Breakpoint Distance and PQ-Trees

Haitao Jiang[1,2], Cedric Chauve[3], and Binhai Zhu[1]

[1] Department of Computer Science, Montana State University, Bozeman, MT 59717, USA
bhz@cs.montana.edu

[2] School of Computer Science and Technology, Shandong University, China
htjiang@cs.montana.edu

[3] Department of Mathematics, Simon Fraser University, 8888 University Drive, Burnaby, BC V5A 1S6, Canada
cedric.chauve@sfu.ca

Abstract. The PQ-tree is a fundamental data structure that can encode large sets of permutations. It has recently been used in comparative genomics to model ancestral genomes with some uncertainty: given a phylogeny for some species, extant genomes are represented by permutations on the leaves of the tree, and each internal node in the phylogenetic tree represents an extinct ancestral genome, represented by a PQ-tree. An open problem related to this approach is then to quantify the evolution between genomes represented by PQ-trees. In this paper we present results for two 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 permutations are given, with $p \geq 1$, and we want to compute a permutation generated by the PQ-tree that minimizes the sum of the breakpoint distances to the p permutations. We show that this problem is Fixed-Parameter Tractable with respect to the breakpoint distance value. This last result applies both on signed and unsigned permutations, and to uni-chromosomal and multi-chromosomal permutations.

1 Introduction

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 [4], it has numerous applications: for example, recognizing interval graphs, testing whether a graph is planar, and creating a contig map from DNA segments [4,1,14]. 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 on Σ (no two leaves can have the same label). A fundamental feature of PQ-trees is that a given PQ-tree encode in linear space a possibly exponential number of permutations.

Recently, 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 [6] and references there). The 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. Note that some other ways to account for uncertainty or contradictory information have been defined, such as partial orders [18], but not in the context of ancestral genomes.

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. Following previous works on comparing structures generating several permutations, we consider the Minimum-Breakpoint-Permutation from PQ-Trees (MBP-PQ): 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 [10,3], this problem is NP-complete. Next, we consider the restricted problem where T_2 generates a single permutation, that we call the One-Sided MBP-PQ, and we show that this problem is Fixed-Parameter Tractable (FPT), with parameter being the optimal breakpoint distance. 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). This problem generalizes naturally the classical Breakpoint-Median Problem, by imposing constraints on the possible medians, at least for permutations that represent uni-chromosomal genomes. As far as we know, our FPT algorithm is only the second occurrence of an FPT result for hard median problems, after [11].

2 Preliminaries

Permutations, breakpoints and medians. 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, 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 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 set of alphabet Σ, we say 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 appears in s_2, then we say 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 [17], 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 frame them by two new markers, that become telomeres, and the distance formula, that we will use in this case, is $d_b(s_1, s_2) = n - a(s_1, s_2)$, 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.

Given p permutations $\{s_1, \ldots, s_p\}$, the Breakpoint-Median Problem asks for a permutation s that minimizes $\sum_{i=1}^{p} d_b(s_i, s)$.

To handle signed markers in permutations, we use the same idea as in [12]: 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.

PQ-trees. Formally, a PQ-tree for unsigned permutations is a plane tree with internal nodes that can be either P-nodes or Q-nodes (P-nodes and Q-nodes have at least 2 children). (Note that when a P-node has 2 children, it is really a Q-node.) Reading the leaves of a PQ-tree in a post-order traversal gives a permutation called the *signature* of this PQ-trees. 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 Figure 1 for an illustration of PQ-trees and generated uni-chromosomal permutations. When dealing with multi-chromosomal permutations, we assume the root of the considered PQ-tree T is a P-node. The set of multi-chromosomal

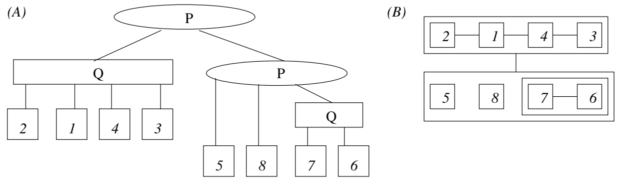

Fig. 1. (A) A PQ-tree T. $\langle 2, 1, 4, 3, 7, 6, 8, 5 \rangle$ and $\langle 3, 4, 1, 2, 5, 6, 7, 8 \rangle$ are permutations generated by this PQ-tree, but not $\langle 1, 2, 3, 4, 5, 6, 7, 8 \rangle$ as 1 has to be adjacent to 4 because they are adjacent siblings in a Q-node. (B) A graph representation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

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.

Problem statements. We now formally state the problems we will investigate in this paper. Each has four different versions, depending on whether the considered permutations are uni-chromosomal or multi-chromosomal, and signed or unsigned.

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 $d_b(s_1, s_2) \leq K$?

The **One-Sided MBP-PQ** is the special case where T_2 generates 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: PQ-trees 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 $\sum_{i=1}^{p} d_b(s, s_i) \leq K$?

FPT algorithms. An FPT (Fixed-Parameter Tractable) algorithm for an optimization problem Π with parameter value p is an algorithm which solves the problem in $O(f(p)n^c)$ time, where f is any function 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 [8].

Existing 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, that is NP-hard, for signed or unsigned, uni-chromosomal or multi-chromosomal permutations [5,16,17]. In the uni-chromosomal case, even in the case where the median is constrained to have only adjacencies that appear in at least one of the genomes s_i, the problem is NP-hard [5]. This implies immediately that the p-MBM-PQ Problem is NP-hard, for $p \geq 3$, in all cases.

The MBP-PQ Problem, which we prove to be NP-complete in next section for unsigned permutations, can be solved by an FPT algorithm whose parameter is $t = P(T_1) \times P(T_2)$, as 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 superexponential 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.

The same argument applies to the p-MBM-PQ Problem, and, even in the case where T has only Q-nodes (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, we consider in Section 5 some real data where $P(T)$ is too large for this approach, which motivates our investigation of an FPT with respect to an alternative parameter. In Section 4, 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 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.

MBP-PQ is obviously in NP and we now show that X3C can be reduced to MBP-PQ in polynomial time.

We first outline the difficulty in the proof and how to handle them one by one. 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 somehow balance the use of P-nodes with Q-nodes. (1) In a solution for X3C, each element belongs to exactly one selected 3-set. 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 must construct a subtree in T_2 such that the number of possible adjacencies (non-breaking point) 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 adjacencies in T_1 and T_2; moreover, these matches imply a solution for X3C. Next we present the details.

We first construct T_1 as follows. The root of T_1, $r(T_1)$, is a Q-node. Each of the children F_i of the root corresponds to an element v_i in V and is of 4 levels (with some leaves possibly compressed in level-3, see Figure 2 (A)), 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 T_f'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 contains 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

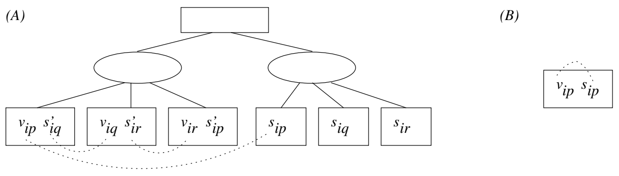

Fig. 2. The subtree F_i. In (A) and (B) the dotted arcs indicate the corresponding adjacencies. (A) shows the construction that v_i appears three times in S. (B) shows the case when v_i appears only once in S.

with t leaves: $s_{i,p_1}, s_{i,p_2}, ..., s_{i,p_t}$. 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 Figure 2 (A), note that $t = 3$.

When v_i appears in S exactly once (say, in S_p), F_i would be a Q-node with two leaves: $v_{i,p}, s_{i,p}$ (Figure 2 (B)). We would have to use some peg markers to compose new leaf nodes to bound $s'_{i,p}$ 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 $1 \leq w \leq t$.*

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 a 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 Figure 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 each can 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 $v_{j,p}s'_{j,p}$.

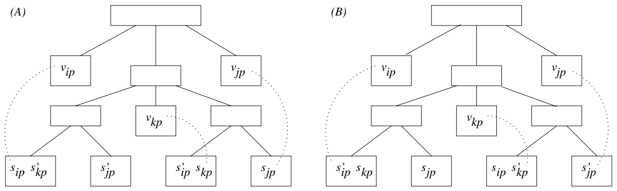

Fig. 3. The subtree H_p corresponding to $S_p = \{v_i, v_j, v_k\}$. (A) and (B) show the two different kinds of adjacencies (marked by dotted arcs).

Following Lemma 1, T_1 can generate exactly n adjacencies in the form of $v_{i,p_w}s_{i,p_w}$ for some $1 \leq w \leq t$. 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 $3m$ 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 an adjacency. Following Lemma 3, we can obtain $3m$ adjacencies in T_1 and T_2.

We now prove the "if" part. Assume that T_1 and T_2 generate exactly $3m$ adjacencies, we first show that there must be n adjacencies in the form of $v_{i,p}s_{i,p}$. If it is not the case, say in T_2 some $v_{i,p}$ is never forming an adjacency with $s_{i,p}$, then the 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, T_2 can generate $3m$ adjacencies.

Now assume that among the $3m$ adjacencies in T_1, T_2 there are n adjacencies in the form of $v_{i,p}s_{i,p}$, we argue that they exactly present a corresponding solution for X3C. By the way we construct T_1, if $v_{i,p}$ forms an adjacency with $s_{i,p}$ then it 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 1. *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 so we can order them in increasing and decreasing order respectively 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.

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 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, T_2)

has breakpoint distance K if and only if (T_1', T_2') has breakpoint distance $K+8$ because we add 8 markers that do not form any adjacency, neither common telomere.

Corollary 1. *MBP-PQ is NP-complete for multi-chromosomal unsigned permutations.*

It is open whether one can design efficient FPT and/or approximation algorithms for the optimization version of MBP-PQ.

4 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.

A graphical representation of PQ-trees. We first introduce a graph-like representation of a PQ-tree, that encodes the adjacency constraints between markers, and was used in [6] to represent ancestral genomes in a linear-like way. The graph G associated to a PQ-tree T has vertices for all nodes (internal and leaves) of T except the root, if it is a P-node. We call the vertices that correspond to leaves *markers*. And the vertices corresponding to P-nodes (resp. Q-nodes) are called *super* P-nodes (resp. Q-nodes). Edges of G are defined only between pairs of markers (or, of course, two super-nodes which must be adjacent): two markers x and y define an edge (x, y) if and only if they are consecutive children of a Q-node. Edges of G are called *black edges*. See Figure 1.

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 Figure 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_1 of a PQ-tree T_1 using another permutation s_2. It turns out that this will be the basis for us to handle the ancestral genome analysis when a phylogeny is given. We start with G_1, and then add an edge, called a *blue* edge, (x, y) in G_1 for every adjacency xy in s_2. We denote this new graph G_1' (note that G_1' conserves the embedding structure we defined on G_1: only blue edges are added). The *degree* of a super-node X in G_1' is the number of edges that connects a marker inside X to a marker outside X. See Figure 1 and Figure 4.

At this point, it is easy to see that the One-Sided MBP-PQ Problem is closely related to the classical Minimum Path Cover Problem.

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.

Lemma 5. *An optimal solution for One-Sided MBP-PQ can be obtained by performing the following operations on G'_1.*

1. *If a marker x is in the middle of a Q-node Y which contains x, 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 could be obtained by allowing at most two blue edges connecting to x.*
3. *If a super-node X is of degree greater than two, then an optimal solution could be obtained by allowing at most two blue edges connecting to some markers inside X.*

Let r be the maximum degree of a super node, after all edge deletion operations at Step 1 of Lemma 5 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 [8] that considers super nodes of degree at least three and, for such a node X, conserves 2 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, 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

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

This recurrence achieves its maximum value when $r = 3$. Therefore,

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

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 less than 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 delete the 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'_1 is computed.

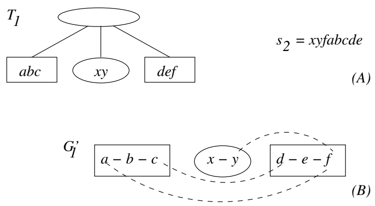

Fig. 4. An example for the FPT algorithm for One-Sided MBP-PQ

In Figure 4, we show a simple example for the algorithm. An example of T_1 and s_2 is illustrated in Figure 4 (A). The augmented graph G_1' is shown in Figure 4 (B). The optimal solution value is $K = 1$. According to the algorithm, we will have to delete one blue (or dashed) edge in G_1'. 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 $s_1 = abcdefyx$ and exactly one breakpoint between s_1 and $s_2 = xyfabcde$. Deleting (y, f) or (c, d) alone both leads to infeasible solutions.

Theorem 2. *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.*

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 with a total weights of $K' \leq K$ from G'' such that the resulting graph is composed of paths. If there are K'' such paths, then 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 then $K' + p(K'' - 1)$. This leads to the following result.

Corollary 2. *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 5.(1) increases the breakpoint distance by 1 but is not considered in the computation. More formally, if d is the number of edges discarded due to Lemma 5.(1), the running time is in fact $O(3^{K-d} n)$. This has been confirmed in our initial computational results. 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 [5]. Indeed, it suffices to forbid deleting blue edges that disconnects the augmented graph, which is obviously connected at first.

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

5 Application to Real Datasets

We present here preliminary computational results on some mammalian and yeast genomes to illustrate the ability of our FPT algorithm to handle real datasets, using a regular Lenovo laptop and C++. Precise data and results are available at the URL http://www.cs.montana.edu/bhz/PQ-TREE.html. In both cases, we change a multi-choromosomal genome into a uni-chromosomal signed permutation; as a consequence, we do not compute exactly the breakpoint distance as defined in [17], as we might create conserved adjacencies and we ignore common telomeres in the computation of the distance. But these results are presented to illustrate the ability of our algorithm to handle datasets with PQ-trees generating a large number of permutations. The running times are still high (varying from two days to about a week), but they are already better than what the theoretical results imply (for the three cases, we have $K = 69, 108,$ and 348).

The mammalian dataset we use is from the following simple phylogenetic tree of five species, (((Human,Macaca)I,(Mouse,Rat)II)III,Dog), given in Newick format, and we are interested in the ancestors of Human and Macaca (node I) and Mouse and Rat (node II). Permutations and PQ-trees at nodes I and II were generated using methods as described as in [15]. In this case, $n = 689$. In the companion webpage, we show in detail the dataset and the sequences generated using the FPT algorithm for 2-MBM-PQ, for node I and node II. For node I, we found that the optimal breakpoint distance is 69, and for node II, the optimal distance is larger, at 108. Notice that these solutions are not unique (in fact in both cases there are about 10! permutations which minimizes $d_b(s, s_1) + d_b(s, s_2)$, due to that the roots of the trees are both P-nodes). So an exhaustive search would not work to generate an optimal permutation for node III.

The yeast data is from [13], the PQ-tree has a root which is a Q-node with 34 children (which are all Q-nodes or leaves). Among these 34 children, 8 of them are leaves. We found an optimal distance of 348. If we wanted to enumerate all generated permutations, we would have to try 2^{26} different 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 problem is NP-complete (not a surprise!), we show that several interesting cases, that are relevant from an applied point of view, are in FPT, parameterized by the optimal breakpoint distance. We also present some preliminary computational results.

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 (hyper?) graph, 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 [4,2].

How to improve the efficiency of the FPT algorithms for One-Sided MBP-PQ and p-MBM-PQ also makes interesting questions. The only other FPT algorithm for a breakpoint median problem, described in [11], 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 is recently proved in [17] that the Breakpoint Median Problem for signed multi-chromosomal genomes is tractable if the median is allowed to have circular chromosomes; it can indeed be solved by a simple maximum weight matching algorithm. In the case of the p-MBM-PQ, the corresponding problem would allow that, in the median, the leaves of one or more subtree rooted at children of the root form a circular chromosome. The complexity of this problem is open.

Finally, what if we consider the problems under other distances such as the DCJ (Double-Cut-and-Join) distance? Intuitively, we can 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 [9] between two genomes represented by the gene content of their chromosomes and with no gene order information, which is NP-hard [7].

Acknowledgments

This research is partially supported by NSF grant DMS-0918034, by NSF of China under grant 60928006, and by NSERC Discovery Grant 249834-2006.

References

1. Alizadeh, F., Karp, R., Weisser, D., Zweig, G.: Physical mapping of chromosomes using unique probes. J. Comp. Biol. 2, 159–184 (1995)
2. Bergeron, A., Blanchette, M., Chateau, A., Chauve, C.: Reconstructing ancestral gene orders using conserved intervals. In: Jonassen, I., Kim, J. (eds.) WABI 2004. LNCS (LNBI), vol. 3240, pp. 14–25. Springer, Heidelberg (2004)

3. Blin, G., Blais, E., Guillon, P., Blanchette, M., ElMabrouk, N.: Inferring Gene Orders from Gene Maps Using the Breakpoint Distance. In: Bourque, G., El-Mabrouk, N. (eds.) RECOMB-CG 2006. LNCS (LNBI), vol. 4205, pp. 99–102. Springer, Heidelberg (2006)
4. Booth, K., Lueker, G.: Testing for the consecutive ones property, interval graphs, and graph planarity using PQ-tree algorithms. J. Computer and System Sciences 13, 335–379 (1976)
5. Bryant, D.: The complexity of the breakpoint median problem. Technical Report CRM-2579. Centre de Recherches en Mathématiques, Université de Montréal (1998)
6. Chauve, C., Tannier, E.: A methodological framework for the reconstruction of contiguous regions of ancestral genomes and its application to mammalian genome. PLoS Comput. 4:e1000234 (2008)
7. DasGupta, B., Jiang, T., Kannan, S., Li, M., Sweedyk, E.: On the Complexity and Approximation of Syntenic Distance. Discrete Appl. Math. 88(1–3), 59–82 (1998)
8. Downey, R., Fellows, M.: Parameterized Complexity. Springer, Heidelberg (1999)
9. Feretti, V., Nadeau, J.H., Sankoff, D.: Original synteny. In: Hirschberg, D.S., Meyers, G. (eds.) CPM 1996. LNCS, vol. 1075, pp. 159–167. Springer, Heidelberg (1996)
10. Fu, Z., Jiang, T.: Computing the breaking distance between partially ordered genomes. In: APBC 2007, pp. 237–246 (2007)
11. Gramm, J., Niedermeier, R.: Breakpoint medians and breakpoint phylogenies: A fixed-parameter approach. Bioinformatics 18(Suppl. 2), S128–S139 (2002)
12. Hannenhalli, S., Pevzner, P.: Transforming cabbage into turnip: polynomial algorithm for sorting signed permutations by reversals. J. ACM 46(1), 1–27 (1999)
13. Jean, G., Sherman, D.M., Nikolski, M.: Mining the semantic of genome super-blocks to infer ancestral architectures. J. Comp. Biol. 16(9), 1267–1284 (2009)
14. Landau, G., Parida, L., Weimann, O.: Gene proximity analysis across whole genomes via PQ-trees. J. Comp. Biol. 12, 1289–1306 (2005)
15. Ouangraoua, A., McPherson, A., Tannier, E., Chauve, C.: Insight into the structural evolution of amniote genomes. In: Preliminary version in Cold Spring Harbor Laboratory Genome Informatics Meeting 2009, poster 137 (2009)
16. Pe'er, I., Shamir, R.: The median problems for breakpoints are NP-complete. Elec. Colloq. Comput. Complexity, TR-98-071 (1998)
17. Tannier, E., Zheng, C., Sankoff, D.: Multichromosomal median and halving problems under different genomic distances. BMC Bioinformatics 10, 120 (2009)
18. Zheng, C., Lennert, A., Sankoff, D.: Reversal distance for partially ordered genomes. Bioinformatics 21(Suppl. 1), i502–i508 (2005)