, $kT = 0.4$	99.6	0.6	98.1	1.9	92.5	9.1	88.4	9.9	82.7	16.8
RINGO – DeClone weights, $kT = 0.8$	99.2	1.2	97.5	2.9	91.8	9.9	88.0	10.5	82.5	17.1
MLGO	99.6	0.3	98.6	1.3	94.6	5.0	91.8	7.8	85.6	13.8
MGRA	99.9	0.0	99.3	0.5	95.1	3.5	85.8	12.8	70.4	26.7
Indel length $\in [1, 5]$										
RINGO – Sim. branch lengths	99.4	0.3	96.7	1.4	92.0	2.6	73.8	6.2	81.0	4.9
RINGO – Est. branch lengths with ME	99.4	0.3	96.6	1.4	91.6	2.6	71.8	5.8	79.5	4.3
RINGO – Est. branch lenghts with LS	99.4	0.2	96.7	1.4	91.4	2.6	70.2	5.1	77.0	3.8
RINGO – DeClone weights, $kT = 0.1$	99.3	0.7	95.4	5.7	90.6	10.4	72.0	14.1	79.3	13.4
RINGO – DeClone weights, $kT = 0.4$	99.3	0.7	95.7	5.7	91.3	11.0	75.2	19.9	82.1	16.9
RINGO – DeClone weights, $kT = 0.8$	99.1	1.1	95.1	6.8	90.7	12.3	74.9	20.3	81.9	17.4
MLGO	98.9	0.7	95.9	3.7	90.6	7.0	75.0	15.9	81.7	13.8
MGRA	99.3	0.3	99.7	0.3	94.8	3.8	65.5	34.5	57.4	42.4

RINGO algorithm was tested with the InferCARs adjacency weights, using the simulated tree branch lengths and with branch lenghts estimated with Minimum Evolution or Least Squares methods. RINGO was also ran with DeClone weights, with varying *kT* values

consequence of the way that MGRA models the prosthetic chromosomes by adding an edge v^t, v^h for each missing gene, which implicitly assumes that this gene is part of an unitary indel. In that respect, using a DCJ indel model such as the one in RINGO, that allows for block indels, will give better results when block indels do occur, which we believe is the more realistic case.

Using the DCJ-Indel distance [20], we also measured how far the reconstructed genomes are from the simulated genomes in average, and the results are shown on Fig. 7. As the quality results indicate, at lower rate, specially with unitary indels, MGRA has the smallest distances, but they increase rapidly for higher rates. Comparing MLGO and RINGO for the higher rates, even though MLGO has a higher percentage of false positives, it has the smallest distances to the ancestral genomes. We believe that this is caused by the fact that the DCJ distance strongly penalizes fragmentation. Therefore, comparing conservative methods like RINGO, that have a lower percentage of false positives, with more aggressive methods like MLGO, with more true positives at the cost of higher false positive percentage, the latter methods will have smaller distances. For instance, consider an ancestral unichromosomal genome $A = (a, b, c, d)$, where the letters represent four blocks. If a method correctly reconstructs all four blocks, but not the connection between them, that is, a fragmented genome $B = (a)(b)(c)(d)$ with four chromosomes, then we have that the DCJ distance is $d(A, B) = 3$. Now, consider another method that also reconstructs the four blocks correctly, but gives a wrong ordering, such as $C = (a, c, b, d)$. Surprisingly, we have that $d(A, C) = 2$, even though this reconstruction has the same number of correct adjacencies as the previous one, but more false positives. Indeed, for the general case of an ancestral genome $A = (a_1, a_2, \ldots, a_n)$ and a fragmented reconstruction $B = (a_1) \ldots (a_n)$, we have $d(A, B) = n - 1$, which is in fact the DCJ diameter for n blocks, that is, the maximum possible DCJ distance. Any ordering of the blocks $a_1, \ldots, a_n$, even completely random, will have an equal or smaller distance to genome A. Therefore, aggressive methods that try to minimize fragmentation by adding adjacencies, even with small support, will have a smaller distance to the correct ancestral genome, but we argue that this is no indication of a better reconstruction.

While comparing running times, for RINGO and MGRA the determining parameter is the rate of evolution, controlled by the parameter D. For MLGO, the running times stayed around one minute regardless of the rate. On Table 2, we show the average running times for all repetitions and indel rates $I \in \{0, 0.2, 0.4, 0.6\}$, in each of the different diameters, for the two datasets $I = 1$ and $I = 5$. The running time of RINGO is smaller than MLGO

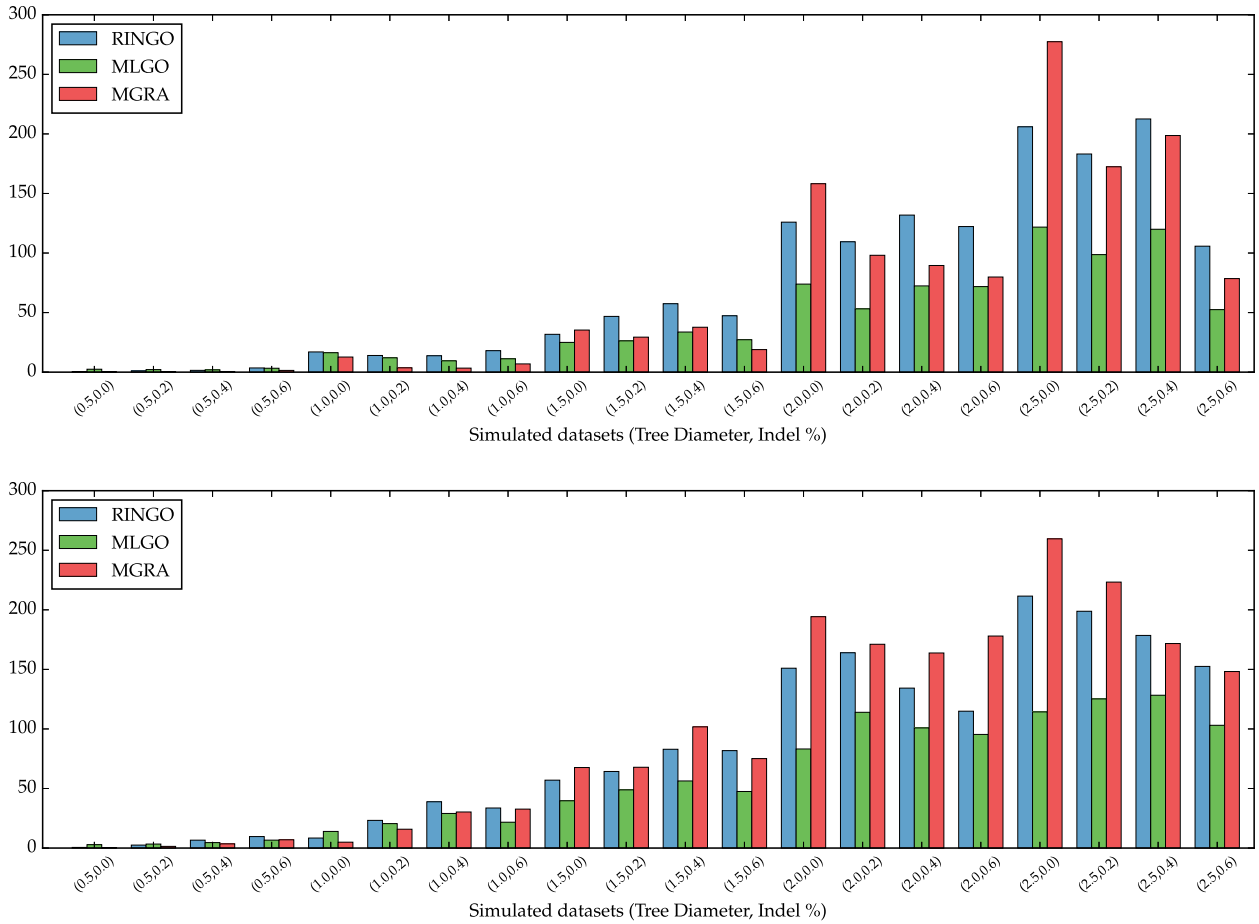

Fig. 7 DCJ-Indel distance between the simulated and reconstructed genomes, with only single gene indels ($I = 1$, *top plot*) and with indel size $\in[1, 5]$ ($I = 5$, *bottom plot*). Each column group represents the average distances of RINGO, MLGO and MGRA on each dataset

and increases in a much slower rate than MGRA, which increases exponentially for larger rates of evolution.

In summary, these results show that algorithms based on intermediate genomes can perform at quality levels equal or higher than current approaches for ancestral reconstruction, while also being much faster.

Conclusion

In this paper we proposed a new method for ancestral reconstruction of gene orders for genomes with unequal gene content by expanding a previous approach for genomes with same gene content. The IG algorithm is faster and in many datasets has a better reconstruction quality than MGRA and MLGO, specially for higher rates of evolution. We believe that one of the strongest points of our approach is the use of extra information, in the form of intermediate genomes, and not simply relying on the parsimonious idea of minimizing tree distances. With that, not only the quality of the reconstructed genomes is improved but also the search space is drastically reduced, resulting in faster algorithms. We also think that a combined approach with ideas from both worlds

Table 2 Average running time of 20 runs of each algorithm and all indel rates $I \in \{0, 0.2, 0.4, 0.6\}$, for different tree diameters with two different parameters I determining the size of the indels in number of genes

Dataset	$I = 1$, unitary indels					$I = 5$, indels with 1 to 5 genes				
Diameter (D)	0.5 n	1 n	1.5 n	2 n	2.5 n	0.5 n	1 n	1.5 n	2 n	2.5 n
RINGO	3 s	3 s	5 s	7 s	7 s	3 s	4 s	5 s	7 s	8 s
MLGO	1 m 6 s	1 m 10 s	1 m 7 s	1 m 9 s	1 m 16 s	57 s	60 s	1 m 4 s	1 m 7 s	1 m 10 s
MGRA	7 s	1 m 46 s	12 m 12 s	56 m 55 s	2 h 2 m 41 s	23 s	6 m 56 s	48 m 42 s	2 h 1 m 44 s	2 h 40 m 18 s

could deliver very good results. As an example, we could think of combining the space reduction power of intermediate genomes with the strong space search techniques of MGRA.

There are many ways that we can improve the ideas presented in this paper. For one, instead of using a heuristic for solving the maximum weight intermediate genome, we will test how solving this problem exactly changes the results, whether by using a FPT such as the one described, or resorting to an integer linear programming for the more complex cases. We also plan to extend the current framework to allow the presence of duplicated genes.

The proposed algorithms were implemented as Python 2.7 scripts in a software called RINGO, that can be downloaded at https://github.com/pedrofeijao/RINGO. Also included are scripts to generate simulations and parse the reconstruction results on the simulated datasets, comparing RINGO with other algorithms.

Acknowledgements
We would like to thank Nina Luhmann for providing the scripts that were used for obtaining DeClone adjacency weights.

Declarations
EA is funded from the Brazilian research agency CNPq grant Ciência sem Fronteiras Postdoctoral Scholarship 234234/2014x-8. We acknowledge support of the publication fee by Deutsche Forschungsgemeinschaft and the Open Access Publication Funds of Bielefeld University.

This article has been published as part of BMC Bioinformatics Vol 17 Suppl 14, 2016: Proceedings of the 14th Annual Research in Computational Molecular Biology (RECOMB) Comparative Genomics Satellite Workshop: bioinformatics. The full contents of the supplement are available online at https://bmcbioinformatics.biomedcentral.com/articles/supplements/volume-17-supplement-14.

Availability of data and materials
RINGO is available at https://github.com/pedrofeijao/RINGO.

Authors' contributions
PF and EA developed the theoretical results of the proposed algorithms and wrote the article. PF implemented the software, generated the simulations and obtained the experimental results. Both authors read and approved the final manuscript.

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

Consent for publication
Not applicable.

Ethics approval and consent to participate
Not applicable.

Author details
[1]Technische Fakultät and CeBiTec, Universität Bielefeld, Universitätsstr. 25, 33615 Bielefeld, Germany. [2]Faculdade de Computação, Universidade Federal de Mato Grosso do Sul – UFMS, Campo Grande, MS, Brazil.

Published: 11 November 2016

References

1. Sankoff D, Blanchette M. Multiple genome rearrangement and breakpoint phylogeny. J Comput Biol. 1998;5(3):555–70.
2. Moret BM, Wyman S, Bader Da, Warnow T, Yan M. A new implementation and detailed study of breakpoint analysis. In: Pacific Symposium on Biocomputing. Singapore: World Scientific Publishing; 2001. p. 583–94.
3. Bourque G, Pevzner PA. Genome-scale evolution: reconstructing gene orders in the ancestral species. Genome Res. 2002;12(1):26.
4. Yancopoulos S, Attie O, Friedberg R. Efficient sorting of genomic permutations by translocation, inversion and block interchange. Bioinformatics. 2005;21(16):3340–6.
5. Bergeron A, Mixtacki J, Stoye J. A unifying view of genome rearrangements. Lect Notes Comput Sci. 2006;4175:163–73.
6. Zheng C, Sankoff D. On the PATHGROUPS approach to rapid small phylogeny. BMC bioinformatics. 2011;12 Suppl 1(Suppl 1):4.
7. Xu W, Moret B. GASTS: Parsimony scoring under rearrangements. In: Proceedings of the 11th International Workshop on Algorithms in Bioinformatics (WABI 2011). Berlin Heidelberg: Springer; 2011. p. 351–63.
8. Alekseyev MA, Pevzner PA. Breakpoint graphs and ancestral genome reconstructions. Genome Res. 2009;19(5):943–57.
9. Avdeyev P, Jiang S, Aganezov S, Hu F, Alekseyev MA. Reconstruction of ancestral genomes in presence of gene gain and loss. J Comput Biol. 2016;23(3):150–64.
10. Feijao P, Meidanis J. SCJ: a breakpoint-like distance that simplifies several rearrangement problems. IEEE/ACM Trans Comput Biol Bioinforma. 2011;8:1318–1329.
11. Biller P, Feijão P, Meidanis Ja. Rearrangement-based phylogeny using the single-cut-or-join operation. IEEE/ACM Trans Comput Biol Bioinforma. 2013;10(1):122–34.
12. Ma J, Zhang L, Suh BBB, Raney BJBJ, Burhans RC, Kent WJJ, Blanchette M, Haussler D, Miller W. Reconstructing contiguous regions of an ancestral genome. Genome Res. 2006;16(12):1557–1565.
13. Gagnon Y, Blanchette M, El-Mabrouk N. A flexible ancestral genome reconstruction method based on gapped adjacencies. BMC bioinformatics. 2012;13 Suppl 1(Suppl 19):4.
14. Jones BR, Rajaraman A, Tannier E, Chauve C. ANGES: reconstructing ANcestral GEnomeS maps. Bioinformatics (Oxford, England). 2012;28(18):2388–390.
15. Hu F, Zhou J, Zhou L, Tang J. Probabilistic Reconstruction of Ancestral Gene Orders with Insertions and Deletions. IEEE/ACM Trans Comput Biol Bioinforma. 2014;5963(c):1–1.
16. Hu F, Lin Y, Tang J. MLGO: phylogeny reconstruction and ancestral inference from gene-order data. BMC bioinformatics. 2014;15(1):354.
17. Perrin A, Varré J-s, Blanquart S, Ouangraoua A. ProCARs : Progressive Reconstruction of Ancestral Gene Orders. BMC Genomics. 2015;16(Suppl 5):6.
18. Luhmann N, Thévenin A, Ouangraoua A, Wittler R, Chauve C. In: Bourgeois A, Skums P, Wan X, Zelikovsky A, editors. The SCJ Small Parsimony Problem for Weighted Gene Adjacencies. Cham: Springer; 2016, pp. 200–10.
19. Feijão P. Reconstruction of ancestral gene orders using intermediate genomes. BMC Bioinformatics. 2015;16(Suppl 14):3.
20. Compeau PE. DCJ-Indel sorting revisited. Algorithms Mole Biol: AMB. 2013;8(1):6.
21. Braga MDV, Willing E, Stoye J. Double cut and join with insertions and deletions. J Comput Biol. 2011;18(9):1167–84.
22. Arndt W, Tang J. Emulating Insertion and Deletion Events in Genome Rearrangement Analysis. 2011 IEEE Int Conf Bioinforma Biomed. 2011:105–108.
23. Braga MDV, Stoye J. The solution space of sorting by DCJ. J Comput Biol. 2010;17(9):1145–65.
24. Swenson K, Blanchette M. Models and Algorithms for Genome Rearrangement with Positional Constraints In: Pop M, Touzet H, editors. Algorithms in Bioinformatics SE - 18. Lecture Notes in Computer Science. Berlin, Heidelberg: Springer; 2015. p. 243–56.
25. Xu AW. The median problems on linear multichromosomal genomes: graph representation and fast exact solutions. J Comput Biol. 2010;17(9):1195–211.
26. Zhang M, Arndt W, Tang J. An exact solver for the DCJ median problem. Pacific Symposium on Biocomputing. Pacific Symposium on Biocomputing. 2009;14:138–49.
27. Haghighi M, Sankoff D. Medians seek the corners, and other conjectures. BMC bioinformatics. 2012;13 Suppl 1(Suppl 19):5.

28. Caprara A. The reversal median problem. INFORMS J Comput. 2003;15(1): 93–113.
29. Tannier E, Zheng C, Sankoff D. Multichromosomal median and halving problems under different genomic distances. BMC bioinformatics. 2009;10:120.
30. Valiente G. A New Simple Algorithm for the Maximum-Weight Independent Set Problem on Circle Graphs In: Ibaraki T, Katoh N, Ono H, editors. Algorithms and Computation SE - 15. Lecture Notes in Computer Science. Berlin, Heidelberg: Springer; 2003. p. 129–37.
31. Edmonds J. Maximum matching and a polyhedron with 0, l-vertices. J Res Nat Bur Standards B. 1965;69(1965):125–30.
32. Chauve C, Ponty Y, Zanetti JPP. Evolution of genes neighborhood within reconciled phylogenies: an ensemble approach. BMC Bioinformatics. 2015;16(19):1–9.
33. Waterman MS, Smith TF, Singh M, Beyer WA. Additive evolutionary trees. J Theor Biol. 1977;64(2):199–213.
34. Fitch WM, Margoliash E. Construction of Phylogenetic Trees. Science. 1967;155(3760):279–84.
35. Kolmogorov V. Blossom V: a new implementation of a minimum cost perfect matching algorithm. Math Program Comput. 2009;1(1):43–67.