Information and Computation 275 (2020) 104584

Contents lists available at ScienceDirect

Information and Computation

www.elsevier.com/locate/yinco

Breakpoint distance and PQ-trees ☆

Haitao Jiang [a], Hong Liu [b], Cedric Chauve [c], Binhai Zhu [d,*]

[a] School of Computer Science and Technology, Shandong University, China
[b] School of Software, Shandong University, China
[c] Department of Mathematics, Simon Fraser University, 8888 University Drive, Burnaby, BC V5A 1S6, Canada
[d] Gianforte School of Computing, Montana State University, Bozeman, MT 59717-3880, USA

ARTICLE INFO

Article history:
Received 23 April 2014
Received in revised form 2 June 2017
Accepted 24 March 2020
Available online 3 April 2020

Keywords:
Computational genomics
PQ-trees
Ancestral genome reconstruction
Algorithms
NP-hardness

ABSTRACT

The PQ-tree is a fundamental data structure that has also been used in comparative genomics to model ancestral genomes with some uncertainty. To quantify the evolution between genomes represented by PQ-trees, in this paper we study two fundamental problems of PQ-tree comparison motivated by this application. First, we show that the problem of comparing two PQ-trees by computing the minimum breakpoint distance among all pairs of permutations generated respectively by the two considered PQ-trees is NP-complete for unsigned permutations. Next, we consider a generalization of the classical Breakpoint Median problem, where an ancestral genome is represented by a PQ-tree and $p \geq 1$ permutations are given and we want to compute a permutation generated by the PQ-tree that minimizes the sum of the breakpoint distances to the p permutations (or k). We show that this problem is also NP-complete for $p \geq 2$, and is fixed-parameter tractable with respect to k for $p \geq 1$.

1. Introduction

The PQ-tree is a fundamental data structure in computer science. First invented by Booth and Lueker as a tool to verify whether a matrix has the consecutive ones property [5], it has numerous applications: for example, recognizing interval graphs, testing whether a graph is planar, and creating a contig map from DNA segments [5,1,18]. In short, a PQ-tree on the set $\Sigma = \{1, \ldots, n\}$ is a plane rooted tree with three kinds of nodes: P-nodes, Q-nodes and leaves, with n leaves labeled by Σ (no two leaves can have the same label). A fundamental feature of PQ-trees is that a given PQ-tree can encode in linear space a possibly exponential number of permutations.

Since a decade ago, PQ-trees have been used to represent extinct ancestral genomes from a set of extant genomes represented by permutations on the same set of markers (see [7] and references therein). A PQ-tree representing an extinct ancestral genome generates possible marker orders that accounts for some uncertainty regarding the order of some markers along the ancestral chromosomes [16,19]. Note that some other ways to account for uncertainty or contradictory information have been defined, such as partial orders [24], but not in the context of ancestral genomes (possibly due to the lacking of the corresponding real datasets).

☆ This research is partially supported by National Natural Science Foundation of China under project 61628207, 61732009 and 61872427, by National Science Foundation grant DMS-0918034, by Natural Science Foundation of Shandong Province (China) under project ZR2017MF006.

* Corresponding author.
E-mail addresses: htjiang@sdu.edu.cn (H. Jiang), hong-liu@sdu.edu.cn (H. Liu), cedric.chauve@sfu.ca (C. Chauve), bhz@montana.edu (B. Zhu).

https://doi.org/10.1016/j.ic.2020.104584
0890-5401/

Once the internal nodes of a phylogenetic tree are each labeled with a PQ-tree representing the corresponding extinct genome, a natural question is to use this information to infer quantitative properties on the evolution that generated the observed extant genomes. For branches linking two internal nodes in the tree, this amounts to quantify the similarity between these two PQ-trees. We consider here the breakpoint distance [23]. Following the previous works on comparing structures generating several permutations, we consider the Minimum-Breakpoint-Permutation from PQ-Trees (MBP-PQ) Problem: given two PQ-trees T_1 and T_2, find a permutation s_1 generated by T_1 and a permutation s_2 generated by T_2 such that the breakpoint distance between s_1 and s_2 is minimum. We show that, as for partial orders [13,4], this problem is NP-complete. Next, we consider the restricted problem where T_2 degenerates to a single permutation, that we call One-Sided MBP-PQ, and we show that this problem is fixed-parameter tractable (FPT), with parameter being the optimal breakpoint distance. (The algorithm is only practical when this parameter is relatively small, which very much holds for all FPT algorithms.) We show that the same result holds for the more general *median* problem that considers p permutations $\{s_1, \ldots, s_p\}$ and a PQ-tree T and asks for a permutation s generated by T that minimizes the sum of the p breakpoint distances between s and each permutation in $\{s_1, \ldots, s_p\}$, that we call the p-Minimum-Breakpoint-Median from PQ-Tree (p-MBM-PQ) Problem. As far as we know, our FPT algorithm is only the second occurrence of an FPT result for hard median problems, after [14].

The p-MPM-PQ problem generalizes naturally the classical Breakpoint Median Problem, by imposing constraints on the possible medians, at least for permutations that represent uni-chromosomal genomes. In this paper, we prove that 2-MPM-PQ is NP-complete, together with the known results that the Breakpoint Median problem is NP-complete for at least three permutations [6,20], this implies that p-MPM-PQ is NP-complete for $p \geq 2$. (We comment that an earlier version of this paper appeared in CPM'10 [17], but the NP-completeness result for p-MPM-PQ, with $p \geq 2$, was not there and is a completely new result in this journal version.)

This paper is organized as follows. In Section 2, we give some definitions, state precisely the problems we attack and motivate them with applications in comparative genomics. In Section 3, we prove that MBP-PQ is NP-complete. In Section 4, we prove that 2-MBP-PQ is NP-complete. In Section 5, we show that several special, but practically meaningful, cases are in FPT. Finally, in Section 6, we conclude the paper with several open questions.

2. Preliminaries

2.1. Permutations, breakpoints and medians

As this paper is based on computational biology, our formal definitions will follow the biological convention. On the other hand, it can be seen by the end of this section that chromosome/genome/markers/telomeres can be interchangeable with permutation/strings/alphabet/delimiters.

Genomes with unique gene content are encoded using permutations on an alphabet of genome markers. Let Σ be such an alphabet of n markers. A *uni-chromosomal permutation* is a permutation on Σ. Given a permutation s, represented either in the form of *abcde* or $\langle a, b, c, d, e\rangle$, an *adjacency* a, b is composed of two markers that form a substring in s, either as *ab* or *ba*. A *linear* permutation of n markers contains then $n-1$ adjacencies. From now on, we omit the term linear and consider that by default every permutation is linear. The two extremities/delimiters, i.e., the leftmost and rightmost elements, of a permutation are called *telomeres*. A *multi-chromosomal* permutation having k chromosomes is a set of k permutations on k disjoint subsets of Σ. It then contains $n-k$ adjacencies and $2k$ telomeres.

Given two permutations s_1 and s_2, over the same alphabet Σ, we say that *ab* forms a *common adjacency* if *ab* or *ba* is a substring in both s_1 and s_2. Otherwise, if *ab* appears in s_1 and neither *ab* nor *ba* appear in s_2, then we say that *ab* forms a *breakpoint*. A marker a is a *common telomere* to s_1 and s_2 if it is a telomere in both permutations. We denote by $a(s_1, s_2)$ (resp. $t(s_1, s_2)$) the number of common adjacencies (resp. telomeres) between s_1 and s_2. The *breakpoint distance* between s_1 and s_2 is defined, as in [22], by the following formula: $d_b(s_1, s_2) = n - a(s_1, s_2) - t(s_1, s_2)/2$. Note that when s_1 and s_2 are uni-chromosomal permutations, it is common to delimit them by two new markers that become telomeres, and the distance formula, that we will use in this case, is $d_b(s_1, s_2) = n - 1 - a(s_1, s_2)$, which is the number of breakpoints between s_1 and s_2. In both cases, the breakpoint distance can obviously be computed in linear time.

We show a simple example for uni-chromosomal permutations. Let $s_1 = 12345$ and let $s_2 = 43512$. Then the common adjacencies between s_1 and s_2 are 12 and 34. The breakpoint distance is two, i.e., $d_b(s_1, s_2) = 5 - 1 - 2 = 2$. (We could add delimiters, like two #'s, at the ends of s_1 and s_2. That will change $d_b(s_1, s_2)$ to 4, as the length n is increased by 2.) We comment that in this case an unsigned permutation s is considered the same as its reversal, e.g., if $s = 12345$, then s is considered the same as 54321 in terms of calculating the breakpoint distances.

Given p permutations $\{s_1, \ldots, s_p\}$, the Breakpoint Median Problem asks for a permutation s that minimizes $\sum_{i=1}^{p} d_b(s_i, s)$. This problem has been shown to be NP-hard for $p \geq 3$ [6,20].

In many situations the markers in input genomes are signed (with either $+$ or $-$ signs), where the sign indicates which of the two DNA strands that marker is located. To handle signed markers in permutations, we use the same idea as in [15]: we double the number of markers and for marker i, we represent it with the two consecutive markers $(2i-1)$ $(2i)$, and for marker $-i$ we represent it with $(2i)$ $(2i-1)$. Common adjacencies and telomeres can then be described as common adjacencies for the corresponding unsigned permutations.

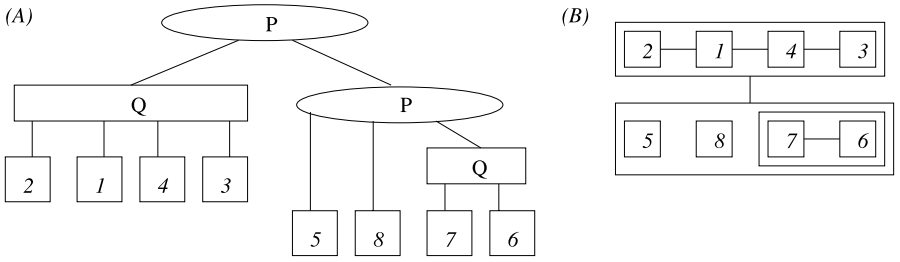

Fig. 1. (A) A PQ-tree T. $\langle 2,1,4,3,7,6,8,5\rangle$ and $\langle 3,4,1,2,5,6,7,8\rangle$ are permutations generated by this PQ-tree, but not $\langle 1,2,3,4,5,6,7,8\rangle$ as 1 has to be adjacent to 4 because they are adjacent siblings in a Q-node. (B) A graph representation G of T.

2.2. PQ-trees

Formally, a PQ-tree for unsigned permutations is a plane (ordered) tree with internal nodes that can be either P-nodes or Q-nodes (P-nodes and Q-nodes, which are conventionally oval and rectangular respectively, have at least 2 children). (Note that when a P-node has only 2 children, it can also be considered as a Q-node.) Reading the leaves of a PQ-tree in a post-order traversal gives a permutation called the *signature* of this PQ-tree. The operations of reordering the children of a P-node in an arbitrary way and reversing the children of a Q-node (and mirroring the corresponding subtrees) are called *allowed operations*. These operations define an equivalence relation between PQ-trees: two PQ-trees are equivalent if and only if we can transform one into the other by a sequence of allowed operations. The set of uni-chromosomal permutations *generated* by a given PQ-tree is the set of the signatures of all the PQ-trees of its equivalence class. See Fig. 1(A) for an illustration of PQ-trees and generated uni-chromosomal permutations, while Fig. 1(B) shows a natural graph representation for the PQ-tree.

When dealing with multi-chromosomal permutations, we assume that the root of the considered PQ-tree T is a P-node. The set of multi-chromosomal permutations from a PQ-tree is defined as follows: a multi-chromosomal permutation s with k chromosomes is generated by a PQ-tree T if and only if there exists a uni-chromosomal permutation s' generated by T such that, discarding $k-1$ adjacencies in s' formed of markers that belong to subtrees rooted at different children of the root of T results in s. We denote the number of permutations generated by a PQ-tree T by $P(T)$, assuming the context makes it clear if they are uni-chromosomal or multi-chromosomal.

PQ-trees for signed permutations have the additional constraint that, for every i, the leaves $2i$ and $2i-1$ are consecutive siblings of a Q-node. This follows directly from the way a signed marker i is converted into two unsigned markers.

2.3. Problem statements

We now formally state the problems we will investigate in this paper. Each of them has four different versions, depending on whether the considered permutations are uni-chromosomal or multi-chromosomal, and signed or unsigned. We mainly focus on uni-chromosomal unsigned permutations, and will only touch on handling signed permutations in Section 5.

Minimum Breakpoint Permutations from PQ-trees (MBP-PQ):
Input: PQ-trees T_1 and T_2 over the same set of n markers, integer K.
Question: Can T_1 and T_2 generate permutations s_1 and s_2 respectively such that $d_b(s_1,s_2)\leq K$?

For convenience, if the answer to the above question is affirmative, we also say that the breakpoint distance between T_1 and T_2 is at most K. The **One-Sided MBP-PQ** Problem is the special case where T_2 degenerates to a single permutation called s_2. It is a special case of a more general problem, that generalizes the classical Breakpoint Median Problem.

p-Minimum Breakpoint Median from PQ-tree (p-MBM-PQ):
Input: A PQ-tree T and p permutations $s_1,\ldots,s_p$ over the same set of n markers, integer K.
Question: Can T generate a permutation s such that $\sum_{i=1}^{p} d_b(s,s_i)\leq K$?

2.4. FPT algorithms

An FPT (Fixed-Parameter Tractable) algorithm for a decision problem Π with parameter value p is an algorithm which solves the problem in $O(f(p)n^c)=O^*(f(p))$ time, where f is any function depending only on p, n is the input size and c is some fixed constant not related to p. For convenience we also say that Π is in FPT. More details on FPT algorithms can be found in [10].

2.5. Previous results

If T is a PQ-tree generating all possible permutations, the p-MBM-PQ Problem is equivalent to the classical Breakpoint Median Problem described above, which is NP-hard, for signed or unsigned, uni-chromosomal or multi-chromosomal permutations [6,20,22]. In the uni-chromosomal case, even when the median is constrained to have only adjacencies that

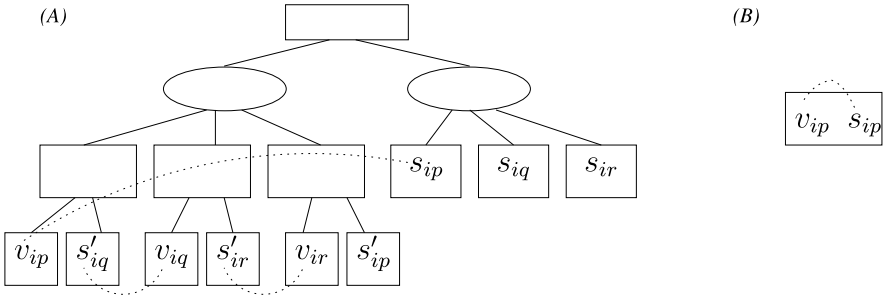

Fig. 2. The subtree F_i. In (A) and (B) the dotted arcs indicate the corresponding adjacencies. (A) shows the construction that v_i appears three times in S. (B) shows the case when v_i appears only once in S.

appear in at least one of the genomes s_i, the problem is NP-hard [6]. This implies immediately that the p-MBM-PQ Problem is NP-hard, for $p \geq 3$, in all cases. (For the unsigned uni-chromosomal case, it is straightforward to construct a 2-level PQ-tree which generates only s_i or its reversal, with the root being a Q-node. Other cases can be handled similarly.)

The MBP-PQ Problem, which we prove to be NP-complete in the next section for unsigned permutations, can be solved by an FPT algorithm whose parameter is $t = P(T_1) \cdot P(T_2)$. The reason is that it is easy to list all permutations generated by T_1 and T_2 in polynomial time and examine each pair of permutations to compute the breakpoint distance. However, $P(T)$ can be exponential for a PQ-tree T with a P-node of large degree, and it is at least exponential in the number of Q-nodes, as each Q-node can be reversed to generate a new signature. Hence, such an algorithm, even works for both signed and unsigned permutations, is probably impractical.

A similar argument applies to the p-MBM-PQ Problem, and, even in the case where T has only (say q) Q-nodes, the time complexity of the algorithm is $O(2^q n)$. In datasets where ancestral genomes are well defined and $P(T)$ is small, this approach is the most efficient, especially as it allows to consider more precise distances than the breakpoint distance. However, in some other datasets we could have a $P(T)$ too large for this approach, and this motivates our investigation of an FPT algorithm with respect to an alternative parameter. In Section 5, we describe an FPT algorithm parameterized by the value of the searched optimal solution, that is the breakpoint distance of the median permutation to the input permutations.

3. MBP-PQ is NP-complete

In this section, we prove that MBP-PQ is NP-complete for uni-chromosomal and multi-chromosomal permutations, on unsigned markers. We first consider the uni-chromosomal case. We reduce X3C (Exact Cover by 3-Sets) to MBP-PQ. Recall that the input for X3C is a set of 3-sets $S = \{S_1, S_2, ..., S_m\}$. Each set S_i contains exactly 3 elements from a base set $V = \{v_1, v_2, ..., v_n\}$, where $n = 3q$ for some integer q. The problem is to decide whether there are q 3-sets in S which cover each element in V exactly once. X3C is known to be NP-complete [11].

We first outline the general ideas and the difficulty in the proof. The general idea is to construct two PQ-trees T_1 and T_2, where T_1 is a gadget for "the element v_i appears in a 3-set S_j" and T_2 is the gadget for "the 3-set S_i contains an element x_j". In terms of generating permutations, P-nodes give the maximum amount of freedom while Q-nodes give the minimum amount of freedom. So we need to make use of the P-nodes and Q-nodes properly. (1) The first difficulty is that in a solution for X3C, each element belongs to exactly one selected 3-set; moreover, this must be encoded in the PQ-tree constructed. We enforce this by constructing a sub-tree in T_1 for each element, using both P- and Q-nodes, such that the element will appear exactly once in the final solution. (2) The second difficulty is to make sure that we construct a subtree in T_2 such that the number of possible adjacencies (non-breaking points) it could generate has a fixed pattern. We construct such a sub-tree, using no P-nodes, for each 3-set. Once these difficulties are resolved, we still need to have a match between the possible (common) adjacencies in T_1 and T_2; moreover, these matched (common) adjacencies imply a solution for X3C. Next we present the details.

MBP-PQ is obviously in NP and we focus on showing that X3C can be reduced to MBP-PQ in polynomial time. As the arguments are a bit lengthy, we will separate the proof with a few lemmas as the intermediate steps.

We first construct T_1 as follows. The root of T_1, $r(T_1)$, is a Q-node. Each non-leaf child F_i of the root $r(T_1)$ corresponds to an element v_i in V and F_i is of four levels when v_i appears at least twice in S (see Fig. 2(A)), moreover, and these children are further separated by peg markers (which are leaf nodes directly under the root $r(T_1)$). Note that peg markers are only used to separate F_i's. Let v_i appear in $S_{p_1}, S_{p_2}, ..., S_{p_t}$. For each v_i, we construct a subtree F_i as follows. The left child of $r(F_i)$ is a P-node which has t Q-nodes as children, and the contents of these Q-nodes are: $v_{i,p_1}s'_{i,p_2}$, $v_{i,p_2}s'_{i,p_3}$, ..., $v_{i,p_t}s'_{i,p_1}$. The right child of $r(F_i)$ is a P-node with t leaves: $s_{i,p_1}, s_{i,p_2}, ..., s_{i,p_t}$. Here, $v_{i,j}$ can be interpreted as "the element v_i appears in the 3-set S_j" and $s_{i,j}$ can be interpreted as "the 3-set S_i contains the element v_j". We use $s'_{i,j}$'s to help us obtain an exact number of adjacencies in T_1 and T_2. (For the purpose of clarity, in the figures, we omit the commas in the indices to use v_{ij}, s_{ij}, s'_{ij} respectively.) Intuitively, $v_{i,p_w}s_{i,p_w}$ forms an adjacency iff S_{p_w} is selected (to cover v_i) in the final X3C solution. In Fig. 2(A), note that $t = 3$.

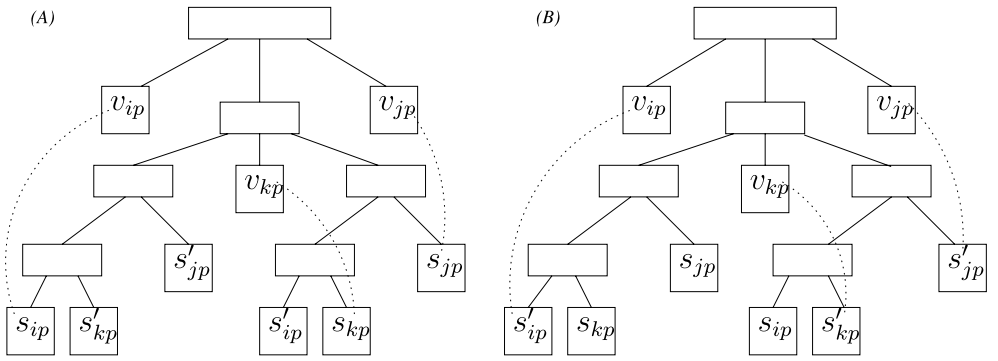

Fig. 3. The subtree H_p corresponding to $S_p = \{v_i, v_j, v_k\}$. (A) and (B) show the two different kinds of adjacencies (marked by dotted arcs).

When v_i appears in S exactly once (say, in S_p), F_i would be a Q-node with two leaves: $v_{i,p}, s_{i,p}$ (Fig. 2(B)). We will have to use some peg markers to compose new leaf nodes to restrict $s'_{i,p}$ so that it will never be adjacent to $v_{i,p}$. We will cover this special case at the end of the whole proof. At this point, we assume that each v_i appears in the 3-sets in S at least twice. We summarize the construction of F_i's with the following lemma.

Lemma 1. *F_i can generate at most one adjacency $v_{i,p_w}s_{i,p_w}$ for some $1 \le w \le t$.*

Proof. As the root of F_i has only two children, an adjacency in the form $v_{i,p_w}s_{i,p_w}$ must be formed by these two children, with the left one contributing v_{i,p_w} while the right one contributing s_{i,p_w}. □

We now construct T_2. The root of T_2 is also a Q-node. Each of the children of $r(T_2)$ is a subtree H_p with the root being a Q-node. H_p corresponds to a 3-set $S_p = \{v_i, v_j, v_k\}$. An illustration of H_p is shown in Fig. 3. Notice that H_p has five levels. We have the following lemmas.

Lemma 2. *H_p can generate exactly two sets of adjacencies in the form of $\{v_{i,p}s_{i,p}, v_{j,p}s_{j,p}, v_{k,p}s_{k,p}\}$ or $\{v_{i,p}s'_{i,p}, v_{j,p}s'_{j,p}, v_{k,p}s'_{k,p}\}$.*

Lemma 3. *T_1 and T_2 can each generate at most $3m$ adjacencies in the form of $v_{i,p}s_{i,p}$ or $v_{i,p}s'_{i,p}$.*

Proof. Following Lemma 2, T_2 can generate at most $3m$ adjacencies in the form of $v_{i,p}s_{i,p}$ or $v_{j,p}s'_{j,p}$.

Following Lemma 1, T_1 can generate n adjacencies in the form of $v_{i,p_w}s_{i,p_w}$ for some $1 \le w \le t$. The remaining $3m - n$ adjacencies can obviously be generated in the form of $v_{i,p}s'_{i,p}$. □

Lemma 4. *The input X3C instance has a valid solution if and only if T_1 and T_2 can generate $3m$ common adjacencies.*

Proof. The "only if" part is easy to prove. Assume that the instance (S, V) has a solution, let $S_p = \{v_i, v_j, v_k\}$ be in the solution. We permute the P-nodes in F_i and the Q-nodes in H_p such that $v_{i,p}s_{i,p}$ forms a common adjacency. Following Lemma 3, we can obtain $3m$ common adjacencies in T_1 and T_2.

We now prove the "if" part. Assume that T_1 and T_2 generate exactly $3m$ common adjacencies. We first show that there must be n common adjacencies in the form of $v_{i,p}s_{i,p}$. If this is not the case, say in T_2 some $v_{i,p}$ is never forming an adjacency with $s_{i,p}$, then the common adjacencies in T_1, T_2 will not reach $3m$. Symmetrically, if in T_1 one of the subtrees F_i cannot generate t adjacencies, then there is no way T_1 and T_2 can generate $3m$ common adjacencies.

Now assume that among the $3m$ common adjacencies in T_1 and T_2 there are n common adjacencies in the form of $v_{i,p}s_{i,p}$, we argue that they present exactly a corresponding solution for X3C. By the way we construct T_1, if $v_{i,p}$ forms an adjacency with $s_{i,p}$, then the adjacency implies that S_p is selected as part of the solution for the X3C instance. As we have exactly n adjacencies in the form of $v_{i,p}s_{i,p}$, each of the element appears in the X3C solution exactly once and we have a valid solution for the X3C instance (S, V). □

Theorem 5. *MBP-PQ is NP-complete for uni-chromosomal unsigned permutations.*

Proof. Now it is necessary to cover the special case when v_i appears in S exactly once. In this case we use some peg markers as leaves to bound $s'_{i,p}$ such that it will never be adjacent to $v_{i,p}$. The peg markers will be directly under the roots of T_1 and T_2, and we can order them properly so that the peg markers will not form adjacencies in T_1 and T_2. It is easy to see that we will not use more than $O(n)$ peg markers.

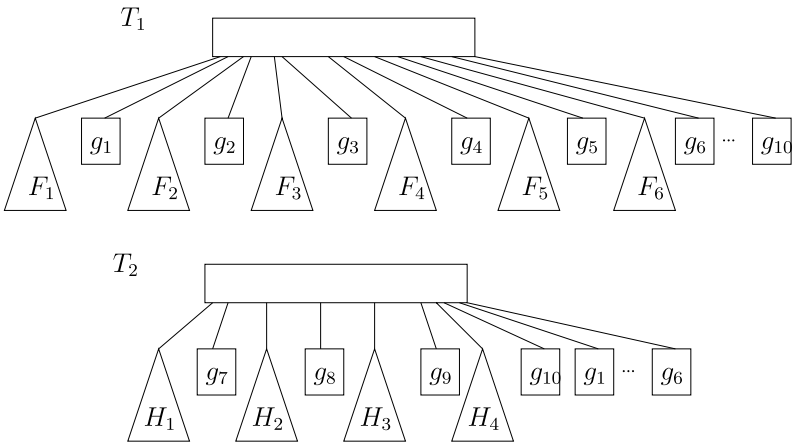

Fig. 4. The sketch of a simple example for the reduction.

Let N be the number of peg markers used in the construction. Following Lemma 4, there are $9m$ markers in T_1 and T_2. Therefore, the input X3C instance has a valid solution if and only if T_1 and T_2 can generate two permutations with $N + 6m - 1$ breakpoints.

It is clear that the whole transformation takes linear time. Hence, MBP-PQ is NP-complete. □

We show a sketch of a simple example for the above reduction in Fig. 4. The X3C instance is as follows: $V = \{v_1, v_2, v_3, v_4, v_5, v_6\}$; $S_1 = \{v_1, v_2, v_4\}$, $S_2 = \{v_2, v_4, v_6\}$, $S_3 = \{v_3, v_5, v_6\}$, $S_4 = \{v_1, v_3, v_6\}$. The solution for this X3C instance is $\{S_1, S_3\}$. As both T_1 and T_2 have over 70 nodes, we just show a sketch of T_1 and T_2, with F_i's and H_p's defined as above. We use 10 peg markers: $g_1, \ldots, g_{10}$. This is just to enforce that two markers in F_i and F_{i+1} respectively (and in H_p and H_{p+1}) cannot form any common adjacency. It is easy to check that T_1 and T_2 can generate 12 common adjacencies: $v_{11}s_{11}, s'_{14}v_{14}, v_{21}s_{21}, s'_{22}v_{22}, v_{33}s_{33}, s'_{34}v_{34}$, $v_{41}s_{41}, s'_{42}v_{42}, v_{53}s_{53}, s'_{62}v_{62}$, $v_{63}s_{63}$ and $s'_{64}v_{64}$.

We can extend the proof to the multi-chromosomal case. Given an instance (T_1, T_2) of the uni-chromosomal case, create an instance (T'_1, T'_2) by adding to T_1 (resp. T_2) a P-node root and two children Q-nodes with each 4 leaves $n+1, n+2, n+3, n+4$ (resp. $n+2, n+4, n+1, n+3$), in this order in both cases, and $n+5, n+6, n+7, n+8$ (resp. $n+6, n+8, n+5, n+7$), again in this order in both cases. There are no common telomeres in T_1 and T_2. Therefore, T_1 and T_2 has breakpoint distance K if and only if T'_1 and T'_2 has breakpoint distance $K+8$ because we add 8 markers that do not form any adjacency, neither common telomere.

Corollary 6. *MBP-PQ is NP-complete for multi-chromosomal unsigned permutations.*

With the above negative result, it is natural to ask whether one can design FPT and/or approximation algorithms for the optimization version of MBP-PQ.

4. 2-MBM-PQ is NP-complete

In this section, we show that 2-MBM-PQ is NP-complete for uni-chromosomal unsigned permutations, by a reduction from the *Hamiltonian path in cubic planar bridgeless graph* problem, which is to compute a simple path visiting all the vertices exactly once in the corresponding graph and is known to be NP-complete [11].

The general idea of the reduction is to use a PQ-tree to generate a permutation of the vertices of a given cubic planar bridgeless graph C, and use two permutations to represent all the edges of the graph. We then show that there is a Hamiltonian path in C if and only if the permutation generated by the PQ-tree contains exactly $n-1$ adjacencies, each of which corresponds to an edge of the graph.

Formally, given a cubic planar bridgeless graph $C = (C(V), C(E))$, we construct an instance of 2-MBM-PQ as follows. The PQ-tree has two layers. The root is a P-node, which has two children: a P-node and a Q-node, the P-node stores the set of vertices of $C(V)$ as leaves, while the Q-node stores some peg markers (to be defined in the following).

It remains to construct two permutations s_1 and s_2 of all the vertices in $C(V)$, as well as some peg markers, such that for each pair of vertices $v_i, v_j \in C(V)$, if $(v_i, v_j) \in C(E)$, then v_i and v_j are adjacent either in s_1 or in s_2 but not both; if $(v_i, v_j) \notin C(E)$, then they are separated either by other vertices or by a peg marker. Next, we show that for a cubic planar bridgeless graph C, this can be done in polynomial time. We review the following lemma first.

Lemma 7. *Any cubic planar bridgeless graph has a perfect matching* [21]*, which can be computed in linear time* [3].

Our algorithm to convert the edges in $C(E)$ into two permutations is as follows.

1. Compute a perfect matching M of the cubic planar bridgeless graph $C = (C(V), C(E))$. Then the graph $C' = (C(V), C(E) - M)$ is composed of disjoint cycles.
2. Start from an arbitrary vertex v_i, traverse all the vertices along the cycle in C' which v_i belongs to, and obtain a path $P = \langle v_i, \ldots, v_j \rangle$. For the vertex v_k where $(v_j, v_k) \in M$, if $v_k \in P$, then put this path P into s_1; otherwise, if $v_k \notin P$, then the path P is expanded to visit another cycle through the edge $(v_j, v_k) \in M$. Run this process iteratively until all the cycles are visited. Finally, we join these paths, using edge in M, into maximal paths which will not form any cycle.
3. Let the set of paths obtained in Step 2 be P'.
4. Put the remaining paths in C which is not covered by any path in P' as substrings for s_2, which will be separated with peg markers subsequently.

Lemma 8. *Each vertex of $C(V)$ appears exactly once in s_1.*

Proof. Since C is a cubic planar bridgeless graph and M is its perfect matching, $C' = (C(V), C(E) - M)$ is composed of disjoint cycles. By Step 2, all the cycles are covered by paths put in s_1 and each cycle would be covered exactly once. Then all the vertices of $C(V)$ will appear in s_1 exactly once. □

Let C'' be the graph obtained by deleting the edges covered by the path set P' from C.

Lemma 9. *There is no cycle in C''.*

Proof. We prove a more strict statement that each vertex of degree-2 in C'' has a neighbor of degree-1. Note that each vertex of degree-2 in C'' must be the end of some path in P'. So, if a vertex v_r of degree-2 is connected to two vertices v_s and v_t of degree-2, at least one of v_s and v_t is not in the same path with v_r, which means that the two paths could be joined together by Step 2. □

Lemma 10. *Each vertex of $C(V)$ appears exactly once in s_2.*

Proof. Since each vertex of $C(V)$ appears exactly once in s_1 (by Lemma 8), the degrees of vertices in C'' are one or two. Since there is no cycle in C'' (by Lemma 9), we can put all the vertices into s_2 along disjoint paths in C''. □

Finally we complete the reduction by handling the peg markers.

1. Separate the paths in s_1 and the paths in s_2 by distinct peg markers.
2. Construct four sub-permutations $x_1x_2x_3x_4$, $x_2x_4x_1x_3$, $x_5x_6x_7x_8$, $x_6x_8x_5x_7$, put $x_1x_2x_3x_4$ and $x_5x_6x_7x_8$ at the two ends of s_1 and s_2 respectively; put $x_2x_4x_1x_3$ on the left side of the child P-node and $x_6x_8x_5x_7$ on the right side of the child P-node.
3. Put all the other peg markers of s_2 on the left of x_1 in s_1; put all the other peg markers of s_1 on the right of x_8 in s_1; put all the other peg markers of s_1 and s_2 in between $x_2x_4x_1x_3$ and $x_6x_8x_5x_7$ in the child P-node.

Lemma 11. *The cubic planar bridgeless graph C has a Hamiltonian path if and only if the PQ-tree T can generate a permutation s, such that the number of common adjacencies between s and s_1 plus the number of common adjacencies between s and s_2 is exactly $n - 1$.*

Proof. Firstly, obviously, there will not be common adjacencies involving peg markers.

($\Rightarrow$) If the cubic planar bridgeless graph C has a Hamiltonian path, the PQ-tree can generate a permutation of the vertices in the order of the Hamiltonian path, so each pair of adjacent vertices is an edge in G. Since each edge is put in either s_1 or s_2 but not both, so the number of common adjacencies is exactly $n - 1$.

($\Leftarrow$) Assume that s can be generated from T such that the number of common adjacencies between s and s_1 plus those between s and s_2 is exactly $n - 1$. Since T can generate at most $n - 1$ common adjacencies and each adjacency corresponds to an edge in C; moreover, by Lemma 8 and Lemma 10 each vertex of C appears exactly once in s_1 and s_2, these $n - 1$ edges form a Hamiltonian path. □

We show a simple example for this reduction. Let C be a graph composed of two triangles $\{u_1, u_2, u_3\}$ and $\{v_1, v_2, v_3\}$, together with three edges $(u_i, v_i), i = 1..3$. The (u_i, v_i) edges form a perfect matching M. Following our reduction, we have

$$s_1 = \#x_1x_2x_3x_4\ u_1u_2u_3v_3v_1v_2\ x_5x_6x_7x_8,$$

and

$$s_2 = x_1x_2x_3x_4\ v_1u_1u_3\#u_2v_2v_3\ x_5x_6x_7x_8.$$

Here, # is the only peg marker. The PQ-tree T has a root, which is a P-node, with two children, one of them is a P-node and the other is a Q-node. The content of the P-node is: $\{u_1, u_2, u_3, v_1, v_2, v_3\}$. The content of the Q-node is: $\langle x_2, x_4, x_1, x_3, \#, x_6, x_8, x_5, x_7\rangle$. If we order the content of the P-node as $[u_1, u_2, u_3, v_3, v_2, v_1]$, which corresponds to a Hamiltonian path in C, we have a total of 5 common adjacencies between the obtained permutation s and s_1, s_2: u_1u_2, u_2u_3, u_3v_3, v_3v_2 and v_2v_1.

Theorem 12. *2-MBM-PQ is NP-complete for uni-chromosomal unsigned permutations.*

Proof. It is clear that 2-MBM-PQ is in NP. It is obvious that our reduction can be computed in polynomial time, from Lemma 11, 2-MBM-PQ is NP-complete for uni-chromosomal unsigned permutations. □

Since the Breakpoint Median Problem for three permutations is NP-complete, we conclude that p-MBM-PQ is NP-complete for $p \geq 3$. Together with the above theorem we have the following corollary.

Corollary 13. *p-MBM-PQ is NP-complete for $p \geq 2$ for uni-chromosomal unsigned permutations.*

5. An FPT algorithm for One-Sided MBP-PQ and p-MBM-PQ

In this section, we solve both One-Sided MBP-PQ and p-MBM-PQ with an FPT algorithm, whose parameter is the value of the optimal breakpoint distance. We first describe our algorithm for the uni-chromosomal case, then discuss its generalization to the multi-chromosomal case.

5.1. A graphical representation of PQ-trees

We first introduce a graph representation of a PQ-tree, which encodes the adjacency constraints between markers, and was used in [7] to represent ancestral genomes in a linear-like way. The graph G associated to a PQ-tree T has vertices for all nodes of T except possibly the root if it is a P-node. We call the vertices that correspond to leaves *markers*. And the vertices corresponding to the P-nodes (resp. Q-nodes) are called *super* P-nodes (resp. Q-nodes). Edges of G are defined only between pairs of markers as follows: two markers x and y define an edge (x, y) if and only if they are consecutive children of a Q-node. Similarly, edges are defined between two super-nodes which must be adjacent. Edges of G are called *black edges*. See Fig. 1(B), where black edges are solid.

We also add an additional structure on G by embedding the vertices following the recursive structure of T: the vertices of G corresponding to the children of a node are embedded into the vertex representing this node (see Fig. 1(B)). A vertex (leaf or super-node) X is *contained* in another vertex Z if $X \neq Z$ and the node corresponding to X is a descendant of the one corresponding to Z in T (hence Z is a super-node); as a consequence, all the strings generated by X are substrings of those generated by Z.

We now describe how to augment the graph representation G of a PQ-tree T using another input permutation s_1. It turns out that this will be the basis for us to handle the ancestral genome analysis when some reference permutation is given. We start with G, and then add an edge, called a *blue* edge, (x, y) in G for every adjacency xy in s_1. We denote this new graph G' (note that G' conserves the embedding structure we defined on G: only blue edges are added). The *degree* of a super-node X in G' is the number of blue edges that connects a marker inside X to a marker outside X. See Fig. 1 and Fig. 5, where black edges are solid and blue edges are dashed.

At this point, it can be seen that the One-Sided MBP-PQ Problem is closely related to the classical Minimum Path Cover Problem, where given a graph one needs to find the minimum number of vertex-disjoint paths to cover all vertices in the graph [9]. Of course, for the One-Sided MBP-PQ Problem, the underlying graph is a special hypergraph; moreover, our objective function is the minimum number of breakpoints (instead of the minimum number of paths). So the technical details will be different.

5.2. An FPT algorithm for the One-Sided MBP-PQ problem

We first state an easy lemma that describes constraints on the blue edges that can be conserved in an optimal solution of the problem. This naturally gives us an FPT algorithm using the bounded search tree method.

Lemma 14. *An optimal solution for One-Sided MBP-PQ can be obtained by performing the following operations on G'.*

1. *If a marker x is inside a Q-node Y and x has two neighboring markers in Y, then one can delete all the blue edges incident to x to obtain an optimal solution.*
2. *If a marker x is of degree greater than two, then an optimal solution can be obtained by deleting all but at most two blue edges connecting to x.*

3. *If a super-node X is of degree greater than two, then an optimal solution can be obtained by deleting all but at most two blue edges connecting to some markers inside X.*

Proof. We just argue for case 3 as the other two are easy. If X has degree greater than two, to embed X in some path, some edges incident to X must be deleted. If we allow more than two blue edges connecting to some markers inside X then it is impossible to embed X in any path. □

The outline of the algorithm One-MBP-PQ(-,-) is given separately. Let r be the maximum degree of a super-node, after all edge deletion operations at Step 1 of Lemma 14 have been performed. (If $r \leq 2$ the problem is trivially solvable. So we assume that $r \geq 3$.) The principle of the FPT algorithm is to use a bounded search tree [10] that considers the super-nodes of degree at least three and, for such a node X, conserves at most two blue edges that link a marker inside X and a marker outside X. Let K be the optimal solution value for One-Sided MBP-PQ (i.e., the number of deleted edges in U), and $f(K)$ be the size (number of nodes) of the search tree. It is sufficient to keep deleting edges such that the resulting nodes have degree at most two, so we have the following recurrence relation

$$f(K) = \begin{cases} 0 & \text{if } K = 0, \\ 1 & \text{if } K = 1, \\ \leq \binom{r}{r-2} f(K-r+2) & \text{if } K > 1. \end{cases}$$

The main recurrence can be simplified as

$$f(K) \leq \binom{r}{2} f(K-r+2) = \frac{r(r-1)}{2} f(K-r+2).$$

By expanding the recursion at most $\frac{K}{r-2}$ times, we can solve this recurrence to have

$$f(K) \leq \left(\frac{r(r-1)}{2}\right)^{\frac{K}{r-2}}.$$

Then it is easily seen that this recurrence achieves its maximum value when $r = 3$ (where the recurrence becomes $f(K) \leq 3f(K-1)$). Therefore,

$$f(K) \leq 3^K.$$

Algorithm *One-MBP-PQ(T, s_1)*
Input: PQ-tree T and a permutation s_1 over the same alphabet, and integer K.
Output: A realization of a sequence s from T s.t. the number of breakpoints between s and s_1 is at most K.
1 Compute the association graph G, using black edges, from T.
2 Augment G following the recursive structure of T.
3 Augment G into G', using blue edges, for all the adjacencies in s_1.
4 Let U be the set of blue edges to be deleted from G'.
5 While $|U| \leq K$
6.1 If a marker x with two neighbors is in a Q-node, then put all the blue edges incident to x in U and delete them from G'.
6.2 If a marker x is of degree at least three, then arbitrarily delete all but two blue edges and put the deleted edges in U.
6.3 If a super-node X is of degree at least three, then arbitrarily delete all but two blue edges from some markers inside X and put the deleted edges in U.
7 If the resulting graph is composed of paths, then return the resulting sequence s; otherwise, return 'No Solution'.

Once K blue edges are deleted from G', all we need to do is to check whether the resulting graph on Σ defined by the markers and the remaining black and blue edges is composed of paths. If after K blue edges are deleted and there is no vertex of degree at least three left, we can check whether there are still any (disjoint) cycles left (if so, then just delete some blue edges accordingly to break these cycles). If, after K blue edges are deleted and no valid solution is found, then we report 'No solution of size K'. This can be easily done in $O(n)$ time as at this point the maximum degree of any vertex is at most two. Therefore, we can use this bounded search tree method to obtain an algorithm which runs in $O(3^K n)$ time, once G' is computed.

In Fig. 5, we show a simple example for the algorithm. An example of T and s_1 is illustrated in Fig. 5(A). The augmented graph G' is shown in Fig. 5(B). The optimal solution value is $K = 1$. According to the algorithm, we will have to delete one

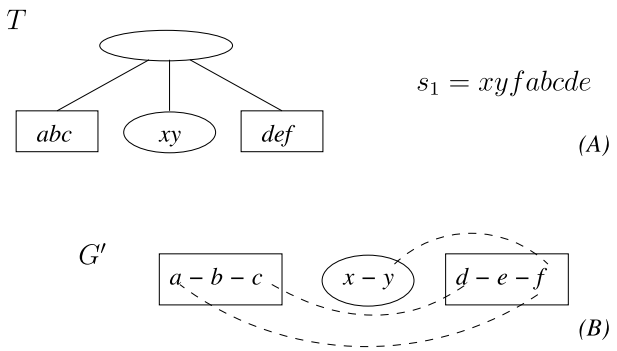

Fig. 5. An example for the FPT algorithm for One-Sided MBP-PQ.

blue (or dashed) edge in G'. The algorithm has the choice of deleting either (a, f), (y, f), or (c, d). Clearly, deleting (a, f) gives us the optimal solution with $s = abcdefyx$ and exactly one breakpoint between s and $s_1 = xyfabcde$. Deleting (y, f) or (c, d) alone both leads to infeasible solutions.

Theorem 15. *One-Sided MBP-PQ can be solved in $O(3^K n)$ time for uni-chromosomal signed and unsigned permutations, where n is the number of markers and K is the number of breakpoints in the optimal solution.*

5.3. Solving the p-MBM-PQ problem

It is easy to see that p-MBM-PQ can be solved in $O(3^K n)$ time as well. The idea is to compute the graph G for the input PQ-tree T and then add blue edges from adjacencies in s_i, for $i = 1, \ldots, p$. Now a blue edge (x, y) is weighted, with the weight corresponding to the total number of adjacencies xy or yx in s_i, for $i = 1, \ldots, p$. So such a weight can be an integer in $[1, p]$. Let this augmented (weighted) graph be G''. Then the problem is clearly equivalent to deleting blue edges from G'', with a total weight of $K' \leq K$, such that the resulting graph is composed of paths. If there are K'' such paths, then some adjacencies need to be added to transform them into a single path, and arbitrary adjacencies can be used, each contributing p to the breakpoint distance, that is $K' + p(K'' - 1)$. This leads to the following result.

Corollary 16. *p-MBM-PQ can be solved in $O(3^K n)$ time for uni-chromosomal signed and unsigned permutations.*

Note that the actual running time of the FPT algorithm we described is in general much faster than $O(3^K n)$ as any adjacency in one of the genomes s_i that is discarded following Lemma 14.(1) increases the breakpoint distance by one but is not considered in the subsequent computation. More formally, if d is the number of edges discarded due to Lemma 14.(1), the running time is in fact $O(3^{K-d} n)$. This has been confirmed in our initial computational results [17]. We can also immediately apply our algorithm to the variant where the median is constrained to contain only adjacencies that appear in at least one permutation s_i, which is also NP-hard for the classical Breakpoint Median Problem [6]. Indeed, it suffices to forbid deleting blue edges that disconnects the augmented graph, which is obviously connected at first.

5.4. Handling multi-chromosomal permutations

Here we need to account for two things: the set of generated permutations is different (larger in fact) and the breakpoint distance requires to consider common telomeres. To deal with both of these issues, we add in the augmented graph a vertex W, that represents telomeres, and a blue edge (W, a) for every telomere a in the s_i's. Then, a set of blue edges defining a valid permutation implies that once edges (W, a) are discarded, the resulting edges comprise of a set of paths. Finally, as common telomeres contribute to half the weight of common adjacencies in the breakpoint distance formula, when the bounded search discards a blue edge (W, a), it increases the distance by $1/2$ instead of 1. This proves the following result.

Corollary 17. *p-MBM-PQ can be solved in $O(3^{2K} n)$ time for multi-chromosomal signed and unsigned permutations.*

6. Conclusion

In this paper, we make the first step in comparing the similarity of PQ-trees, with application to comparative genomics. While the general MBP-PQ problem is NP-complete, we show that several interesting cases, that are relevant from an applied point of view, are in FPT, parameterized by the optimal breakpoint distance.

Our first open question is how to construct a general graph or hypergraph incorporating all the information regarding two PQ-trees T_1 and T_2. Without such a hypergraph, it seems difficult to design approximation and FPT algorithms for the

optimization version of MBP-PQ (and possibly some other ways to compare the similarity of T_1 and T_2). A related question would be to find an FPT algorithm for MBP-PQ whose parameter is the breakpoint distance. When this distance is zero, the problem is in fact easy to solve: it is easy to decide if T_1 and T_2 can generate the same permutation [5,2].

How to improve the efficiency of the FPT algorithms for One-Sided MBP-PQ and p-MBM-PQ also yields interesting questions. (In the conference version, we have shown some empirical results for One-Sided MBP-PQ [17]. But the algorithm is inefficient for many practical datasets; in fact, for some cases it took more than a week to run the code.) The only other FPT algorithm for a breakpoint median problem, described in [14], has complexity $O(2.15^K n)$, and it remains to see how the ideas used in that algorithm can be translated to the case where the median is constrained to be generated by a given PQ-tree.

Regarding p-MBM-PQ, it has been proved in [22] that the Breakpoint Median Problem for signed multi-chromosomal genomes is polynomially solvable if the median is allowed to have circular chromosomes; it can indeed be solved by a maximum weight matching algorithm. In the case of p-MBM-PQ, the corresponding problem would allow that, in the median, the leaves of one or more subtree rooted at the children of the root form a circular chromosome.

Finally, the most natural open problems consist of comparing PQ-trees and computing the median from a PQ-tree under other distances such as the DCJ (Double-Cut-and-Join) distance. However, we expect that such problems are hard too. For example, comparing two PQ-trees of height 2 (every path between a leaf and the root contains at most two edges) whose internal nodes are all P-nodes is equivalent to computing the syntenic distance [12] between two genomes represented by the gene content of their chromosomes and with no gene order information, which is an NP-hard problem that admits a factor-2 approximation [8].

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

We thank the anonymous referees for several insightful comments.

References

[1] F. Alizadeh, R. Karp, D. Weisser, G. Zweig, Physical mapping of chromosomes using unique probes, J. Comput. Biol. 2 (1995) 159–184.
[2] A. Bergeron, M. Blanchette, A. Chateau, C. Chauve, Reconstructing ancestral gene orders using conserved intervals, in: Workshop on Algorithms in Bioinformatics (WABI'04), in: Lecture Notes in Comput. Sci., vol. 3240, Springer, 2004, pp. 14–25.
[3] T. Biedl, P. Bose, E. Demaine, A. Lubiw, Efficient algorithms for Peterson's matching theorem, J. Algorithms 38 (1) (2001) 110–134.
[4] G. Blin, E. Blais, P. Guillon, M. Blanchette, N. ElMabrouk, Inferring gene orders from gene maps using the breakpoint distance, in: RECOMB Computational Genomics Workshop (RCG'06), in: Lecture Notes in Comput. Sci., vol. 4205, Springer, 2006, pp. 99–102.
[5] K. Booth, G. Lueker, Testing for the consecutive ones property, interval graphs, and graph planarity using PQ-tree algorithms, J. Comput. Syst. Sci. 13 (1976) 335–379.
[6] D. Bryant, The complexity of the breakpoint median problem, Technical Report CRM-2579, Centre de Recherches en Mathématiques, Université de Montréal, 1998.
[7] C. Chauve, E. Tannier, A methodological framework for the reconstruction of contiguous regions of ancestral genomes and its application to mammalian genome, PLoS Comput. Biol. 4 (2008) e1000234.
[8] B. DasGupta, T. Jiang, S. Kannan, M. Li, E. Sweedyk, On the complexity and approximation of syntenic distance, Discrete Appl. Math. 88 (1–3) (1998) 59–82.
[9] R. Diestel, Graph Theory, 4th edition, Springer-Verlag, 2010.
[10] R. Downey, M. Fellows, Fundamentals of Parameterized Complexity, Springer-Verlag, 2013.
[11] M. Garey, D. Johnson, Computers and Intractability: A Guide to the Theory of NP-Completeness, W.H. Freeman, 1979.
[12] V. Feretti, J. Nadeau, D. Sankoff, Original synteny, in: Combinatorial Pattern Matching (CPM'96), in: Lecture Notes in Comput. Sci., vol. 1075, Springer, 2006, pp. 159–167.
[13] Z. Fu, T. Jiang, Computing the breaking distance between partially ordered genomes, J. Bioinform. Comput. Biol. 5 (5) (2007) 1087–1101.
[14] J. Gramm, R. Niedermeier, Breakpoint medians and breakpoint phylogenies: a fixed-parameter approach, Bioinformatics 18 (Suppl 2) (2002) 128–139.
[15] S. Hannenhalli, P. Pevzner, Transforming cabbage into turnip: polynomial algorithm for sorting signed permutations by reversals, J. ACM 46 (1) (1999) 1–27.
[16] G. Jean, D.M. Sherman, M. Nikolski, Mining the semantic of genome super-blocks to infer ancestral architectures, J. Comput. Biol. 16 (9) (2009) 1267–1284.
[17] H. Jiang, C. Chauve, B. Zhu, Breakpoint distance and PQ-trees, in: Combinatorial Pattern Matching (CPM'10), in: Lecture Notes in Comput. Sci., vol. 6129, Springer, 2010, pp. 112–124.
[18] G. Landau, L. Parida, O. Weimann, Gene proximity analysis across whole genomes via PQ-trees, J. Comput. Biol. 12 (2005) 1289–1306.
[19] A. Ouangraoua, E. Tannier, C. Chauve, Reconstructing the architecture of the ancestral amniote genome, Bioinformatics 27 (19) (2011) 2664–2671.
[20] I. Pe'er, R. Shamir, The median problems for breakpoints are NP-complete, Electron. Colloq. Comput. Complex. (1998), TR-98–071.
[21] J. Peterson, Die Theorie der regulären Graphs (The theory of regular graphs), Acta Math. 15 (1891) 193–220.
[22] E. Tannier, C. Zheng, D. Sankoff, Multichromosomal median and halving problems under different genomic distances, BMC Bioinform. 10 (2009) 120.
[23] G. Watterson, W. Ewens, T. Hall, A. Morgan, The chromosome inversion problem, J. Theor. Biol. 99 (1982) 1–7.
[24] C. Zheng, A. Lennert, D. Sankoff, Reversal distance for partially ordered genomes, Bioinformatics 21 (Suppl 1) (2005) i502–i508.