Tractability Results for the Consecutive-Ones Property with Multiplicity*

Cedric Chauve[1], Ján Maňuch[1,2], Murray Patterson[2], and Roland Wittler[1,3]

[1] Department of Mathematics, Simon Fraser University, Burnaby, BC, Canada
[2] Department of Computer Science, UBC, Vancouver, BC, Canada
[3] Technische Fakultät, Universität Bielefeld, Bielefeld, Germany
cchauve@sfu.ca,{jmanuch,murrayp}@cs.ubc.ca,roland@cebitec.uni-bielefeld.de

Abstract. A binary matrix has the Consecutive-Ones Property (C1P) if its columns can be ordered in such a way that all 1's in each row are consecutive. We consider here a variant of the C1P where columns can appear multiple times in the ordering. Although the general problem of deciding the C1P with multiplicity is NP-complete, we present here a case of interest in comparative genomics that is tractable.

1 Introduction

A binary matrix M has the *Consecutive-Ones Property* (C1P) if there exists a permutation of its columns such that all 1's in each row are consecutive. Deciding if a matrix has the C1P can be done in linear-time and space [3, 5, 11, 9, 10]. This problem has been considered in genomics, for problems such as physical mapping [2, 7] or ancestral genome reconstruction [1, 4, 8].

Recently, Wittler and Stoye in [12], motivated by handling duplicated genes in reconstructing ancestral gene clusters, introduced a generalized problem: Given several sets of genes and a maximum multiplicity for each gene, decide whether there exists a sequence of genes which meets the multiplicity constraint for each gene and in which each set of genes occurs consecutively. This can be phrased in terms of a binary matrix, where a column corresponds to a gene and a set of genes is represented by a row containing a 1 for each gene in the set in the respective column and 0's in all other columns. Now each column c of the matrix is given a multiplicity threshold $\boldsymbol{m}(c)$: M satisfies the *mC1P* (for C1P with multiplicity) if there is a sequence S of columns of M, in which at most $\boldsymbol{m}(c)$ occurrences of column c can appear, and for each row r of M, the columns containing 1 in r appear consecutively somewhere in S. The sequence S corresponds then to a valid gene order. Deciding if a binary matrix M with multiplicity satisfies the mC1P is tractable if every row of M contains at most two entries 1 (which corresponds in gene clusters models to gene adjacencies) [12], but the problem is NP-complete if M contains rows with at most three entries 1 [13]. The mC1P can also be related to gene proximity analysis with duplicated genes [6].

In this work, we present a tractability result for a restricted mC1P decision problem. After some technical preliminaries (Section 2), we give in Section 3 a tractability result for a family of matrices where every row of M has (i) at most one entry 1 in columns with multiplicity greater than one, or (ii) exactly two entries 1 in columns with multiplicity greater than one and no other entries. This result is motivated by handling telomeres in ancestral gene order reconstruction (described in Appendix A). Our proofs rely on two classical concepts: PQ-trees and Eulerian cycles in graphs. We conclude by discussing future work.

2 Preliminaries

Let M be a binary matrix, with m rows $\mathcal{R} = \{r_1, \ldots, r_m\}$, n columns $\mathcal{C} = \{c_1, \ldots, c_n\}$ and ℓ entries 1. We represent a row r of M as a subset of $\mathcal{C}$, defined as the set of c_i such that $M[r, c_i] = 1$. A *multiplicity vector*

* To appear in the proceedings of CPM 2011, the 22nd Annual Symposium on Combinatorial Pattern Matching. Version of March 25, 2011.

$\boldsymbol{m}$ for M is a sequence of positive integers $[\boldsymbol{m}(c_1), \ldots, \boldsymbol{m}(c_n)]$: $\boldsymbol{m}(c_i)$ is called the multiplicity of column c_i. A column c with multiplicity $\boldsymbol{m}(c) > 1$ is called a *multicolumn* and a row r containing a multicolumn (i.e., $M[r, c] = \mathtt{1}$ for some column c with $\boldsymbol{m}(c) > 1$) is called a *multirow*. A multirow that does not contain any other multirow is called *minimal*. We say a binary matrix M with multiplicity vector $\boldsymbol{m}$ has *matched multirows* if, for every multirow $r \subseteq \mathcal{C}$ that contains at least two entries 1 in non-multicolumns, there exists a row $\hat{r}$ which is a copy of r where all entries in multicolumns have been discarded (i.e., switched from 1 to 0). We denote by $\hat{M}$ the binary matrix obtained from M by discarding all multicolumns. In this work, we assume that all matrices we deal with have matched multirows unless otherwise stated. Figure 1 illustrates the above definitions.

M	1	2	3	4	5	a	b
r_1	1	1	0	0	0	1	1
$\hat{r}_1$	1	1	0	0	0	0	0
r_2	1	1	1	0	0	0	0
r_3	0	0	1	1	1	0	1
$\hat{r}_3$	0	0	1	1	1	0	0
r_4	0	0	0	1	1	0	1
$\hat{r}_4$	0	0	0	1	1	0	0
r_5	1	0	0	1	1	0	0

$\hat{M}$	1	2	3	4	5
r_1	1	1	0	0	0
r_2	1	1	1	0	0
r_3	0	0	1	1	1
r_4	0	0	0	1	1
r_5	1	0	0	1	1

Fig. 1. Left: Binary matrix M, with matched multirows. Let $\boldsymbol{m}(1) = \cdots = \boldsymbol{m}(5) = 1$ and $\boldsymbol{m}(a) = \boldsymbol{m}(b) = 2$: a and b are multicolumns and r_1, r_3 and r_4 are multirows. Row r_3 is not minimal, because it contains r_4. Right: The corresponding matrix $\hat{M}$. Since in $\hat{M}$, by definition $\hat{r}_i = r_i$ for all multirows r_i, the matched multirows are discarded.

Definition 1. *Matrix M has the* Consecutive-Ones Property with multiplicity *(mC1P) for multiplicity vector $\boldsymbol{m}$ if there exists a sequence $S = s_1 \ldots s_p$ on the alphabet $\mathcal{C}$ such that it meets*

(1) the consecutivity requirement*: for each row r of M there are two integers j, k, with $j < k$ such that $r = \{s_j, s_{j+1}, \ldots, s_k\}$ (the columns in r are consecutive in S), and*
(2) the multiplicity requirement*: each c_i appears at most $\boldsymbol{m}(c_i)$ times.*

The sequence S is then called an mC1P-ordering *of M.*

Given row $\{1, 2, 3, 4\}$, an example of a sequence that satisfies condition (1) of the above definition is the sequence 5142435, since $\{1, 2, 3, 4\} = \{1, 4, 2, 4, 3\}$.

The mC1P generalizes the classical Consecutive-Ones Property (C1P), where $\boldsymbol{m}(c_i) = 1$ for every c_i. Lemma 1 below, whose proof is straightforward, relates both problems.

Lemma 1. *Every mC1P-ordering of M with multiplicity vector $\boldsymbol{m}$ contains a C1P-ordering of $\hat{M}$ as a subsequence. As a consequence, if a binary matrix M has the mC1P, then $\hat{M}$ has the C1P.*

This lemma suggests that, to decide if M has the mC1P for a given multiplicity vector $\boldsymbol{m}$, we can first check if $\hat{M}$ has the C1P, and then extend a C1P-ordering of $\hat{M}$ into an mC1P-ordering of M by adding copies of multicolumns. Note that the matrix $\hat{M}$ in Figure 1 does not have C1P, and hence, M does not have mC1P. However, if we omit column r_5, then 12345 is a C1P ordering of $\hat{M}$, which can be extended to the following mC1P-ordering of M: $ab12345b$. To account for the fact that there can be an exponential number of C1P-orderings of $\hat{M}$, we use PQ-trees, a linear size structure that can describe all C1P-orderings of $\hat{M}$, defined below. For a more complete treatment of PQ-trees, we refer the reader to [3, 9].

Definition 2. *A PQ-tree on $\mathcal{C}$ is a rooted ordered tree with leaves labeled by $\mathcal{C}$ and two kinds of internal nodes,* P-nodes *and* Q-nodes*. Each P-node has at least two children and each Q-node has at least three children.*

The frontier *$F(T)$ of a PQ-tree T is the sequence of $\mathcal{C}$ obtained by reading the labels of its leaves from left to right. The* frontier *of a node N in T is the frontier of the subtree rooted at N. Let $\{F(N)\}$ be the set of elements appearing in the sequence $F(N)$.*

Two PQ-trees are equivalent if one can be obtained from the other by applying a sequence of the following transformation rules: (RP) arbitrarily permute the children of a P-node; (RQ) reverse the order of the children of a Q-node.

Theorem 1. [3] *If a binary matrix M has the C1P, there exists a unique equivalence class PQ_M of PQ-trees with the property that there is a one-to-one correspondence between the C1P-orderings of M and the frontiers of the PQ-trees of PQ_M, and a PQ-tree belonging to PQ_M can be constructed in linear time.*

Each PQ-tree in the equivalence class PQ_M satisfies the following properties (that are implicitly given in [3, 9]) which we will use in this paper.

Property 1. Let M be a binary matrix that has C1P with rows $\mathcal{R}$ and T a PQ-tree in the equivalence class PQ_M. Then

1. for every row $r \in \mathcal{R}$, there is a node N in T such that either $\{F(N)\} = r$, if N is a P-node, or r is consecutive in $F(N)$, if N is a Q-node;
2. for every node N different from the root of T, there is a row $r \in \mathcal{R}$ such that $\{F(N)\} \subseteq r$; and
3. for every Q-node N, and every two consecutive children N_1 and N_2 of N, there is a row $r \in \mathcal{R}$ such that $\{F(N_1)\} \cup \{F(N_2)\} \subseteq r$.

Finally, we recall briefly the technique used to prove that matrices with two entries `1` per row (usually called matrices of *degree* 2) form a class of tractable instances for deciding the mC1P as we will use it to prove our main result. Such matrices can be naturally represented as a collection of adjacency constraints $\mathcal{A} = \{\{a_i, b_i\}\}_{i=1}^{m}$ on the set $\mathcal{C}$, where $a_i \neq b_i$ and the collection is a set (no duplicate elements). Collection $\mathcal{A}$ is *consistent* with respect to $\boldsymbol{m}$ if there is a sequence S on $\mathcal{C}$ such that each adjacency is consecutive in S. We will refer to this sequence as a *consistency sequence* of $\mathcal{A}$ and $\boldsymbol{m}$. Note that an mC1P-ordering of M is a consistency sequence of the corresponding collection $\mathcal{A}$ and $\boldsymbol{m}$, and vice versa, and hence, M has the mC1P for $\boldsymbol{m}$ if and only if $\mathcal{A}$ is consistent with respect to $\boldsymbol{m}$. Given a collection of adjacencies $\mathcal{A}$, we define the graph $G_{\mathcal{A}}$ with vertex set $\mathcal{C}$ and edges given by adjacencies.

Theorem 2. [12] *A collection of adjacencies $\mathcal{A}$ is consistent with respect to a multiplicity vector $\boldsymbol{m}$ if and only if for all $c_i \in \mathcal{C}$, $\text{degree}_{G_{\mathcal{A}}}(c_i) \leq 2\boldsymbol{m}(c_i)$ and for each connected component $B \subseteq \mathcal{C}$ of $G_{\mathcal{A}}$, for at least one $c_i \in B$, $\text{degree}_{G_{\mathcal{A}}}(c_i) < 2\boldsymbol{m}(c_i)$.*

The above theorem relies on the fact that the graph $G_{\mathcal{A}}$ satisfying the above conditions can be extended to a multigraph on $\mathcal{C} \cup \{c_0\}$ that has an Eulerian cycle. It can be easily seen that the proof presented in [12] applies to generalized adjacencies, where we allow $a_i = b_i$ and the collection to be a multiset, and we require that each adjacency in $\mathcal{A}$ appears in S in a unique position. Note that $G_{\mathcal{A}}$ is now a multigraph with self-loops. We have the following corollary.

Corollary 1. *A collection of generalized adjacencies $\mathcal{A}$ is consistent with respect to a multiplicity vector $\boldsymbol{m}$ if and only if for all $c_i \in \mathcal{C}$, $\text{degree}_{G_{\mathcal{A}}}(c_i) \leq 2\boldsymbol{m}(c_i)$ and for each connected component $B \subseteq \mathcal{C}$ of $G_{\mathcal{A}}$, for at least one $c_i \in B$, $\text{degree}_{G_{\mathcal{A}}}(c_i) < 2\boldsymbol{m}(c_i)$.*

3 A Tractable Case of the mC1P Decision Problem

Our main result is that deciding the mC1P is tractable for a large family of matrices with constraints on the maximum number of entries `1` in multicolumns a row can have. The motivation for studying this particular family of matrices arises from incorporating information on telomeres in ancestral gene order reconstruction (Appendix A).

Theorem 3. *Let M be a binary matrix and $\boldsymbol{m}$ a multiplicity vector such that (1) M has matched multirows, and (2) each row contains either (i) at most one entry `1` in multicolumns, or (ii) two entries `1` in multicolumns and no other entries. Deciding if M has the mC1P for $\boldsymbol{m}$ can be done in polynomial time and space.*

We split the proof into two parts. In Section 3.1, we consider the case (2i) where M with multiplicity vector $\boldsymbol{m}$ contains a single multicolumn, and we show that deciding if M has the mC1P for $\boldsymbol{m}$ can be done efficiently using PQ-trees. Then, in Section 3.2, we show how to handle the general case using Corollary 1 which relies on Eulerian cycles. Finally, in Section 3.3, we give an algorithm for building a PQ-tree which describes all sequences that satisfy the consecutivity requirement (condition (1) of Definition 1).

3.1 The Case of a Single Multicolumn

We assume that the multiplicity vector $\boldsymbol{m}$ defines only one multicolumn denoted by c'. According to Lemma 1, M satisfies the mC1P only if $\hat{M}$ has the C1P, which can be checked in linear time (Theorem 1). Assume that $\hat{M}$ has the C1P and let T be a PQ-tree from the equivalence class $PQ_{\hat{M}}$. We then aim at finding a PQ-tree from $PQ_{\hat{M}}$ (by applying operations (RP) and (RQ) on T) whose frontier can be extended to a valid mC1P-ordering by inserting copies of c'. We say that inserting a copy of c' into $F(T)$ *breaks* a row r of $\hat{M}$ if r is not consecutive in the resulting sequence. An example is given in Figure 2.

	1	2	3	4	5	6	7	8	9	c'
r_1	1	1	0	0	0	0	0	0	0	1
$\hat{r}_1$	1	1	0	0	0	0	0	0	0	0
r_2	1	1	1	0	0	0	0	0	0	0
r_3	0	0	1	1	0	0	0	0	0	1
$\hat{r}_3$	0	0	1	1	0	0	0	0	0	0
r_4	0	0	0	0	0	0	1	1	0	1
$\hat{r}_4$	0	0	0	0	0	0	1	1	0	0
r_5	0	0	0	0	0	0	0	1	1	0
r_6	0	0	0	0	1	1	0	0	0	0

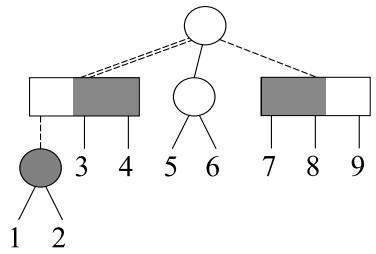

Fig. 2. Left: Binary matrix M, with matched multirows. Let $\boldsymbol{m}(c') = 2$. Right: PQ-tree belonging to the equivalence class $PQ_{\hat{M}}$. P-nodes are represented by circular nodes and Q-nodes by rectangular nodes. An example of a valid mC1P-ordering is $c'\,1\,2\,3\,4\,c'\,7\,8\,9\,5\,6$ which is obtained by taking the equivalent PQ-tree with frontier $1\,2\,3\,4\,7\,8\,9\,5\,6$ and inserting two copies of c' into the corresponding positions. Notice that inserting c' between 2 and 3 would break row r_2.
Illustration of Algorithm 1. LCA($\hat{r}_1$) and the respective segments of LCA($\hat{r}_{3,4}$) are highlighted in gray and the respective paths are depicted by dashed lines. The upper left edge is contained in two paths. Here, $K_1 = 1$ and $K_2 = 1$, thus $K = 2 \leq \boldsymbol{m}(c') = 2$.

Recall that rows are subsets of $\mathcal{C}$. As M has matched multirows, all rows in $\hat{M}$ are also rows in M. Since the consecutivity of the 1's in each row of $\hat{M}$ in the frontier $F(T)$ has to be maintained when inserting copies of c', no c' can be inserted into a position where it breaks any row of $\hat{M}$. Lemma 2 below is a consequence of this observation.

Lemma 2. *Let M be a binary matrix with matched multirows, and $\boldsymbol{m}$ be a multiplicity vector defining exactly one multicolumn c'. Assume that M has the mC1P, and let T be a PQ-tree from $PQ_{\hat{M}}$ and T' an extension of T whose frontier $F(T')$ is an mC1P-ordering of M.*

1. *If the root of T is a P-node, then, for each child node N of the root, c' can only appear as the first or last element of the frontier $F(N)$ in T'.*
2. *If the root of T is a Q-node, the copies of c' in T' can only appear as the first and/or last element of the frontier $F(T')$.*

Proof. It follows by Property 1.2 that for every child N of the root of T, any pair of consecutive leaves in $F(N)$ belongs to a row of $\hat{M}$, and hence, inserting c' between these leaves breaks this row.

In addition, if the root of T is a Q-node, then by Property 1.3, for any two consecutive children N_1 and N_2 of the root, there is a row of $\hat{M}$ that contains elements of $F(N_1)$ and of $F(N_2)$. This prevents the insertion of c' into root between N_1 and N_2 as this would break such a row. Hence c' can appear only at the extremities of $F(T')$. □

Lemma 2 rules out many positions in $F(T)$ where to insert copies of c': indeed, copies of c' can only be inserted at extremities of the subsequences of $F(T)$ formed by children of the root (and only at the extremities of $F(T)$, if the root is a Q-node). On the other hand, each multirow specifies a position where a copy of c' must be inserted. These two constraints give rise to a polynomial algorithm which we describe in the following.

Algorithm 1 starts with a PQ-tree for $\hat{M}$ and works in two stages. First (Step 3), based on Lemma 2, it checks if there is a way to permute nodes in the subtrees rooted at each child of the root such that for each multirow $r = \hat{r} \cup \{c'\}$, rows in $\hat{r}$ appear as a prefix or a suffix of the frontier of some child. To satisfy the consecutivity requirement for each multirow r it is enough to add copies of c' to $F(T)$ before or after the frontier of the child of the root containing $\hat{r}$. To satisfy the multiplicity requirement, we need to permute the children of the root and possibly reverse the order of the frontier of some children. The basic idea is that we can save one copy of c' if a child requiring a copy of c' on the right is followed by a child requiring a copy of c' on the left. Whether enough copies of c' can be saved to satisfy the multiplicity requirement is checked in Steps 4–5.

Let $r = \hat{r} \cup \{c'\}$ be a multirow. By Property 1.1, there is in T either a P-node that contains exactly the columns in $\hat{r}$ in its subtree, or a Q-node with a segment of two or more consecutive children which together contain exactly the columns in $\hat{r}$ in their subtrees. This node is the least common ancestor in T of the columns in $\hat{r}$, and hence, will be denoted by $\mathrm{LCA}(\hat{r})$.

Algorithm 1 Deciding the mC1P for a matrix M with matched multirows and a multiplicity vector $\boldsymbol{m}$ defining a single multicolumn c'.

1. Check if $\hat{M}$ has the C1P.
2. If not, return false, else let T be a PQ-tree from $PQ_{\hat{M}}$.
3. For each minimal multirow $r = \hat{r} \cup \{c'\}$ in M do
   a. Locate $N := \mathrm{LCA}(\hat{\mathrm{r}})$.
   b. Let P_r be the path from N to the root of T.
   c. For each edge $e = \{U, V\}$ in P_r, where U is the parent of V do
      i. If U is a Q-node and V is neither its first nor its last child, return false;
      ii. If the root of T is a Q-node and e also belongs to the path $P_{r'}$ defined by another minimal multirow r', return false;
      iii. If U is not the root of T and e also belongs to the path defined by another minimal multirow, return false;
      iv. If U is the root of T and e also belongs to the paths defined by at least two other minimal multirows, return false.
4. If the root of T is a Q-node, return true.
5. If the root of T is a P-node:
   a. Let K_1 and K_2 be the number of children of the root of T belonging to exactly one or two paths defined by minimal multirows, respectively.
   b. $K := \left\lceil \frac{K_1}{2} \right\rceil + K_2 + \begin{cases} 1 \text{ if } K_1 = 0 \text{ and } K_2 > 0, \\ 0 \text{ otherwise.} \end{cases}$
   c. Return $K \leq \boldsymbol{m}(c')$

Now to argue that Algorithm 1 is correct. If condition 3.c.i applies, r would require the insertion of a copy of c' within $F(U)$ in any PQ-tree of $PQ_{\hat{M}}$, which contradicts Lemma 2.

The paths indicate positions where copies of c' have to be added to the frontier so that the consecutivity requirement is satisfied. Following Lemma 2, we have to verify whether we can transform T such that all paths lie on the outside of the subtree of a child of the root of T. If conditions 3.c.ii–3.c.iv apply, there are

two or more competing multirows, and we cannot transform T such that all of the corresponding paths lie on the outside of the subtree of a child of the root of T. Paths that are sub-paths of one another are excluded by not considering any multirow $r = \hat{r} \cup \{c'\}$ which contains another multirow $r' = \hat{r}' \cup \{c'\}$ (line 3). These rows do not need to be considered at this stage, because in any ordering with c' adjacent to the elements in $\hat{r}'$, since $\hat{r}' \subseteq \hat{r}$, c' is also adjacent to the elements in $\hat{r}$. If the root of T is a P-node, we have to consider the children of the root node separately: We could insert a copy of c' on both sides of a frontier of a child of the root, i.e., at most two paths can join above such a child node. In levels below the root, only one path can be moved to the border of the subtree, i.e., no two edges can join.

If conditions 3.c.i–iv do not apply for a multirow r, there is a way to transform T (with rules (RP) and (RQ)) in the nodes on the path P_r (excluding the root) so that the frontier of $N = \mathrm{LCA}(\hat{r})$ appears as a prefix or suffix of the frontier of N', where N' is a child of the root lying on the path P_r. Next, we will show that all these transformations can be performed simultaneously without any conflict. Obviously, the conflicts could only occur if the paths P_r share vertices other than root. Condition 3.c.iv guarantees that there are never three or more minimal multirows in the same subtree rooted at a child N' of the root. Condition 3.c.iii guarantees that if there are two minimal multirows in the same subtree rooted at a child N' of the root, their paths must meet only in N', and hence, one can appear as a prefix and one as a suffix of the frontier of N'. However, if the root is a Q-node, by Lemma 2, column c' can be attached only on one side of the frontier of N', and hence, only one minimal multirow can appear in the subtree rooted at N', which is checked in condition 3.c.ii.

Hence, if Step 3 succeeds for all rows, there is a PQ-tree in $PQ_{\hat{M}}$ from which we can obtain a sequence of the columns fulfilling the consecutivity requirement of M by inserting copies of c' into its frontier at positions indicated by the paths of multirows. Steps 4–5 check if the multiplicity constraint imposed by $\boldsymbol{m}$ can be satisfied. Note, that if the root of T is a Q-node (Step 4), then the multiplicity constraint is satisfied since $\boldsymbol{m}(c') \geq 2$.

In Step 5, we count the number of copies of c' required to satisfy all multirows. The position where to insert these copies are given by the paths. Since the root of T is a P-node, we can rearrange the children of the root such that one copy of c' would coincide with two paths (from neighboring children). For instance, we can greedily pair nodes with one path each, using $\lceil K_1/2 \rceil$ copies and then include nodes with two paths (one path on each side) in-between, requiring one further copy each, K_2 in total. If $K_1 = 0$ and $K_2 > 0$, chaining the two-path nodes results in $K_2 + 1$ copies of c'. It is easy to see that this joining process is optimal.

If the number of required copies of c' does not exceed the given maximum multiplicity $\boldsymbol{m}(c')$, the given matrix M with multiplicity vector $\boldsymbol{m}$ has the mC1P. Finally, to complete the proof of the correctness of the algorithm, we only need to notice that the result of Algorithm 1 does not depend on the choice of the PQ-tree T of $PQ_{\hat{M}}$, as the LCAs and paths are invariant under the transformation rules (RQ) and (RP).

The analysis of the time and space complexity of Algorithm 1 is as follows. First, Steps 1 and 2 can be completed in $O(m + n + \ell)$ time and space using the algorithm described in [9]; note that T can then be encoded in $O(n)$ space. Next, Step 3 is composed of at most m iterations, each of them requiring time $O(n)$, the maximum length of a path from N to the root of T, as each path is obviously processed in time linear in its length. This gives an $O(mn)$ time complexity for Step 3. For similar reasons, Step 4 can be achieved in time $O(mn)$, which gives an overall worst-case time complexity of $O(mn)$. This completes the proof of the case of a single multicolumn in Theorem 3.

3.2 Completing the Proof of Theorem 3

Proof (Proof of Theorem 3). Given matrix M with multiplicity vector $\boldsymbol{m}$ and having matched multirows, let $\mathcal{C}'$ be its set of multicolumns. A multirow containing multicolumn $c' \in \mathcal{C}'$, will be called a *c'-multirow*. Algorithm 2 works in the same two stages as Algorithm 1. However, the second stage is more complex. It requires building the collection of generalized adjacencies $\mathcal{A}$ on set $\mathcal{C}' \cup \{c_0\}$ by replacing each child of the root of the PQ-tree T for $\hat{M}$ by an adjacency and then applying Corollary 1.

Correctness of Step 1 follows from the correctness of the first stage of Algorithm 1. If Step 1 succeeds, we can assume that the root of T is a P-node (the case when the root is a Q-node is handled in Step 1), and hence, it is enough to satisfy the multiplicity requirement by permuting the children of the root and

Algorithm 2 Deciding the mC1P for a matrix M with matched multirows and a multiplicity vector $\boldsymbol{m}$.

1. Run the first 4 steps of Algorithm 1, where c' is any element of $\mathcal{C}'$.
2. Construct a multiset of generalized adjacencies $\mathcal{A}$ on set $\mathcal{C}' \cup \{c_0\}$ as follows. For every child N of the root of T do
   a. If N belongs to exactly one path defined by multirows, say by a c'-multirow, add adjacency $\{c', c_0\}$ to $\mathcal{A}$;
   b. If N belongs to two paths defined by multirows, say by a c'-multirow and a d'-multirow (c' and d' may be equal), add adjacency $\{c', d'\}$ to $\mathcal{A}$.
3. Report if $\mathcal{A}$ is consistent with respect to $\boldsymbol{m}$ (use Corollary 1).

possibly reversing the order of the frontiers of some children. Let π be this order of children of the root. In Step 2, the algorithm constructs the multiset of generalized adjacencies $\mathcal{A}$ whose consistency sequence (produced in Step 3) describes the way to do this as follows. Children that belong to zero paths defined by multirows will not introduce any adjacency constraints and can be placed at the end of π in any order and orientation. For any other child of the root, we have a unique position in the consistency sequence, hence we can order and orient these children based on these positions. Next, we insert copies of multicolumns as follows. For each subsequence $c_1c_2c_3$ of the consistency sequence, where adjacency $\{c_1, c_2\}$ corresponds to child N_1 and $\{c_2, c_3\}$ to N_2, if $c_2 \neq c_0$, we insert a copy of c_2 between the frontiers of N_1 and N_2 in $F(T)$. Hence, the number of copies of a multicolumn $c' \in \mathcal{C}'$ is equal to the number of its occurrences in the consistency sequence. Therefore, the frontier $F(T)$ with all required copies of multicolumns inserted satisfies the multiplicity requirement given by $\boldsymbol{m}$. It is easy to see that if there is an mC1P ordering of M, then we can extract from it an ordering of the children of the root which gives this consistency sequence.

The analysis of the time complexity is as follows. The first stage of the algorithm is a subroutine of Algorithm 1, and hence, has a time and space complexity of order $O(mn)$. Since the number of children of the root of T that belong to at least one path defined by multirows is at most m, the number of adjacencies in $\mathcal{A}$ is at most m, and hence, building $\mathcal{A}$ takes time $O(m)$. Finally, checking the degree conditions (applying Corollary 1) takes time $O(n)$. Hence, the total time and space complexity of the algorithm is $O(mn)$.

Finally, Algorithm 2 can also be easily extended to the case when the matrix also contains rows of degree 2 containing two multicolumns, as follows. First, we run Steps 1 and 2 where we ignore multirows containing two multicolumns. Then, we add to $\mathcal{A}$ also an adjacency for every such multirow. Finally, we run Step 3 of the algorithm on this new collection $\mathcal{A}$. It is easy to see that the time complexity of this new algorithm is still $O(mn)$. Hence, the theorem holds. □

3.3 Building a PQ-tree which Describes All Sequences that Satisfy the Consecutivity Requirement

Here, we describe how a given PQ-tree $T \in PQ_{\hat{M}}$ can be augmented to a PQ-tree T' which represents the set of all sequences S, up to "pumping" occurrences of multicolumns, that satisfy the consecutivity requirement (condition (1) of Definition 1) in that the frontier of any tree in the equivalence class of T' is such a sequence S. However, not all frontiers meet the multiplicity requirement (condition (2) of Definition 1). For some trees in the equivalence class of T', the respective frontier contains pairs of adjacent occurrences of a multicolumn c', each of which can be replaced by one occurrence of c' without breaking any row of M (violating the consecutivity requirement). This reduces the number of used copies of the multicolumns. Only such shortened frontiers which meet the multiplicity requirement are valid mC1P orderings, and, in fact, the set of such shortented frontiers is exactly the set of valid mC1P orderings of M. Figure 3 shows an example.

To construct an augmented PQ-tree T', we process the original tree T in a bottom-up fashion along the paths P_r defined in Algorithm 1, starting with the LCAs. We replace an LCA by a new Q-node which has a copy of its corresponding multicolum c' as its first child and further children, depending on whether the LCA itself and its parent are P or Q-nodes. These intuitive transformation rules are detailed in Figure 4.

Then, any parent node of a newly obtained Q-node is refined to a new Q-node, moving up the copy of c', as shown in Figure 5.

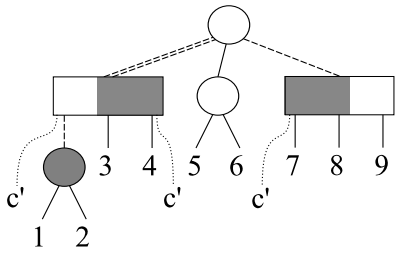

Fig. 3. Augmented PQ-tree T' for the matrix given in Figure 2. (In fact, to get an augmented PQ-tree from the original PQ-tree shown in Figure 2, no modifications are necessary other than attaching leaf nodes labeled c' at appropriate locations.) Only the trees in the equivalence class of T' where the left side of the right Q-node is placed adjacent to the left Q-node have shortened frontiers that meet the multiplicity requirement ($\boldsymbol{m}(c') = 2$), for example, $c'\,1\,2\,3\,4\,c'\,7\,8\,9\,5\,6$.

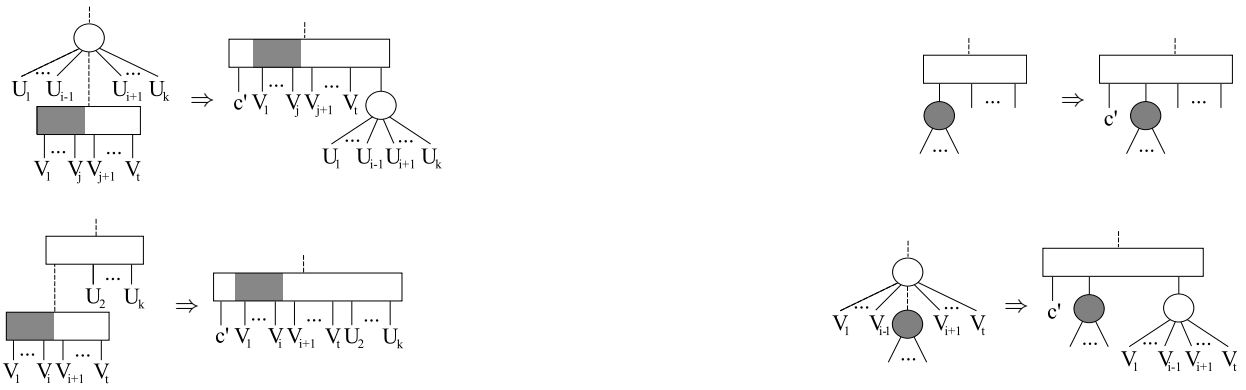

Fig. 4. Transformation rules for the LCAs to construct an augmented PQ-tree. An LCA and its parent node are replaced by the nodes shown on the right. The LCA (or the segment of an LCA, respectively) are highlighted in gray.

Fig. 5. Transformation rules for bottom-up iteration to construct an augmented PQ-tree. A newly created Q-node and its parent node are replaced by the nodes shown on the right.

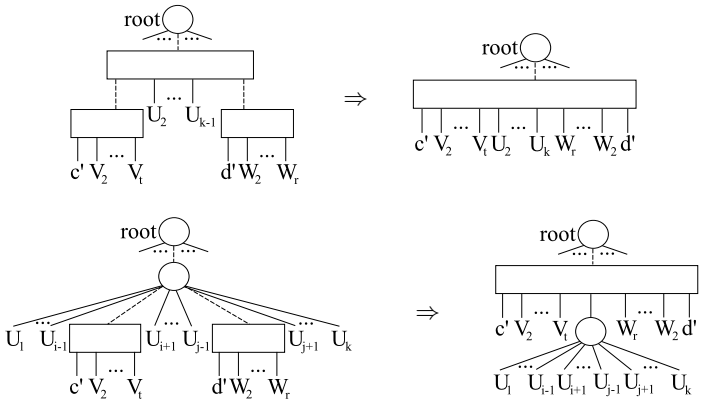

Fig. 6. Special transformation rules for bottom-up iteration to construct an augmented PQ-tree. A newly created Q-node two levels below the root node and its parent node are replaced by the nodes shown on the right.

This process is iterated until we reach the root node. Since a node that is a child of the root can be contained in two paths, separate (but similar) rules are required, illustrated in Figure 6.

Further specific rules which apply if an LCA is a child of the root of T or if the root node is a Q-node are straightforward. In some cases, after generating the tree as described above, simplifications can be carried out, such as replacing a P-node with a single child by a direct edge or substituting a Q-node with two children by a P-node.

Analogously to Algorithm 1, that only checks if a matrix has the mC1P, the above construction of an augmented PQ-tree T' can be carried out in $O(mn)$ time.

4 Conclusion

In the present work, we extend the domain of tractable instances of deciding the C1P with multiplicity for binary matrices. Our approach relies on previously used techniques to decide the C1P and simpler instances of the mC1P, and answers a natural problem in reconstructing ancestral gene orders. Several questions remain open. Naturally, one can ask to relax the condition that M has matched multirows, which is crucial in our proofs. It seems however that the problem becomes hard in this case, and some less rigid constraints on M would then have to be introduced to recover tractability. Also it is open to exhibit an extension of the notion of the PQ-tree that could encode all mC1P-orderings of a binary matrix that satisfies this property. Even for the case of a matrix with matched multirows, our techniques lead to a data structure which only captures the consecutivity requirement but not the multiplicity requirement. From an algorithmic complexity point of view, our algorithm has an $O(mn)$ time complexity, and it remains open to see if this case can be solved in $O(m + n + \ell)$ time.

Acknowledgments. We would like to thank Éric Tannier for suggesting the idea of using the mC1P for dealing with telomeres in ancestral genome reconstruction. Part of this research was funded by an NSERC Discovery Grant to C.C. and J.M.

References

1. Z. Adam, M. Turmel, C. Lemieux, D. Sankoff. Common intervals and symmetric difference in a model-free phylogenomics, with an application to streptophyte evolution. *J. Comput. Biol.* 14, pp. 436–445. 2007.
2. F. Alizadeh, R. Karp, D. Weisser, G. Zweig. Physical mapping of chromosomes using unique probes. *J. Comput. Biol.* 2, pp. 159–184. 1995.
3. K.S. Booth, G.S. Lueker. Testing for the consecutive ones property, interval graphs, and graph planarity. *J. Comput. Syst. Sci.* 13, pp. 335–379. 1976.

4. 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, paper e1000234. 2008.
5. M. Habib, R.M. McConnell, C. Paul, L. Viennot. Lex-BFS and partition refinement, with applications to transitive orientation, interval graph recognition and consecutive ones testing. *Theoret. Comput. Sci.* 234, pp. 59–84. 2000.
6. G.M. Landau, L. Parida, O. Weimann. Gene proximity analysis across whole genomes via PQ trees. *J. Comput. Biol.* 12, pp. 1289–1306. 2005.
7. W.-F. Lu, W.-L. Hsu. A test for the Consecutive Ones Property on noisy data – application to physical mapping and sequence assembly. *J. Comput. Biol.* 10, pp. 709–735. 2003.
8. J. Ma, L. Zhang, B.B. Suh, B.J. Raney, R.C. Burhans, W.J. Kent, M. Blanchette, D. Haussler, W. Miller. Reconstructing contiguous regions of an ancestral genome. *Genome Res.* 16, pp. 1557–1565. 2006.
9. R.M. McConnell. A certifying algorithm for the consecutive-ones property. In *SODA 2004*, pp. 761–770. ACM, 2004.
10. J. Meidanis, O. Porto, G.P. Telles. On the consecutive ones property. *Discrete Appl. Math.* 88, pp. 325–354. 1998.
11. W.-L. Hsu. A simple test for the Consecutive Ones Property. *J. Algorithms* 43, pp. 1–16. 2002.
12. R. Wittler, J. Stoye. Consistency of sequence-based gene clusters. In *RECOMB-CG 2010*, vol. 6398 of *LNCS*, pp. 252–263. Springer, 2010.
13. R. Wittler, J. Maňuch, M. Patterson, J. Stoye. Consistency of sequence-based gene clusters. unpublished manuscript.

Appendix A Ancestral Gene Orders and Telomeres

In the context of ancestral genome reconstruction as approached in [4], $\mathcal{C}$ is an alphabet of genomic markers that are believed to appear uniquely in the extinct ancestral genome. An *ancestral synteny* is a set of markers that are believed to have been consecutive along a chromosome of the ancestor. A set of ancestral syntenies can then be represented by a binary matrix M: columns represent markers and the `1` entries of a given row define an ancestral synteny. If all ancestral syntenies are true positives (i.e., represent sets of markers that were consecutive in the ancestor), then M has the C1P. Otherwise, some ancestral syntenies are false positives, and a general approach to handle such conflicts is to remove from M some rows (optimizing some criterion) such that the resulting matrix M' has the C1P. Each subtree rooted at a child of the root of the resulting PQ-tree represents a set of markers that are believed to have been contiguous in the ancestral genome (with partial information regarding the order of the markers along this segment), called a CAR (Contiguous Ancestral Region) following [8].

A CAR is an ancestral chromosomal segment, but it is not guaranteed to be a complete ancestral chromosome. In fact, it is common that the number of CARs obtained is larger than the expected number of ancestral chromosomes. This raises the following natural question: which CARs are believed to form complete ancestral chromosomes, or more generally, to contain an extremity of an ancestral chromosome (an ancestral telomere)? Indeed, a CAR with two ancestral telomeres is in fact a complete ancestral chromosome. Moreover, when CARs are grouped into syntenic sets, that is, sets of CARs that are believed to belong to the same ancestral chromosome, each such syntenic set of CARs can contain only two ancestral telomeres.

We address this question as follows. A column c' with multiplicity (bounded, for example, by twice the maximum expected number of ancestral chromosomes, or more generally with infinite multiplicity) can then be used to represent telomeres, that is, virtual extremities of ancestral chromosomes. Then any ancestral synteny that contains putatively a marker that is an extremity of an ancestral chromosome (for example because the ancestral synteny is telomeric in two extant descendants of the considered ancestor) can be represented by two rows in M: a row representing the ancestral synteny, plus a copy of this row with an additional entry `1` in column c' (hence M has matched multirows). This structure, as seen earlier, ensures that if M has the mC1P, then the occurrences of c are located at the extremities of the CARs. Otherwise (M does not have the mC1P), some rows can be discarded to result in a matrix M' that has the mC1P,

with the same property. The assumption that M has matched multirows is fundamental to leave open the possibility for any ancestral synteny to be at the extremity of a CAR or to be embedded inside a CAR.

Considering several columns with multiplicity can be used to model more precise knowledge about possible ancestral telomeres, provided that the fundamental assumption that the matrix M has matched multirows is maintained, and that any ancestral synteny (i.e., row) contains at most one putative ancestral telomere, which are the assumptions of our main result.