JOURNAL OF COMPUTATIONAL BIOLOGY
Volume 21, Number 1, 2014
© Mary Ann Liebert, Inc.
Pp. 1–15
DOI: 10.1089/cmb.2013.0004

Research Articles

On the Complexity of Rearrangement Problems under the Breakpoint Distance

JAKUB KOVÁČ

ABSTRACT

We study the complexity of rearrangement problems in the generalized breakpoint model of Tannier et al. and settle several open questions. We improve the algorithm for the median problem and show that it is equivalent to the problem of finding maximum cardinality nonbipartite matching (under linear reduction). On the other hand, we prove that the more general small phylogeny problem is NP-hard. Surprisingly, we show that it is already NP-hard (or even APX-hard) for a quartet phylogeny. We also show that in the unichromosomal and the multilinear breakpoint model the halving problem is NP-hard, refuting the conjecture of Tannier et al. Interestingly, this is the first problem that is harder in the breakpoint model than in the double cut and join or reversal models.

Key words: breakpoint distance, halving, matching, median, NP-hard, phylogeny.

1. INTRODUCTION

In this article, we study several rearrangement problems in different variants of the breakpoint model. Throughout evolution, large-scale rearrangement mutations, such as inversions or translocations, affect the order of genes in a genome. When two genes (or conserved segments or markers) are adjacent in one genome, but not in the other, we call this position a breakpoint. We can then define the breakpoint distance simply by counting the number of breakpoints.

The *breakpoint model* was originally introduced by Sankoff and Blanchette (1998) who tried to reconstruct the ancestral gene orders, given a phylogenetic tree and gene orders of the extant species, based on the parsimony criterion, minimizing the sum of distances along the branches of the tree. This is known as the *small phylogeny* problem.[1] Unfortunatelly, the problem is NP-hard already when we have three species—an important special case known as the *median* problem. In fact, the median problem turns out to be NP-hard for almost all rearrangement distances [breakpoint (Bryant, 1998; Pe'er and Shamir, 1998; Tannier et al., 2009), reversal (Caprara, 2003), and double cut and join (DCJ) (Tannier et al., 2009)].

One notable exception is the general breakpoint model. Tannier et al. (2009) observed that if we drop the condition that genomes are unichromosomal and that all chromosomes are linear, we get a simple model where the median problem is solvable in polynomial time. Even though this model is not very biologically plausible and more realistic models exist, the breakpoint model may still be useful for upper and lower bounds, and solutions in this model may serve as good starting points for the more elaborate and complicated models.

Department of Computer Science, Comenius University, Bratislava, Slovakia.
[1]As opposed to the large-phylogeny problem, where the phylogenetic tree is not given and is part of the solution.

1.1. Previous results and our contribution

There are several variants of the breakpoint model depending on what karyotypes we allow. In the *unichromosomal* (linear or circular) model, the genome may only consist of one chromosome. In the *multilinear model*, the genome may consist of multiple linear chromosomes, and finally, the *mixed model* allows for any number of linear and circular chromosomes (even though this is not biologically plausible).

For the unichromosomal model, Pe'er and Shamir (1998) and Bryant (1998) showed that the median problem is NP-hard. This result was extended to the multilinear model by Tannier et al. (2009), and Zheng et al. (2008) showed the NP-hardness for a related problem called guided halving (see Preliminaries).

Curiously, the ordinary halving problem was not studied before in the breakpoint model, and Tannier et al. (2009) also leave it open. Moreover, they conjecture that the problem is polynomially solvable; this might perhaps be attributed to the fact that the halving problem is polynomially solvable in far more complicated models such as reversal/translocation (RT) (El-Mabrouk and Sankoff, 2003) or DCJ (Alekseyev and Pezner, 2007; Mixtacki, 2008; Warren and Sankoff, 2009; Kováč et al., 2011). Nevertheless, we refute this conjecture (unless P = NP) by proving that the halving problem is NP-complete in the unichromosomal and multilinear models.

In the mixed model, Tannier et al. (2009) showed that median, halving, and guided halving problems are solvable in polynomial time. Two interesting open questions remained in their work. These are also articulated in the monograph by Fertin et al. (2009):

1. The best time complexity for the median and guided halving problems under the breakpoint distance on multichromosomal genomes (with circular chromosomes allowed) is $O(n^3)$, using a reduction to the maximum weight perfect matching problem. It is an open problem to devise an ad-hoc algorithm with better complexity.
2. The small phylogeny and large phylogeny problems under the breakpoint distance are open regarding multichromosomal signed genomes where linear and circular chromosomes are allowed.

We resolve the first question in a positive way by showing a more efficient algorithm running in $O(n\sqrt{n})$ time. In fact, we show that maximum cardinality matching and our problem have the same complexity. The same technique also improves the algorithms for halving and guided halving.

The second question is resolved negatively: small and large phylogeny problems are NP-hard under the breakpoint distance. Surprisingly, the problems are NP-hard (and APX-hard) even for four species, i.e., a quartet phylogeny. In other words, while finding an ancestor for three species is easy, finding two ancestors for four species is already hard.

The previous work and our new results are summarized in Table 1.

1.2. Road map

In the next section, we define the different variants of the breakpoint model and state the rearrangement problems. In Section 3, we refute the conjecture of Tannier et al. (2009) and prove that the halving problem is NP-hard for the unichromosomal and multilinear breakpoint model. In the following two sections, we

Table 1. Our New Results in Context of the Previously Known Results

Breakpoint model	*Median*	*Halving*	*Guided halving*	*Small phylogeny*
Unichromosomal (linear or circular)	NP-hard[a]	NP-hard*	NP-hard[b]	NP-hard[c]
Multilinear	NP-hard[d]	NP-hard*	NP-hard[b]	NP-hard[c]
Multichromosomal (circular or mixed)	$O(n^3)$[d] $O(n\sqrt{n})$*	$O(n^3)$[d] $O(n)$*	$O(n^3)$[d] $O(n\sqrt{n})$*	NP-hard*

*New.
[a]Bryant, 1998; Pe'er and Shamir, 1998. [b]Zheng et al., 2008. [c]Trivial. [d]Tannier et al., 2009.

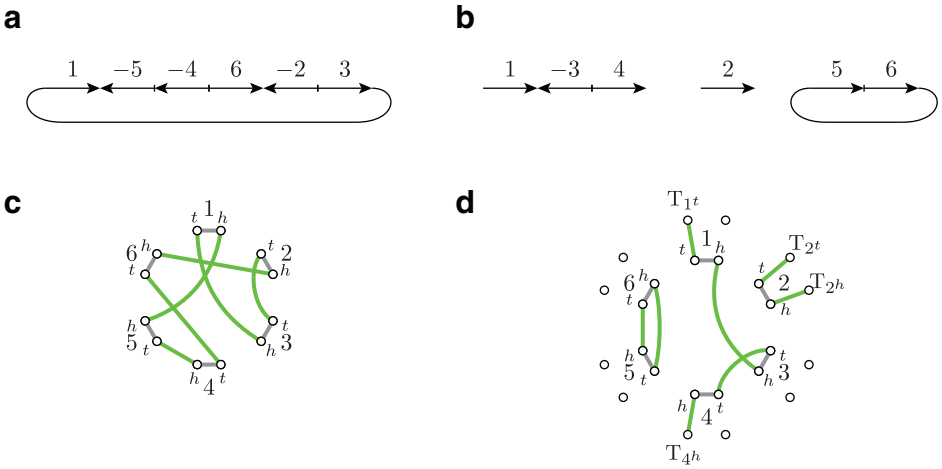

FIG. 1. Example of a circular genome (*left*) and a mixed genome (*right*) and their representations by matchings. **(a)** The order of genes in a genome. Each arrow corresponds to a single gene with known orientation. (**b**) A genome with two linear and one circular chromosome. Such genomes are not found in nature; however, the model is motivated by tractability of the rearrangement problems such as median. (**c**) Representation of the genome above by a perfect matching. The green edges are the adjacencies of π; the gray edges form the base matching B. The Hamiltonian cycle $\pi \cup B$ corresponds to the single chromosome. (**d**) The genome above represented as a set of adjacencies (green matching). Gray edges form the base matching B. Components of $\pi \cup B$ are paths and cycles, corresponding to linear and circular chromosomes, respectively.

study the general breakpoint model. In Section 4, we improve upon the algorithm of Tannier et al. (2009) and show that it is equivalent to the maximum matching problem. The hardness of the small phylogeny problem is studied in Section 5, and we conclude in Section 6.

2. PRELIMINARIES

2.1. Genome models and the breakpoint distance

We assume that all of the studied genomes have the same gene content $\mathcal{G}$. Each gene $g \in \mathcal{G}$ is an oriented segment of DNA having two *extremities*: a *head* g_h and a *tail* g_t. We represent a circular genome π by a set of edges: An edge between extremities x and y, called *adjacency*, indicates that x and y are adjacent in the genome. Every extremity is adjacent to exactly one other extremity, so we can *identify genomes with perfect matchings* over the set of extremities.

Let us define an auxiliary *base matching* $B = \{g_h g_t : g \in \mathcal{G}\}$ where each edge connects the two ends of some gene. Then all vertices have degree 2 in the union[2] $\pi \cup B$, and $\pi \cup B$ decomposes into a set of cycles, which naturally correspond to the circular chromosomes of our genome (see Fig. 1, left).

In the *general* (multichromosomal circular) model, genomes can have multiple circular chromosomes and any perfect matching π corresponds to a genome. In the unichromosomal *circular* model, we require that the genome only consists of a single chromosome, so $\pi \cup B$ is a Hamiltonian cycle. Such a matching π is sometimes called a Hamiltonian matching.

Let π_1 and π_2 be two genomes—two perfect matchings. Then the breakpoint distance between π_1 and π_2 is defined as $d(\pi_1, \pi_2) = n - \text{sim}(\pi_1, \pi_2)$, where n is the number of genes and $\text{sim}(\pi_1, \pi_2)$ is the number of common adjacencies. The breakpoint distance satisfies all the properties of a metric and is used in the literature; however, we find it easier to work directly with the similarity measure $\text{sim}(\pi_1, \pi_2)$.

In models involving linear chromosomes, we add a vertex T_x for each extremity x. These vertices are called *telomeres*, and a *telomeric adjacency* xT_x indicates that x is an end of a linear chromosome (see Fig. 1, right). Genomes will again correspond to matchings with an added condition that T_x may only be adjacent to x. If π is such a matching, $\pi \cup B$ consists of cycles and paths ending in telomeres, which

[2]Technically, this is a disjoint or multiset union; we allow parallel edges forming 2-cycles.

correspond to circular and linear chromosomes, respectively. In the *mixed* model, any such matching π represents a genome; in the *multilinear* model, we require that every chromosome is linear; and in the *linear* model, we only allow a single linear chromosome.

We can write the breakpoint distance again in the form $d(\pi_1, \pi_2) = n - \text{sim}(\pi_1, \pi_2)$, where this time, $\text{sim}(\pi_1, \pi_2)$ is the number of common adjacencies plus *half* the number of common telomeric adjacencies (as introduced by Tannier et al., 2009).

2.2. Duplicated genomes

We will also work with *duplicated* genomes that have exactly two copies of each gene. For each gene g, let us label the first copy g^1 and the second copy g^2. Then we can represent a duplicated genome by an ordinary genome δ over the gene set $\{g^1, g^2 : g \in \mathcal{G}\}$. However, note that the labels were introduced arbitrarily and we consider two genomes that differ only in the subscripts of some genes as equivalent. A duplicated genome corresponds to the equivalence class $[\delta]$. A breakpoint distance (similarity) between two duplicated genomes $[\gamma]$ and $[\delta]$ is the minimum distance (maximum similarity) between ordinary genomes $\gamma' \in [\gamma]$ and $\delta' \in [\delta]$.

Let us write $\theta = \pi \oplus \pi$ for a *perfectly* duplicated genome. For each linear chromosome in π, θ contains two copies of the chromosome, and for each circular chromosome in π, θ contains either two copies of the chromosome or one chromosome consisting of the two copies consecutively. The distance between an ordinary genome π and a duplicated genome $[\delta]$, also called *double distance* and denoted $dd(\pi, \delta)$, is then the distance between $\pi \oplus \pi$ and $[\delta]$.

We say that π and $[\delta]$ have adjacency xy in common, if x, y are adjacent in π and x^i, y^j are adjacent in δ for some i and j. We say that they have the adjacency xy twice in common, if either x^1y^1 and x^2y^2, or x^1y^2 and x^2y^1 are adjacent in δ. Tannier et al. (2009) showed that the double distance $dd(\pi, \delta)$ can be computed simply as $dd(\pi, \delta) = 2n - \text{sim}(\pi, \delta)$, where $\text{sim}(\pi, \delta)$ is the number of adjacencies in common plus half the number of telomeric adjacencies in common (adjacencies twice in common are counted as 2).

2.3. Rearrangement problems

Once we have defined a genome model and a distance measure, we can define the following parsimony problems of interest. Assume that we have two genomes π_1 and π_2, and we would like to reconstruct their common ancestor α. Using a third, outgroup genome π_3, we can formulate the task as the *median* problem: Given π_1, π_2, and π_3, find genome α (called *median*) that minimizes the total distance from π_1, π_2, and π_3. In the *breakpoint median* problem, we are minimizing the breakpoint distance, which is the same as *maximizing* the median score $S(\alpha) = \text{sim}(\alpha, \pi_1) + \text{sim}(\alpha, \pi_2) + \text{sim}(\alpha, \pi_3)$. Note that the genome model imposes further constraints on the solution: the number and the type of chromosomes.

We can generalize the median problem to the median of k genomes problem, where given genomes $\pi_1, \ldots, \pi_k$, we should find genome α that maximizes the score $S(\alpha) = \sum_i \text{sim}(\alpha, \pi_i)$. However, an even more important generalization is the *small phylogeny* problem, where we are given a phylogenetic tree and gene orders of the extant species (leaves of the tree). The task is to reconstruct all the ancestral genomes, i.e., to find gene orders for each internal vertex, while minimizing the sum of breakpoint distances along the edges of the phylogenetic tree. The median problem is a special case of the small phylogeny problem with just three species.

Another classical problem is the *halving* problem. Imagine a genome π that underwent a whole genome duplication. The perfectly duplicated genome $\theta = \pi \oplus \pi$ was then rearranged to its present-day form γ. In the *halving* problem, we would like to reconstruct the preduplication ancestor π given the present-day genome γ. More precisely, we would like to find an ordinary genome α that minimizes the double distance from γ. The *halving* problem has usually many equivalent solutions. For better results, we can use an ordinary outgroup genome ρ (such that the speciation happened before the whole genome duplication) and search for genome α that minimizes the sum $dd(\alpha, \gamma) + d(\alpha, \rho)$. This is called the *guided genome halving* problem.

3. HALVING PROBLEM

Bryant (1998) showed that the median problem is NP-hard in the circular breakpoint model by reduction from the *directed hamiltonian cycle* problem. The halving problem was not studied previously in the

breakpoint model, but we show that it suffers the same ''Hamiltonian'' curse as the median problem. As the halving problem is polynomially solvable in more realistic models, such as the RT model (El-Mabrouk and Sankoff, 2003) or the DCJ model (Alekseyev and Pezner, 2007; Mixtacki, 2008; Warren and Sankoff, 2009; Kováč et al., 2011), the halving problem under the breakpoint distance will remain a mere curiosity: It is the first problem that is easier in the DCJ or even in the RT model than in the breakpoint model. Furthermore, it is the only known case where halving is NP-hard, even though the double distance is computable in polynomial time [in the DCJ model, the opposite is true: halving is easy, whereas the double distance is NP-hard (Tannier et al., 2009)].

Theorem 1. *Halving problem is NP-hard in the circular, linear, and multilinear breakpoint models.*

Proof. The proof is by reduction from the directed hamiltonian cycle problem. Plesník (1979) proved that this problem is still NP-hard for graphs with maximum degree 2, and the construction implies the problem is also NP-hard if all in-degrees and out-degrees are equal to 2. Note that such graphs have an Eulerian cycle.

Let $G = (V, E)$ be such a directed graph; the corresponding doubled genome δ will have two copies of a gene for each vertex in G, and an Eulerian cycle in G traversing each vertex twice will be the order of genes in δ. More precisely, let $G' = (V', E')$, where $V' = \{x_h^1, x_t^1, x_h^2, x_t^2 : x \in V\}$ and the edges in E' are defined as follows: traverse the Eulerian walk and, for each edge $xy \in E$, include edge $x_h^i y_t^j$ in E', where i and j are 1 if we are visiting the vertex for the first time, and 2 if we are visiting the vertex for the second time. Note that all edges go from head to tail, E' is a perfect matching, and G' defines the doubled genome δ consisting of a single circular chromosome.

Let α be a circular genome that is a solution to the halving problem. Note that δ has no double adjacencies, so α can have at most n adjacencies in common (none twice in common). This maximum can be attained if and only if all the adjacencies in α are of the form $x_h y_t$ (from head to tail), and for each such adjacency, $x_h^i y_t^j$ is an adjacency in δ for some i, j. This is true if and only if $xy \in E$. So by contracting the base matching (each head and tail of a gene into a single vertex) and orienting the edges (from head to tail), we get a directed Hamiltonian cycle in G.

For the linear and multilinear models, remove one edge xy from G and consider the problem of deciding whether G contains a directed Hamiltonian path. This problem is still NP-hard and can be reduced to the halving problem in the linear models: G now has an Eulerian path starting in y and ending in x. We replace the last adjacency $x_h^2 y_t^1$ in δ (corresponding to the removed edge) by two telomeric adjacencies $x_h^2 \mathrm{T}_{x_h^2}$ and $y_t^1 \mathrm{T}_{y_t^1}$ to get a linear genome. If α is a linear or multilinear solution to the halving problem, it can reach the maximum similarity if and only if all of its adjacencies (including the telomeric adjacencies) are in common with δ, and this is true if and only if contraction of α is a directed Hamiltonian path in G. ■

4. MEDIAN AND HALVING PROBLEMS IN THE GENERAL MODEL

From now on, we will study the *general* breakpoint model, i.e., the multichromosomal circular model where genomes are perfect matchings. We will also show how to extend the results to the mixed model and use the same techniques for halving and guided halving problem.

4.1. Breakpoint median

Tannier et al. (2009) noticed that finding a breakpoint median can be reduced to finding a maximum weight perfect matching. This can be done in $O(n^3)$ time by the algorithm of Gabow (1973) and Lawler (1976). We improve on this by showing an $O(n\sqrt{n})$ algorithm.

The solution by Tannier et al. (2009) (if we rephrase it using the similarity measure instead of the breakpoint distance) was to create a complete weighted graph G where vertices are extremities and weight $w(xy)$ of edge xy is the number of genomes that contain the adjacency xy. Any perfect matching α corresponds to some genome, and the weight of the matching is equal to its median score $S(\alpha)$.

Notice that instead of finding a maximum weight *perfect* matching, we can remove all the zero-weight edges from G and find an ordinary (not necessarily perfect) matching. We can then complete the genome by joining the free vertices arbitrarily. As the number of edges in G is now linear, maximum weight matching

can be found in $O(n^2 \log n)$ time by the algorithm of Gabow (1990) or even in $\tilde{O}(n\sqrt{n})$ time by the state-of-the art algorithm of Gabow and Tarjan (1991) using the fact that the weights are small integers.

Theorem 2. *The breakpoint median problem for k genomes can be solved in $O(kn\sqrt{n} \cdot \log(kn)\sqrt{\alpha(kn, n)\log n})$ time in the general model.* [*Here, $\alpha(m, n)$ is the inverse Ackermann function.*]

We further improve the algorithm for the most important special case, $k = 3$: Notice that when xy is an edge with weight 3, there is no other edge incident to x or y. Therefore, xy must belong to the maximum weight matching. Moreover, if xy has weight 2, there is a maximum weight matching that contains xy. Suppose to the contrary that xu and yv were matched in α instead. Then $w(xu)$ and $w(yv)$ are at most 1, and by exchanging these edges for xy and uv with weights $w(xy) = 2$ and $w(uv) \geq 0$, we get a matching with the same or even higher weight.

Thus, we can include all edges of weight 2 and 3 in the matching and remove matched vertices together with their incident edges. The remaining graph has only unit edge weights, so it suffices to find maximum *cardinality* matching. This can be done in $O(m\sqrt{n})$ time by the algorithm of Micali and Vazirani (1980). Thus, we have the following claim.

Theorem 3. *The breakpoint median problem for three genomes can be solved in $O(n\sqrt{n})$ time (in the general model).*

One might still wonder whether there is an even better algorithm for the median problem, which perhaps can avoid computation of maximum matching. Alas, we show that improving upon our result would be very hard, because it would immediately imply a better algorithm for the matching problem, beating the result of Micali and Vazirani (1980) (at least on cubic graphs), which has been an open problem for more than 30 years.

Biedl (2001) showed that the maximum matching problem is reducible to the maximum matching problem in *cubic* graphs by a *linear* reduction. This means that we can transform any given graph G with m edges to a cubic graph G' with $O(m)$ edges such that the maximum matching in G can be recovered from one in G' in $O(m)$ time. Thus, any $O(f(m))$ algorithm for finding maximum matching in cubic graphs implies an $O(f(m) + m)$ algorithm for arbitrary graphs.

We say that a reduction is *strongly linear* if it is linear, and both the number of vertices and the number of edges increase at most linearly. Such a reduction preserves the running time $O(f(m, n))$ depending on both the number of vertices and the number of edges.

We prove that the breakpoint median problem is equivalent to matching under linear reduction and to cubic matching under strongly linear reduction. If we write $\leq_\ell$ for linear and $\leq_{s\ell}$ for strongly linear reduction, we have

$$\text{matching} \leq_\ell \text{cubic matching} \leq_{s\ell} \text{breakpoint median} \leq_{s\ell} \text{matching}.$$

The first reduction is by Biedl (2001) and the last one was shown in Theorem 3 [in fact, a reduction to subcubic matching, where the degrees are at most 3, was shown; this is equivalent to cubic matching under the strongly linear reduction (Biedl, 2001)]. We now prove the middle reduction.

Let G be a cubic graph, an instance of the cubic matching problem. In breakpoint median, the input multigraph consists of three perfect matchings, i.e., is edge 3-colorable. However, not all cubic graphs are edge 3-colorable. The solution is to color edges arbitrarily and resolve conflicts as shown in Figure 2a. We can, for example, color the ends of edges at each vertex randomly with three different colors. When both ends of an edge are assigned the same color, we color the edge appropriately. When the ends have different colors, we subdivide the edge into three parts and use the third color for the middle edge (see Fig. 2a). Note that the size of a maximum matching in the modified graph is exactly one more than the size in the original graph: If xy is matched in the original, xu and vy can be matched in the modified graph. If xy is not matched, we can still match uv.

Now, the modified graph is edge 3-colorable but not cubic. We remedy this by duplicating the whole graph and connecting the corresponding vertices of low degree as shown in Figure 2b. As noted above, we may suppose that the auxiliary double edges $a_u a'_u$ and $a_v a'_v$ are matched, so ua_u, $u'a'_u$, va_v, and $v'a'_v$ are not matched, and given the solution for the breakpoint median problem, we can recover the maximum matching of G in $O(n)$ time. The reduction is obviously linear, so we have the following claim.

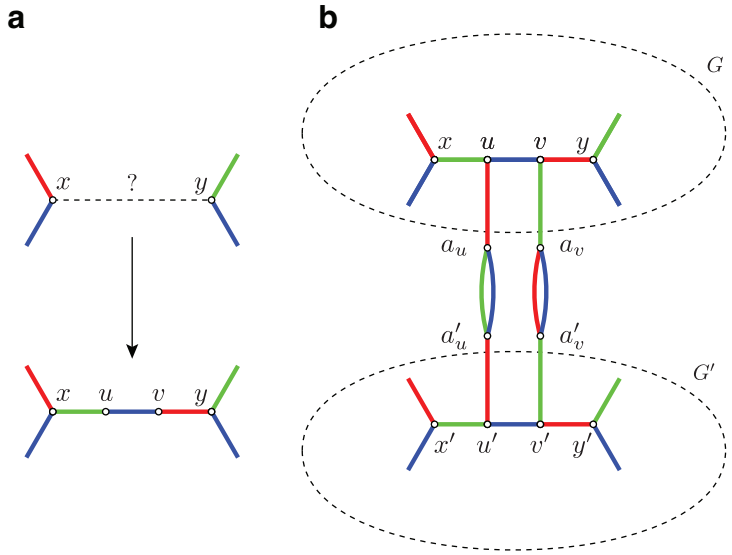

FIG. 2. Linear reduction of maximum matching in cubic graphs to breakpoint median problem. **(a)** Edge xy (*top*) should be colored green (this is the only missing color at x) and red at the same time (this is the missing color at y). We resolve this conflict by subdividing edge xy by two new vertices (*bottom*); we color xu green, vy red, and uv blue. **(b)** In the second phase, we duplicate graph G and connect the corresponding vertices with degree 2 as shown in the figure.

Theorem 4. *The breakpoint median problem (in the general model) has the same complexity as finding maximum cardinality matching in cubic graphs.*

4.2. Median in the mixed model

In the mixed model, the weight of a telomeric adjacency $x\mathrm{T}_x$ equals to half the number of genomes that contain $x\mathrm{T}_x$. If we multiply all weights by 2, we can use the algorithm by Gabow and Tarjan (1991) for integer weights, so the result of Theorem 2 remains valid also in the mixed model.

For the median of three genomes, an $O(n\sqrt{n})$ algorithm exists: We can include all the double and triple adjacencies in the matching, as well as double and triple telomeric adjacencies (edges of weight 1 and $1\frac{1}{2}$). If $w(x\mathrm{T}_x)=1\frac{1}{2}$, $x\mathrm{T}_x$ is a triple adjacency and no other edge is incident to either x or T_x in G. If $w(x\mathrm{T}_x) = 1$ but the median α contains adjacency xy instead, then $w(xy) \le 1$, and as T_x can only be incident to x, it must be unmatched (or matched by a zero-weight edge), and so we can replace xy by $x\mathrm{T}_x$ in α.

The remaining graph consists of edges with unit weight and weight $\frac{1}{2}$. Note, however, that all the $\frac{1}{2}$-weight edges are of the form $x\mathrm{T}_x$, and there is no other edge incident to T_x. We use the doubling trick again: we take two copies of graph G, and replace all pairs $x\mathrm{T}_x, x'\mathrm{T}'_x$ by a single edge xx' of unit weight. We can then remove all the telomere vertices. The resulting graph will have only unit weight edges, and the maximum matching will be exactly twice the size of the maximum matching in the original graph.

4.3. Halving problems in the general model

The same tricks can be used for halving and guided halving problems. Recall that in the halving problem, we are given a duplicated genome γ, and we are searching for genome α that minimizes the double distance $dd(\alpha, \gamma)$; in the guided halving problem, we are also given genome ρ and we are minimizing the sum $dd(\alpha, \gamma) + d(\alpha, \rho)$.

Again, we construct graph G, where this time, the weight of edge xy is the number of adjacencies among x^1y^1, x^1y^2, x^2y^1, x^2y^2 in γ and possibly xy in ρ (in the case of the guided halving problem). The rest of the solution is identical, leading to an $O(n\sqrt{n})$ algorithm for the guided halving problem. In the halving problem, the degrees of vertices in G are at most 2, and after including all the double edges in the solution, the remaining graph consists only of cycles and the maximum matching can be found trivially in linear time.

5. BREAKPOINT PHYLOGENY

In the small phylogeny problem, we reconstruct ancestral genomes given a phylogenetic tree and gene orders of the extant species while minimizing the sum of distances along the edges of the tree. This problem is NP-hard for most rearrangement distances and for most models; this follows trivially from the NP-hardness of the median problem. However, as we have seen in the previous section, this is not the case in the general breakpoint model, and the complexity of the small-phylogeny problem remained open (Fertin et al., 2009; Tannier et al., 2009).

We prove that the small phylogeny problem is NP-hard even for four species. More precisely, given four genomes π_1,π_2,π_3,π_4, the breakpoint quartet problem is to find ancestral genomes α_1, α_2 that maximize the sum of similarities $S(\alpha_1, \alpha_2) = \text{sim}(\pi_1, \alpha_1) + \text{sim}(\pi_2, \alpha_1) + \text{sim}(\alpha_1, \alpha_2) + \text{sim}(\alpha_2, \pi_3) + \text{sim}(\alpha_2, \pi_4)$.

Theorem 5. *The breakpoint quartet problem is NP-hard and even APX-hard in the general breakpoint model.*

The proof is inspired by the work of Dees (2009), who showed that the following problem is NP-hard: Given two graphs $G_1 = (V, E_1)$, $G_2 = (V, E_2)$, find two perfect matchings $M_1 \subseteq E_1$ and $M_2 \subseteq E_2$ with the maximum overlap $M_1 \cap M_2$. The problem is NP-hard even when the components in G_1 and G_2 are just cycles. In our proof, $\pi_1 \cup \pi_2$ will correspond to E_1, $\pi_3 \cup \pi_4$ will correspond to E_2, and the unknown ancestors α_1, α_2 will correspond to the unknown perfect matchings M_1, M_2.

Our proof is, however, much more involved, and there are two reasons for this. First, the problem formulation does not guarantee that $\alpha_1 \subseteq \pi_1 \cup \pi_2$ and $\alpha_2 \subseteq \pi_3 \cup \pi_4$. We will say that solution α_1,α_2 that satisfies this condition is in a *normal form*. The hard part of the proof is showing that we can transform any solution α_1, α_2 into a solution α_1', α_2' that has the same score and, moreover, it is in the normal form. The second major difficulty is that we are maximizing the sum $S(\alpha_1, \alpha_2)$ instead of just the size of the intersection $\text{sim}(\alpha_1, \alpha_2)$. To overcome these difficulties, we had to modify the edge gadget from the original proof and use a more restricted problem for the reduction.

5.1. *Overview of the proof*

The proof is by reduction from the cubic max-cut problem. Given a graph G, the max-cut problem is to find a cut of maximum size. We may rephrase this as a problem of coloring all vertices in G with two colors, red or green, while maximizing the number of red-green edges.

(Partition of V into the red part and the green part defines a cut, and its size is the number of edges with endpoints of different color.) In the cubic max-cut problem, the instances are cubic graphs; this variant is still NP-hard and APX-hard (Alimonti and Kann, 1997).

Let $G = (V, E)$ be a given cubic graph, an instance of the cubic max-cut problem. We will construct genomes π_1, π_2, π_3, and π_4 such that the maximum cut in G can be recovered from the solution α_1, α_2 of the breakpoint quartet problem in polynomial time.

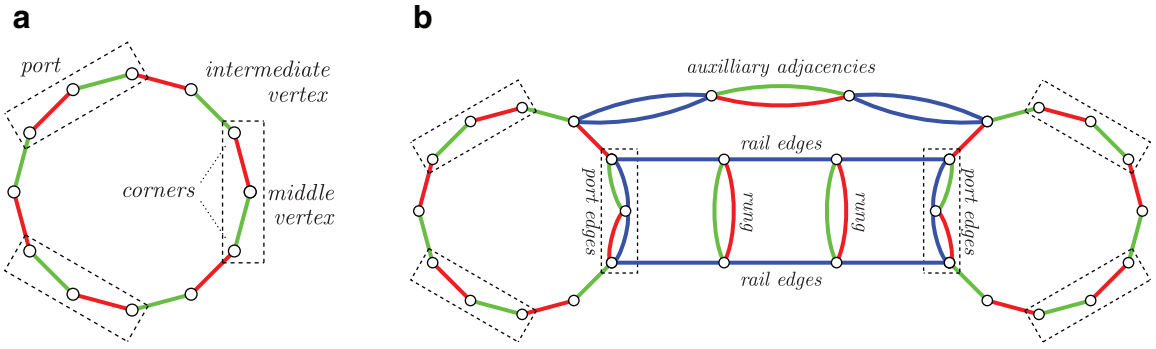

FIG. 3. The vertex and edge gadgets used in our reduction and the terminology used for different types of vertices and edges. The red and green edges are the adjacencies of π_1 and π_2, respectively. The cycles made of blue edges can be decomposed into two matchings: the adjacencies of π_3 and π_4. **(a)** Vertex gadget. **(b)** Edge gadget.

For each vertex of G, there will be a vertex gadget (see Fig. 3a) made of adjacencies of π_1 and π_2. Let π_1 be the red matching and π_2 the green matching. As we will prove later, we may suppose that $\alpha_1 \subseteq \pi_1 \cup \pi_2$, so within each vertex gadget, α_1 will contain either the red edges of π_1 or the green edges of π_2. This naturally corresponds to a red/green vertex coloring in the cubic max-cut problem.

The framed vertices in Figure 3a are called ''ports''; this is where the edge gadgets for the three incident edges are attached. For each edge of G, an edge gadget connecting the ports of the corresponding vertex gadgets is constructed as shown in Figure 3b. The blue cycles consist of two matchings: the adjacencies of π_3 and π_4. Again, as we will prove later, we may suppose that $\alpha_2 \subseteq \pi_3 \cup \pi_4$, i.e., the second ancestor consists only of the blue edges.

For future reference, let us state here again the claims to be proved in the form of a lemma:

Lemma 1 (Normal form). *Let π_1, π_2, π_3, π_4 be an instance of the breakpoint quartet problem constructed from a cubic max-cut instance as described above. Then any solution α_1,α_2 can be transformed in polynomial time into a solution α'_1, α'_2 such that $S(\alpha'_1, \alpha'_2) \geq S(\alpha_1, \alpha_2)$, $\alpha'_1 \subseteq \pi_1 \cup \pi_2$, and $\alpha'_2 \subseteq \pi_3 \cup \pi_4$.*

Once we prove the normal form lemma, the rest of the proof is easy: If α_1, α_2 is a solution in the normal form, term $\mathrm{sim}(\pi_1, \alpha_1) + \mathrm{sim}(\pi_2, \alpha_1)$ is always the same: we get $+6$ for each vertex gadget and $+6$ for each edge gadget. Similarly, term $\mathrm{sim}(\alpha_2, \pi_3) + \mathrm{sim}(\alpha_2, \pi_4)$ is always the same: we get $+9$ for each edge gadget. So the score $S(\alpha_1, \alpha_2)$ is maximized, when $\mathrm{sim}(\alpha_1, \alpha_2) = |\alpha_1 \cap \alpha_2|$ is maximized. Let uv be an edge in our graph G from the cubic max-cut problem. If we choose matchings of the same color for both vertex gadgets u and v, then α_1 and α_2 can only have one edge in common within the edge gadget uv (see Fig. 4a). However, if u and v have matchings of different color, we can set adjacencies of α_2 so that α_1 and α_2 have two edges in common (see Fig. 4b). When we sum up all these contributions, we get $S(\alpha_1, \alpha_2) = 20m + c$, where m is the number of edges in G, and c is the size of the cut corresponding to the matching α_1, so a polynomial algorithm for breakpoint quartet would imply a polynomial algorithm for cubic max-cut.

For the APX-hardness, note that for any graph with m edges, we can easily find a cut of size $c \geq m/2$. Let α^*_1, α^*_2 be an optimal solution for an instance of the breakpoint quartet problem and α_1, α_2 a solution such that $S(\alpha^*_1, \alpha^*_2) \leq (1+\epsilon)S(\alpha_1, \alpha_2)$. Let both solutions be in the normal form, and let c^* and $c \geq m/2$ be the sizes of the corresponding cuts. Then $20m + c^* \leq (1 + \varepsilon)(20m + c)$, and $c^* \leq (1 + \varepsilon)c + 20\varepsilon m \leq (1 + 41\varepsilon)c$. So a $(1 + \varepsilon)$-approximation algorithm for the breakpoint quartet problem would lead to a $(1 + 41\varepsilon)$-approximation algorithm for the cubic max-cut problem.

It can also be proved that the phylogenetic tree $((\pi_1, \pi_2), (\pi_3, \pi_4))$ is the most parsimonious. The alternative quartets $((\pi_1, \pi_3), (\pi_2, \pi_4))$ and $((\pi_1, \pi_4), (\pi_2, \pi_3))$ yield score $\leq 20m$, so this result also implies the NP- and APX-hardness of the large phylogeny problem. It remains an open problem whether computing the correct quartet (without reconstructing the ancestors) is hard.

5.2. *Notation, terminology, and other conventions*

We say that an adjacency $e \in \alpha_1$ is *supported* if $e \in \pi_1 \cup \pi_2$. Similarly, $e \in \alpha_2$ is *supported* if $e \in \pi_3 \cup \pi_4$. An adjacency that is not supported is *unsupported*. Furthermore, let $\Pi = \pi_1 \cup \pi_2 \cup \pi_3 \cup \pi_4$

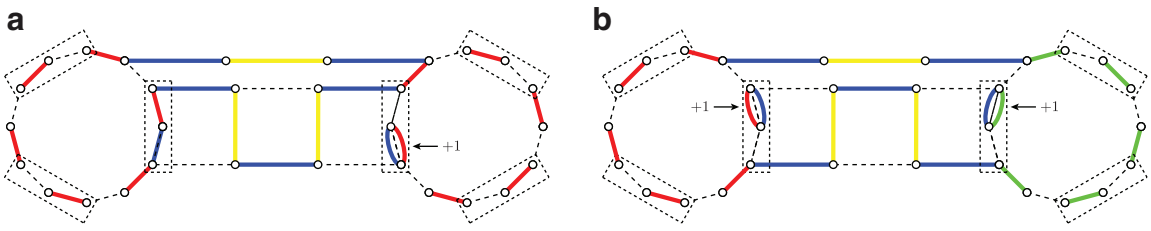

FIG. 4. The dashed edges indicate the underlying vertex and edge gadgets, the blue edges are adjacencies of α_2, and the red, green, and yellow edges are adjacencies of α_1. Here, we assume that α_1 and α_2 are in the normal form. **(a)** Adjacencies of the first ancestor α_1 (red edges) agree with the adjacencies of π_1 at both vertex gadgets. This corresponds to coloring both vertices red in the cubic max-cut problem. Note that α_1 and α_2 can only have one adjacency in common. **(b)** In the first vertex gadget, α_1 agrees with π_1 (red edges), and in the second gadget, α_1 agrees with π_2 (green edges). This corresponds to coloring the first vertex red and the second vertex green in the cubic max-cut problem. In this case, α_1 and α_2 have two adjacencies in common.

be the set of adjacencies present in at least one extant species. We will say that an adjacency $e \in \alpha_i$ is *weakly supported*, if $e \in \prod$.

Let us name the different types of vertices (extremities) and edges (adjacencies) in the following manner. The framed vertices in Figure 3a are called *ports*, and edges from $\pi_1 \cup \pi_2$ that connect them are called *port* edges. We use the same names also for other (extant or ancestral) adjacencies that are parallel to these.

Each port consists of two outer extremities called *corners*, and the *middle* vertex between them. The sets of all ports, corners, and middle vertices are denoted by P, C, and M, respectively ($P = C \cup M$). The set of *intermediate* extremities located between ports of vertex gadgets is denoted by I.

The double edges and the two vertices at the top of Figure 3b are *auxiliary:* they just complete the matchings into perfect matchings.

As the edge gadget without auxiliary and port edges is reminiscent of a *ladder*, we use the following terminology (see Fig. 3b). The red-green double adjacencies are the *rungs* and the blue adjacencies are the *rails* of the ladder. Again, we use the same name for parallel adjacencies. The set of auxiliary extremities is denoted by A, and the set of ladder extremities is denoted by L.

We say that uv is an X–Y-edge if $u \in X$ and $v \in Y$ (X and Y do not have to be disjoint); an X-edge is any edge uv such that $u \in X$ or $v \in X$.

In the proof of the normal form lemma, we will gradually transform a given solution α_1, α_2 by exchanging some of the adjacencies in the solution for other adjacencies. The method is analogous to improving a given matching by an augmenting path. An *α_i-alternating* cycle is a cycle where edges belonging to α_i and edges not belonging to α_i alternate. We will say that C_1, C_2 is a *non-negative pair of cycles* for the solution α_1, α_2, if C_i is an α_i-alternating cycle and exchanging the matched and the unmatched edges of C_i in α_i (for $i = 1, 2$) does not decrease the score: $S(\alpha_1 \oplus C_1, \alpha_2 \oplus C_2) \geq S(\alpha_1, \alpha_2)$. One of the cycles may be empty, in which case we simply say that C_1 or C_2 is a *non-negative cycle*, and if the exchange in fact increases the score, we may speak of an *augmenting* pair of cycles (or an augmenting cycle).

In the figures that follow, we will color adjacencies of α_2 blue and adjacencies of α_1 red, green, or yellow. We use red or green for edges in the vertex gadgets that are shared with π_1 or π_2, respectively (this corresponds to choosing the red or green color in the cubic max-cut problem). We use yellow for the other edges. We use straight lines for the actual adjacencies and wavy lines for the suggested adjacencies in non-negative cycles that should be included instead.

In the proof, we will often say *''we may suppose that the solution has property $\mathcal{P}$''* as a shorthand for a more precise (and longer) statement:

> *Given any solution α_1, α_2, we can transform it to a solution α_1', α_2' with $S(\alpha_1', \alpha_2') \geq S(\alpha_1, \alpha_2)$ having property $\mathcal{P}$ in polynomial time; in particular, if α_1, α_2 is an optimal solution, α_1', α_2' is also optimal, with property $\mathcal{P}$. From now on, we will assume that the solution has property $\mathcal{P}$.*

With this terminology, we may rephrase the normal form lemma more succinctly as follows: *We may suppose that solutions of the instances obtained by reduction from cubic max-cut as described above have all adjacencies supported.*

5.3. Proof of the normal form lemma

First, we focus on the adjacencies that the ancestors α_1 and α_2 have in common. We will show that these may be assumed to be at least weakly supported.

Proposition 1. *We may suppose that all red-green double edges (auxiliary adjacencies and rungs) are matched in α_1 and all blue double edges (auxiliary adjacencies) are matched in α_2, i.e., $\pi_1 \cap \pi_2 \subseteq \alpha_1$ and $\pi_3 \cap \pi_4 \subseteq \alpha_2$.*

Proof. We can alternately replace genome α_1 or α_2 by the median of its neighbors in the phylogenetic tree until we converge to a local optimum. As we have already proved in the previous section, we may assume that a median contains all adjacencies occurring at least twice. ■

Proposition 2. *We may suppose that α_1 and α_2 do not contain unsupported M-edges. In other words, we may suppose that in both α_1 and α_2, one of the edges in each port is chosen.*

Proof. Let $x \in M$. First, consider the case that $xy_1 \in \alpha_1$ and $xy_2 \in \alpha_2$ are both unsupported. Let p be a neighboring corner vertex. Whereas xy_1 and xy_2 contribute at most $+1$ to the score (if $y_1 = y_2$), common adjacency xp would contribute $+3$. Let pz_1 and pz_2 be the actual adjacencies in α_1 and α_2; either $z_1 \neq z_2$, or $z_1 = z_2$ and one of the adjacencies is unsupported. Either way, these two edges contribute at most $+2$ to the score; so xpz_1y_1x and xpz_2y_2x is a non-negative pair of cycles and we can exchange the edges.

Similarly, if one ancestor contains a port edge xp and the other one adjacencies pz and unsupported xy, then $xpzyx$ is a non-negative cycle. ■

Proposition 3. *We may suppose that all L-edges are weakly supported—they are ladder edges.*

Proof. In α_1, all L-edges are the rung edges by Proposition 1 and are supported. Consequently, the contribution of any L-edge in α_2 that is not even weakly supported is zero. Let $\ell_1 x \in \alpha_2$ be such an edge. Let $\ell_1\ell_2$ be the middle rail edge and let $\ell_2 y$ be the adjacency in α_2. If $\ell_2 y$ is not weakly supported, $\ell_1\ell_2 yx\ell_1$ is an augmenting cycle. Otherwise, if $\ell_2 y$ is a rail edge, it contributes $+1$ to the score and $\ell_1\ell_2 yx\ell_1$ is a non-negative cycle.

The last case is that $\ell_2 y$ is a rung edge contributing $+1$ to the score. Let $\ell_3 = y$, let $\ell_3\ell_4$ be the other middle rail edge, and let $\ell_4 z$ be the adjacency in α_2. Again, if $\ell_4 z$ is unsupported, $\ell_1\ell_2\ell_3\ell_4 zx\ell_1$ is an augmenting cycle; otherwise it is a rail edge and the cycle is non-negative.

It is easy to check that with each non-negative pair of cycles, we get rid of an L-edge that is not weakly supported, unless we improve the score, which may be done only $O(n)$ times. In the process, we may introduce unsupported C-edges, which is acceptable and we will deal with them next. ■

Proposition 4. *We may suppose that there are no common C-edges other than port edges.*

Proof. Let xb be a common C–C-edge in $\alpha_1 \cap \alpha_2$. In the proof, we will use the notation introduced in Figure 5. From what we have proved so far, we may assume that α_1 contains the rung edges $\ell_a\ell_b$ and $\ell_c\ell_d$ (Proposition 1), am_1 is a common adjacency of α_1 and α_2, and either m_2c or m_2d is included in α_2 (Proposition 2).

First, assume the latter case that $m_2d \in \alpha_2$ (Fig. 5a,b). As the L-edges are weakly supported, either $\ell_b\ell_c \in \alpha_2$ (Fig. 5a) or both $\ell_a\ell_b$ and $\ell_c\ell_d$ belong to α_2 (Fig. 5b). In either case, we can add ladder edges to form an alternating b–c-path with score $+1$ that will be a part of our non-negative pair of cycles.

Let cz be an adjacency in α_2. As m_2 and ℓ_c are already matched to different vertices, cz is unsupported. Now, either $cz \notin \alpha_1$ and $xb \ldots czx$ is a non-negative cycle (see Fig. 5a), or cz is a common edge and we will also have to exchange some edges in α_1. In particular, $xbczx$ and $xb \ldots czx$ are a non-negative pair of cycles (see Fig. 5b).

Similarly, we can prove the other case when $m_2c \in \alpha_2$; the non-negative cycle pairs are depicted in Fig. 5c,d. It can be easily checked that the proof also works when extremities x and b belong to the same edge gadget (in this case, x coincides with c or d, and b coincides with z). A C–C-edge connecting two corners of a single port is ruled out by Proposition 2.

Note that if α_1 and α_2 have a common z-edge (Fig. 5b,d), we may create a new common unsupported C–C-edge xz. However, the number of common unsupported C–C-edges is decreased by 1 in all cases. ■

Corollary 1. *We may suppose that all the common adjacencies of the ancestors α_1 and α_2 are weakly supported: $\alpha_1 \cap \alpha_2 \subseteq \Pi$. More specifically, we may suppose that the only common adjacencies are port edges and rung edges. Consequently, each unsupported adjacency except for rung edges in α_2 contributes zero to the score.*

We say that α_1 is *uniform* at a vertex gadget, if all the port edges in the gadget have the same color (they all agree with either the π_1 edges or the π_2 edges). Next, we prove that α_1 may be assumed uniform at all gadgets. Such an ancestor α_1 directly corresponds to a cut in G.

Here, we use the fact that G is cubic: Imagine that G was a complete bipartite graph $K_{n,n}$ with one more vertex connected to all the other vertices. Then our reduction would not work, because the optimal ancestors would color one bipartition red, the other green, and the extra vertex half green, half red (i.e., half of the ports would be green and the other half red).

First, let us characterize what the non-uniform gadgets look like.

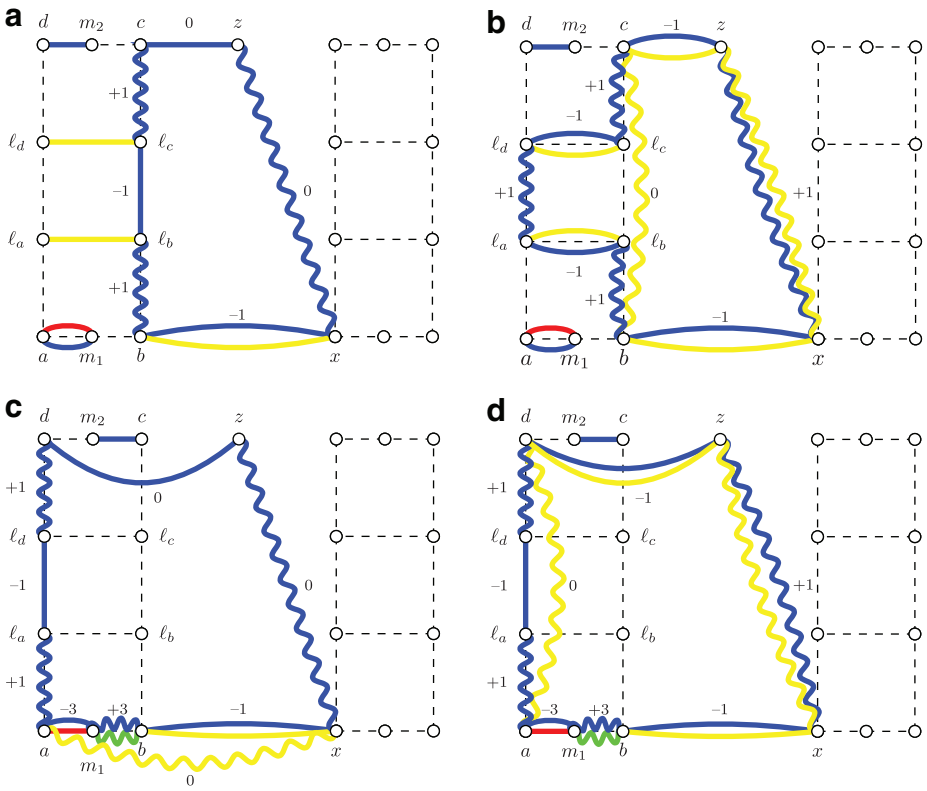

FIG. 5. Different cases that arise when disposing of unsupported common C–C-edges. The dashed edges represent the underlying edge gadgets; adjacencies of α_2 are blue, and adjacencies of α_1 are yellow, red, and green. Wavy lines are the new suggested adjacencies that should be exchanged for the present ones in the non-negative cycles. **(a)** Case 1: α_2 contains m_2d and $cz \notin \alpha_1$. **(b)** Case 2: α_2 contains m_2d and $cz \in \alpha_1 \cap \alpha_2$. **(c)** Case 3: α_2 contains m_2c and $dz \notin \alpha_1$. **(d)** Case 4: α_2 contains m_2c and $dz \in \alpha_1 \cap \alpha_2$.

Proposition 5. *We may suppose that the following statements are equivalent:*

- *Genome α_1 is not uniform at a vertex gadget.*
- *There is one unsupported I-edge in α_1 incident to the vertex gadget.*
- *There is one unsupported C-edge in α_1 incident to the vertex gadget.*

Proof. Let α_1 be nonuniform at a vertex gadget. Without loss of generality, let two of the port edges be green and one be red (see Fig. 6a). Denote r the red and g_1 and g_2 the green edges, such that g_1 is closer to r (as in Fig. 6a). The edge incident to the intermediate extremity between r and g_1 is an unsupported I-edge.

Obviously, if two *neighboring* extremities in a vertex gadget are incident with unsupported edges, there is an augmenting cycle, so we may suppose that the intermediate edge between g_1 and g_2 is green and one of the intermediate edges e or f in Figure 6a belongs to α_1; the other corner has an unsupported C-edge.

Conversely, if there is an unsupported I-edge or C-edge, the neighboring ports cannot have edges of the same color (this would imply two neighboring extremities with unsupported edges in α_1). ■

Now we are ready to prove the normal form lemma.

Proposition 6. *We may suppose that in each vertex gadget, the port edges of α_1 are either all red or all green. Thus, we may suppose that all adjacencies in α_1 are supported: $\alpha_1 \subseteq \pi_1 \cup \pi_2$.*

Proof. We prove that for each vertex gadget, we may simply look at the three port edges and choose the color by majority vote. In the previous proposition, we have shown that nonuniform gadgets have exactly two unsupported edges so they form cycles as in Figure 6b. Figure 6c shows the non-negative

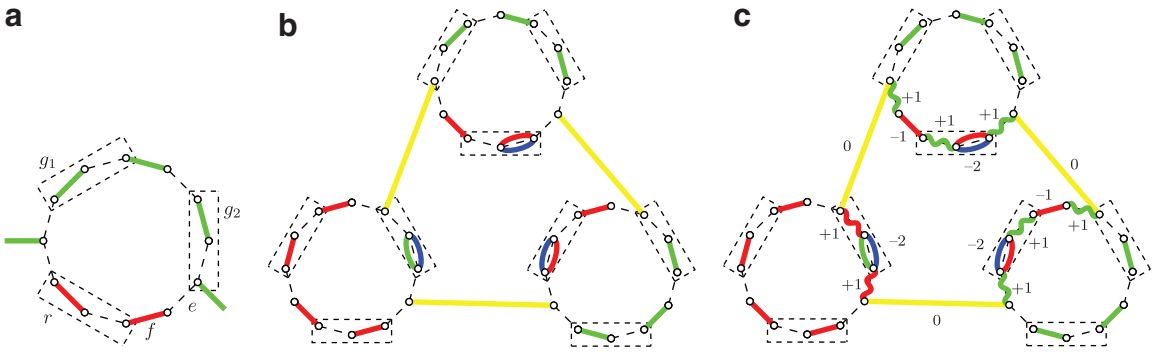

FIG. 6. Nonuniform ancestors at a vertex and a way how to remedy them. **(a)** A nonuniform ancestor α_1 at a vertex gadget. **(b)** The nonuniform gadgets are connected by unsupported *I*-edges and *C*-edges. **(c)** A non-negative cycle used for disposing of nonuniform gadgets and unsupported edges in α_1.

cycle that we get by including the edges decided by majority vote. In each vertex gadget, we may lose 1 point for switching the port edge (if this was a common edge), but we get 1 extra point for increasing the number of supported edges. ■

Proposition 7. *We may suppose that all adjacencies in α_2 are supported: $\alpha_2 \subseteq \pi_3 \cup \pi_4$.*

Proof. The only remaining unsupported edges in α_2 are the rung edges and *C*–*C*-edges. If α_2 contains a rung edge, it must in fact contain both rung edges, and in the adjacent ports, only one corner is covered by a port edge. Thus, the edge gadget is incident to two unsupported *C*–*C*-edges.

Conversely, it is easy to see that if α_2 contains a *C*–*C*-edge, in the incident edge gadgets, α_2 contains either both rung or both middle rail edges and there are *C*–*C*-edges incident to the corners of the opposite ports. So the edge gadgets together with the unsupported *C*–*C*-edges form cycles, and all the rung edges are in these edge gadgets (see Fig. 7).

In each edge gadget, we can join the two corners by a non-negative alternating path (see Fig. 7); we can lose 1 point for destroying a common adjacency of α_1 and α_2, but we gain 1 point for increasing the number of supported edges in α_2. By exchanging edges along these cycles, we fix both the unsupported *C*–*C*-edges and rung edges. ■

This concludes the proof of the normal form lemma, and thus also the proof of NP-hardness and APX-hardness of the breakpoint quartet problem.

6. CONCLUSION

In this article, we have settled several open problems concerning the computational complexity of different rearrangement problems in the breakpoint models. There are at least three intriguing questions in

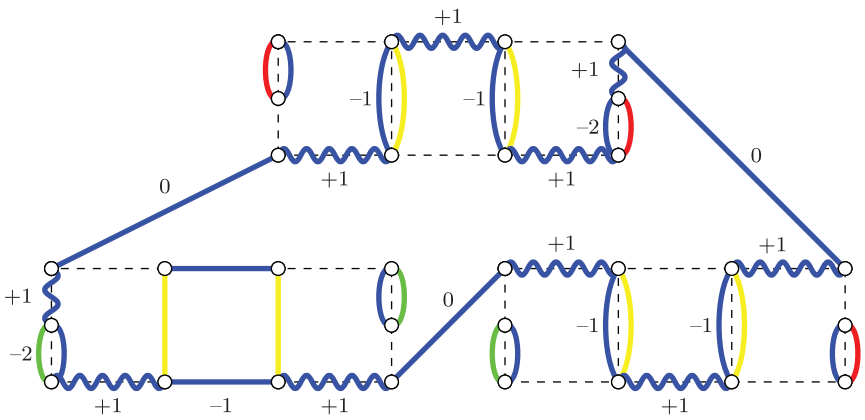

FIG. 7. Example of three edge gadgets connected in a cycle by unsupported *C*–*C*-edges. We can join two corners with unsupported *C*–*C*-edges in an edge gadget by a non-negative path. Note that we also get rid of the blue rung edges in the top and right edge gadgets at the same time.

this area that remain open. The first two are of theoretical interest and are related to approximability of the small phylogeny problem; the third question is more practical:

1. How well can we approximate small phylogeny? For example, the breakpoint quartet problem can be easily formulated as an integer linear program (we can use different variables for the edges present only in α_1, only in α_2, and in the intersection $\alpha_1 \cap \alpha_2$). Its relaxation might lead to an algorithm with a good approximation ratio.
2. In the Steinerization approach to ancestral reconstruction, we repeatedly replace the ancestral genomes by medians of genomes in the neighboring nodes of the tree until we converge to a local optimum. Despite the fact that this is the most common approach to ancestral reconstruction (also in the other models) and that preliminary experiments with simulated data suggest that this heuristic performs very well, no guarantees are known for the method (in any model).
3. Finally, the motivation behind the general breakpoint model is that we can solve the median problem in polynomial time. Using the Steinerization method, we can also get very good solutions of the small phylogeny problem rapidly. The question is: Are these solutions useful in practice? Are they biologically plausible? Or can we adjust them and use them as starting points in more complicated models?

ACKNOWLEDGMENTS

The author would like to thank Broňa Brejová and Tomáš Vinař for many constructive comments. The research of Jakub Kováč is supported by VEGA grant 1/1085/12 and Comenius University grant UK/310/2013. A preliminary version of this article was presented on the RECOMB-CG 2011 satellite workshop.

DISCLOSURE STATEMENT

The author declares that no competing financial interests exist.

REFERENCES

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

Alimonti, P., and Kann, V. 1997. Hardness of approximating problems on cubic graphs. *Proc. CIAC*, 288–298.

Biedl, T.C. 2001. Linear reductions of maximum matching. *Proc. SODA*, 825–826.

Bryant, D. 1998. The complexity of the breakpoint median problem. Technical Report CRM-2579. Centre de Recherches Mathématiques, Université de Montréal.

Caprara, A. 2003. The reversal median problem. *INFORMS J. Comput.* 15, 93.

Dees, J. 2009. Simultaneous matchings in dynamic graphs student research project. Universität Karlsruhe, Karlsruhe, Germany.

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

Fertin, G., Labarre, A., and Rusu, I. 2009. *Combinatorics of Genome Rearrangements*. The MIT Press, Cambridge, MA.

Gabow, H. 1973. Implementation of algorithms for maximum matching on nonbipartite graphs Ph.D. thesis. Stanford University, Stanford, CA.

Gabow, H. 1990. Data structures for weighted matching and nearest common ancestors with linking. *Proc. SODA*, 434–443. Society for Industrial and Applied Mathematics.

Gabow, H., and Tarjan, R. 1991. Faster scaling algorithms for general graph matching problems. *J. ACM* 38, 815–853.

Kováč, J., Warren, R., Braga, M.D.V., and Stoye, J. 2011. Restricted DCJ model: rearrangement problems with chromosome reincorporation. *J. Comput. Biol.* 18, 1231–1241.

Lawler, E. 1976. *Combinatorial Optimization: Networks and Matroids*. Holt, Rinehart and Winston, New York.

Micali, S., and Vazirani, V.V. 1980. An $O(\sqrt{|V||E|})$ algorithm for finding maximum matching in general graphs. *Proc. FOCS*, 17–27. IEEE Computer Society.

Mixtacki, J. 2008. Genome halving under DCJ revisited. *Computing and Combinatorics*, 276–286.

Pe'er, I., and Shamir, R. 1998. The median problems for breakpoints are NP-complete. *Electronic Colloquium on Computational Complexity (ECCC)*, Report 71.

Plesník, J. 1979. The NP-completeness of the Hamiltonian cycle problem in planar digraphs with degree bound two. *Inf. Process. Lett.* 8, 199–201.

Sankoff, D., and Blanchette, M. 1998. Multiple genome rearrangement and breakpoint phylogeny. *J. Comput. Biol.* 5, 555–570.

Tannier, E., Zheng, C., and Sankoff, D. 2009. Multichromosomal median and halving problems under different genomic distances. *BMC Bioinformatics* 10, 120.

Warren, R., and Sankoff, D. 2009. Genome halving with double cut and join. *J. Bioinform. Comput. Biol.* 7, 357–371.

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

Address correspondence to:
Jakub Kováč
Department of Computer Science
Comenius University
Mlynská Dolina
842 48 Bratislava
Slovakia

E-mail: kuko@ksp.sk