Recognition and Orientation Algorithms for P_4-Comparability Graphs

Stavros D. Nikolopoulos and Leonidas Palios

Department of Computer Science, University of Ioannina
GR-45110 Ioannina, Greece
{stavros,palios}@cs.uoi.gr

Abstract. We consider two problems pertaining to P_4-comparability graphs, namely, the problem of recognizing whether a simple undirected graph is a P_4-comparability graph and the problem of producing an acyclic P_4-transitive orientation of a P_4-comparability graph. These problems have been considered by Hoàng and Reed who described $O(n^4)$ and $O(n^5)$-time algorithms for their solution respectively, where n is the number of vertices of the given graph. Recently, Raschle and Simon described $O(n + m^2)$-time algorithms for these problems, where m is the number of edges of the graph.

In this paper, we describe different $O(n + m^2)$-time algorithms for the recognition and the acyclic P_4-transitive orientation problems on P_4-comparability graphs. Instrumental in these algorithms are structural relationships of the P_4-components of a graph, which we establish and which are interesting in their own right. Our algorithms are simple, use simple data structures, and have the advantage over those of Raschle and Simon in that they are non-recursive, require linear space and admit efficient parallelization.

1 Introduction

Let $G = (V, E)$ be a simple non-trivial undirected graph. An *orientation* of the graph G is an antisymmetric directed graph obtained from G by assigning a direction to each edge of G. An orientation (V, F) of G is called *transitive* if it satisfies the following condition: if abc is a chordless path on 3 vertices in G, then F contains the directed edges $\overrightarrow{ab}$ and $\overleftarrow{bc}$, or $\overleftarrow{ab}$ and $\overrightarrow{bc}$, where $\overrightarrow{uv}$ or $\overleftarrow{vu}$ denotes an edge directed from u to v [4]. An orientation of a graph G is called *P_4-transitive* if the orientation of every chordless path on 4 vertices of G is transitive; an orientation of such a path $abcd$ is transitive if and only if the path's edges are oriented in one of the following two ways: $\overrightarrow{ab}$, $\overleftarrow{bc}$ and $\overrightarrow{cd}$, or $\overleftarrow{ab}$, $\overrightarrow{bc}$ and $\overleftarrow{cd}$. The term borrows from the fact that a chordless path on 4 vertices is denoted by P_4.

A graph which admits an acyclic transitive orientation is called a *comparability graph* [3,4]; A graph is a *P_4-comparability graph* if it admits an acyclic P_4-transitive orientation [5,6]. In light of these definitions, every comparability graph is a P_4-comparability graph. Moreover, there exist P_4-comparability

P. Eades and T. Takaoka (Eds.): ISAAC 2001, LNCS 2223, pp. 320–331, 2001.

graphs which are not comparability. The class of the P_4-comparability graphs (along with the P_4-indifference, the P_4-simplicial and the Raspail graphs) was introduced by Hoàng and Reed [6].

Algorithms for many different problems (such as, recognition, coloring, maximum clique, maximum independent set, hamiltonian paths and cycles) on subclasses of perfectly orderable graphs are available in the literature. The comparability graphs in particular have been the focus of much research which culminated into efficient recognition and orientation algorithms [4,7,8,12]. On the other hand, the P_4-comparability graphs have not received as much attention, despite the fact that the definitions of the comparability and the P_4-comparability graphs rely on the same principles [1,2,5,6,11].

Our main objective is to study the recognition and acyclic P_4-transitive orientation problems on the class of P_4-comparability graphs. These problems have been addressed by Hoàng and Reed who described $O(n^4)$ and $O(n^5)$-time algorithms respectively [5,6], where n is the number of vertices of G. Recently, newer results on these problems were provided by Raschle and Simon [11]. Their algorithms work along the same lines, but they focus on the P_4-components of the graph. The time complexity of their algorithms for either problem is $O(n+m^2)$, where m is the number of edges of G, as it is dominated by the time to compute the P_4-components of G. Raschle and Simon also described recognition and orientation algorithms for P_4-indifference graphs [11]; their algorithms run within the same time complexity, i.e., $O(n+m^2)$. We note that Hoàng and Reed [5,6] also presented algorithms which solve the recognition problem for P_4-indifference graphs in $O(n^6)$ time.

In this paper, we present different $O(n+m^2)$-time recognition and acyclic P_4-transitive orientation algorithms for P_4-comparability graphs of n vertices and m edges. Our technique relies on the computation of the P_4-components of the input graph and takes advantage of structural relationships of these components. Our algorithms are simple, use simple data structures, and have the advantage over those of Raschle and Simon in that they are non-recursive, require linear space and admit efficient parallelization [10].

2 Theoretical Framework

Let $abcd$ be a P_4 of a graph G. The vertices b and c are called *midpoints* and the vertices a and d *endpoints* of the P_4 $abcd$. The edge connecting the midpoints of a P_4 is called the *rib*; the other two edges (which are incident to the endpoints) are called the *wings*. For example, the edge bc is the rib and the edges ab and cd are the wings of the P_4 $abcd$. Two P_4s are called *adjacent* if they have an edge in common. The transitive closure of the adjacency relation is an equivalence relation on the set of P_4s of a graph G; the subgraphs of G spanned by the edges of the P_4s in the equivalence classes are the *P_4-components* of G. With slight abuse of terminology, we consider that an edge which does not belong to any P_4 belongs to a P_4-component by itself; such a component is called *trivial*. A P_4-component which is not trivial is called *non-trivial*; clearly a non-trivial P_4-component contains at least one P_4. If the set of midpoints and the set of endpoints of the P_4s of a non-trivial P_4-component $\mathcal{C}$ define a partition of the vertex set $V(\mathcal{C})$, then the P_4-component $\mathcal{C}$ is called *separable*. We can show [9]:

Lemma 1. *Let $G = (V, E)$ be a graph and let $\mathcal{C}$ be a non-trivial P_4-component of G. Then,*

(i) If ρ and ρ' are two P_4s which both belong to $\mathcal{C}$, then there exists a sequence ρ, ρ_1, ..., ρ_k, ρ' of adjacent P_4s in $\mathcal{C}$;
(ii) $\mathcal{C}$ is connected;

The definition of a P_4-comparability graph requires that such a graph admit an acyclic P_4-transitive orientation. However, Hoàng and Reed [6] showed that in order to determine whether a graph is P_4-comparability one can restrict one's attention to the P_4-components of the graph. In particular, what they proved ([6], Theorem 3.1) can be paraphrased in terms of the P_4-components as follows:

Lemma 2. [6] *Let G be a graph such that each of its P_4-components admits an acyclic P_4-transitive orientation. Then G is a P_4-comparability graph.*

Although determining that each of the P_4-components of a graph admits an acyclic P_4-transitive orientation suffices to establish that the graph is P_4-comparability, the directed graph produced by placing the oriented P_4-components together may contain cycles. However, an acyclic P_4-transitive orientation of the entire graph can be obtained by inversion of the orientation of some of the P_4-components. Therefore, if one wishes to compute an acyclic P_4-transitive orientation of a P_4-comparability graph, one needs to detect directed cycles (if they exist) formed by edges belonging to more than one P_4-component and appropriately invert the orientation of one or more of these P_4-components. Fortunately, one does not need to consider arbitrarily long cycles as shown in the following lemma [6].

Lemma 3. ([6], Lemma 3.5) *If a proper orientation of an interesting graph is cyclic, then it contains a directed triangle.*[1]

Given a non-trivial P_4-component $\mathcal{C}$ of a graph $G = (V, E)$, the set of vertices $V - V(\mathcal{C})$ can be partitioned into three sets:

(i) R contains the vertices of $V - V(\mathcal{C})$ which are adjacent to some (but not all) of the vertices in $V(\mathcal{C})$,
(ii) P contains the vertices of $V - V(\mathcal{C})$ which are adjacent to all the vertices in $V(\mathcal{C})$, and
(iii) Q contains the vertices of $V - V(\mathcal{C})$ which are not adjacent to any of the vertices in $V(\mathcal{C})$.

The adjacency relation is considered in terms of the given graph G.

[1] An orientation is proper if the orientation of every P_4 is transitive. A graph is interesting if the orientation of every P_4-component is acyclic.

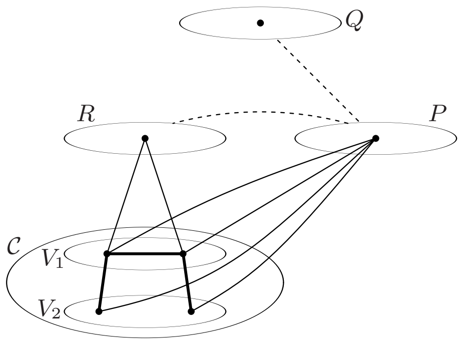

Fig. 1. Partition of the vertex set with respect to a separable P_4-component $\mathcal{C}$

In [11], Raschle and Simon showed that, given a non-trivial P_4-component $\mathcal{C}$ and a vertex $v \notin V(\mathcal{C})$, if v is adjacent to the midpoints of a P_4 of $\mathcal{C}$ and is not adjacent to its endpoints, then v does so with respect to every P_4 in $\mathcal{C}$ (that is, v is adjacent to the midpoints and not adjacent to the endpoints of every P_4 in $\mathcal{C}$). This implies that any vertex of G, which does not belong to $\mathcal{C}$ and is adjacent to at least one but not all the vertices in $V(\mathcal{C})$, is adjacent to the midpoints of all the P_4s in $\mathcal{C}$. Based on that, Raschle and Simon showed that:

Lemma 4. ([11], Corollary 3.3) *Let $\mathcal{C}$ be a non-trivial P_4-component and $R \neq \emptyset$. Then, $\mathcal{C}$ is separable and every vertex in R is V_1-universal and V_2-null*[2]*. Moreover, no edge between R and Q exists.*

The set V_1 is the set of the midpoints of all the P_4s in $\mathcal{C}$, whereas the set V_2 is the set of endpoints. Figure 1 shows the partition of the vertices of a graph with respect to a separable P_4-component $\mathcal{C}$; the dashed segments between R and P and P and Q indicate that there may be edges between pairs of vertices in the corresponding sets. Then, a P_4 with at least one but not all its vertices in $V(\mathcal{C})$ must be a P_4 of one of the following types:

type (1)	vpq_1q_2	where $v \in V(\mathcal{C}),\ p \in P,\ q_1, q_2 \in Q$
type (2)	p_1vp_2q	where $p_1 \in P,\ v \in V(\mathcal{C}),\ p_2 \in P,\ q \in Q$
type (3)	$p_1v_2p_2r$	where $p_1 \in P,\ v_2 \in V_2,\ p_2 \in P,\ r \in R$
type (4)	$v_2pr_1r_2$	where $v_2 \in V_2,\ p \in P,\ r_1, r_2 \in R$
type (5)	rv_1pq	where $r \in R,\ v_1 \in V_1,\ p \in P,\ q \in Q$
type (6)	rv_1pv_2	where $r \in R,\ v_1 \in V_1,\ p \in P,\ v_2 \in V_2$
type (7)	$rv_1v_2v_2'$	where $r \in R,\ v_1 \in V_1,\ v_2, v_2' \in V_2$
type (8)	$v_1'rv_1v_2$	where $r \in R,\ v_1, v_1' \in V_1,\ v_2 \in V_2$

Raschle and Simon proved that neither a P_3 abc with $a \in V_1$ and $b, c \in V_2$ nor a $\overline{P_3}$ abc with $a, b \in V_1$ and $c \in V_2$ exists ([11], Lemma 3.4), which implies that:

[2] For a set A of vertices, we say that a vertex v is A-universal if v is adjacent to every element of A; a vertex v is A-null if v is adjacent to no element of A.

Lemma 5. *Let $\mathcal{C}$ be a non-trivial P_4-component of a graph $G = (V, E)$. Then, no P_4s of type (7) or (8) with respect to $\mathcal{C}$ exist.*

Additionally, Raschle and Simon proved the following interesting result regarding the P_4-components.

Lemma 6. ([11], Theorem 3.6) *Two different P_4-components have different vertex sets.*

Moreover, we can show the following [9]:

Lemma 7. *Let $\mathcal{A}$ and $\mathcal{B}$ be two non-trivial P_4-components of the graph G. If the component $\mathcal{A}$ contains an edge e both endpoints of which belong to the vertex set $V(\mathcal{B})$ of $\mathcal{B}$, then $V(\mathcal{A}) \subseteq V(\mathcal{B})$.*

Let us consider a non-trivial P_4-component $\mathcal{C}$ of the graph G such that $V(\mathcal{C}) \subset V$, and let $S_\mathcal{C}$ be the set of non-trivial P_4-components of G which have a vertex in $V(\mathcal{C})$ and a vertex in $V - V(\mathcal{C})$. Then, each component in $S_\mathcal{C}$ contains a P_4 of type (1)-(8), and thus, by taking Lemma 5 into account, we can partition the elements of $S_\mathcal{C}$ into two sets as follows:

- *P_4-components of type A:* the P_4 components, each of which contains at least one P_4 of type (1)-(5) with respect to $\mathcal{C}$;
- *P_4-components of type B:* the P_4-components which contain only P_4s of type (6) with respect to $\mathcal{C}$.

The following lemmata establish properties of P_4-components of type A and of type B (the proofs are omitted due to lack of space but can be found in [9]).

Lemma 8. *Let $\mathcal{C}$ be a non-trivial P_4-component of a P_4-comparability graph $G = (V, E)$ and suppose that the vertices in $V - V(\mathcal{C})$ have been partitioned into sets R, P, and Q as described earlier in this section. Then, if there exists an edge xv (where $x \in R \cup P$ and $v \in V(\mathcal{C})$) that belongs to a P_4-component $\mathcal{A}$ of type A, then all the edges, which connect the vertex x to a vertex in $V(\mathcal{C})$, belong to $\mathcal{A}$. Moreover, these edges are all oriented towards x or they are all oriented away from x.*

Lemma 9. *Let $\mathcal{B}$ and $\mathcal{C}$ be two non-trivial P_4-components of the graph G such that $\mathcal{B}$ is of type B with respect to $\mathcal{C}$. Then,*

(i) every edge of $\mathcal{B}$ has exactly one endpoint in $V(\mathcal{C})$;
(ii) if an edge of $\mathcal{B}$ is oriented towards its endpoint that belongs to $V(\mathcal{C})$, then so do all the edges of $\mathcal{B}$;
(iii) the edges of $\mathcal{B}$ incident upon the same vertex v are all oriented either towards v or away from it.

Lemma 10. *Let $\mathcal{B}$ and $\mathcal{C}$ be two non-trivial P_4-components of the graph G such that $|V(\mathcal{B})| \geq |V(\mathcal{C})|$ and let $\beta = \sum_{v \in V(\mathcal{C})} d_\mathcal{B}(v)$, where $d_\mathcal{B}(v)$ denotes the number of edges of $\mathcal{B}$ which are incident upon v. Then, $\mathcal{B}$ is of type B with respect to $\mathcal{C}$ if and only if $\beta = |E(\mathcal{B})|$.*

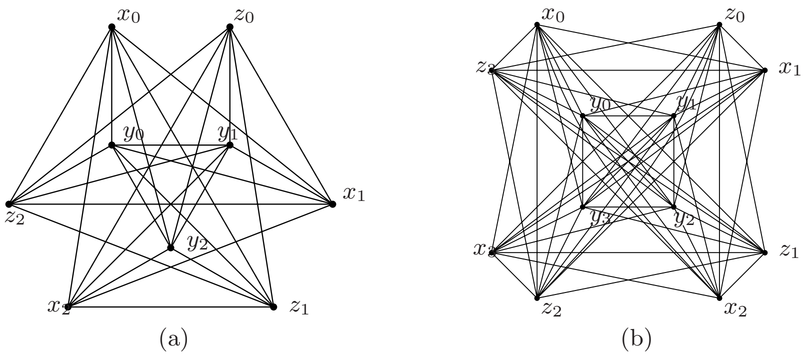

Fig. 2. Graphs that have P_4-components with cyclic P_4-transitive orientation

Lemma 11. *Let $\mathcal{A}$, $\mathcal{B}$, and $\mathcal{C}$ be three distinct separable P_4-components such that $\mathcal{A}$ is of type B with respect to $\mathcal{B}$, $\mathcal{B}$ is of type B with respect to $\mathcal{C}$, and $|V(\mathcal{A})| \geq |V(\mathcal{C})|$. Then, if there exists a vertex which is a midpoint of all three components $\mathcal{A}$, $\mathcal{B}$, and $\mathcal{C}$, the P_4-component $\mathcal{A}$ is of type B with respect to $\mathcal{C}$.*

We close this section by showing that the assignment of compatible directions in all the P_4s of a P_4-component does not imply that the component is necessarily acyclic. Let k be an integer at least equal to 3, and let $X_k = \{x_i \,|\, 0 \leq i < k\}$, $Y_k = \{y_i \,|\, 0 \leq i < k\}$, and $Z_k = \{z_i \,|\, 0 \leq i < k\}$ be three sets of distinct vertices. We consider the graph $G_k = (V_k, E_k)$ where

$$\begin{aligned} V_k &= X_k \cup Y_k \cup Z_k \\ \text{and} \quad E_k &= V_k \times V_k \\ &\quad - \Big(\{x_i y_{i+1} \,|\, 0 \leq i < k\} \cup \{x_i z_i \,|\, 0 \leq i < k\} \cup \{y_i z_i \,|\, 0 \leq i < k\} \Big). \end{aligned}$$

(The addition in the subscripts is assumed to be done mod k.) Figures 2(a) and 2(b) depict G_3 and G_4 respectively. The graph G_k has the following properties:

- ▷ the only P_4s of G_k are the paths $x_i y_i y_{i+1} z_i$, $y_{i+1} z_i z_{i+1} x_i$, and $y_{i+1} x_{i+1} x_i z_{i+1}$ for $0 \leq i < k$;
- ▷ G_k has a single non-trivial P_4-component;
- ▷ the directed edges $y_i y_{i+1}$ ($0 \leq i < k$) form a directed cycle of length k in the non-trivial P_4-component of G_k;
- ▷ no directed cycle of length less than k exists in the non-trivial P_4-component of G_k.

3 Recognition of P_4-Comparability Graphs

The main idea of the algorithm is to build the P_4-components of the given graph G by considering all the P_4s of G; this is achieved by unioning in a single P_4-component the P_4-components of the edges of each such path, while it is made sure that the edges are compatibly oriented. It is important to note that the orientation of two edges in the same P_4-component is not free to change relative to each other; either the orientation of all the edges in the component stays the same or it is inverted for all the edges. If no compatible orientation can be found or if the resulting P_4-components contain directed cycles, then the graph is not a P_4-comparability graph. The P_4s are produced by means of BFS-traversals of the graph G starting from each of G's vertices.

The algorithm is described in more detail below. Initially, each edge of G belongs to a P_4-component by itself.

Recognition Algorithm.

1. For each vertex v of the graph, we construct the BFS-tree T_v rooted at v and we update the level $level(x)$[3] and the parent p_x of each vertex x in T_v; before the construction of each of the BFS-trees, $level(x) = -1$ for each vertex x of the graph. Then, we process the edges of the graph as follows:
   (i) for each edge $e = uw$ where $level(u) = 1$ and $level(w) = 2$, we check whether there exist edges from w to a vertex in the 3rd level of T_v. If not, then we do nothing. Otherwise, we orient the edges vu, uw, vp_w, and $p_w w$ in a compatible fashion; for example, we orient vu and vp_w away from v, and uw and $p_w w$ away from w (note that if $u = p_w$, we end up processing the edges vu and uw only). If any two of these edges belong to the same P_4-component and have incompatible orientations, then we conclude that the graph G is not a P_4-comparability graph. If any two of these edges belong to different P_4-components, then we union these components into a single component; if the edges do not have compatible orientations, then we invert (during the unioning) the orientation of all the edges of one of the unioned P_4-components.
   (ii) for each edge $e = uw$ where $level(u) = i$ and $level(w) = i + 1$ for $i \geq 2$, we consider the edges $p_u u$ and uw. As in the previous case, if the two edges belong to the same P_4-component and they are not both oriented towards u or away from u, then there is no compatible orientation assignment and the graph is not a P_4-comparability graph. If the two edges belong to different P_4-components, we union the corresponding P_4-components in a single component, while making sure that the edges are oriented in a compatible fashion.
   (iii) for each edge $e = uw$ where $level(u) = level(w) = 2$, we go through all the vertices of the 1st level of T_v. For each such vertex x, we check whether x is adjacent to u or w. If x is adjacent to u but not to w, then the edges vx, xu, and uw form a P_4; we therefore union the corresponding P_4-components while orienting their edges compatibly. We work similarly for the case where x is adjacent to w but not to u, since the edges vx, xw, and wu form a P_4.
2. After all the vertices have been processed, we check whether the resulting non-trivial P_4-components contain directed cycles. This is done by applying topological sorting independently in each of the P_4components; if the

[3] The level of the root of a tree is equal to 0.

topological sorting succeeds then the corresponding component is acyclic, otherwise there is a directed cycle. If any of the P_4-components contains a cycle, then the graph is not a P_4-comparability graph.

For each P_4-component, we maintain a linked list of the records of the edges in the component, and the total number of these edges. Each edge record contains a pointer to the header record of the component to which the edge belongs; in this way, we can determine in constant time the component to which an edge belongs and the component's size. Unioning two P_4-components is done by updating the edge records of the smallest component and by linking them to the edge list of the largest one, which implies that the union operation takes time linear in the size of the smallest component. As mentioned above, in the process of unioning, we may have to invert the orientation in the edge records that we link, if the current orientations are not compatible.

The correctness of the algorithm follows from the fact that all the P_4s of the given graph are taken into account (see [9], Lemma 3.1), from the correct orientation assignment on the edges of these paths, and from Lemma 2 in conjunction with Step 2 of the algorithm.

Time and Space Complexity. Computing the BFS-tree T_v of the vertex v of G takes $O(1+m(v)) = O(1+m)$ time, where $m(v)$ is the number of edges in the connected component of G to which v belongs. Processing the tree T_v includes processing the edges and checking for compatible orientation, and unioning P_4-components. If we ignore P_4-component unioning, then, each of the Steps 1(i) and 1(ii) takes constant time per processed edge; the parent of a vertex in the tree can be determined in constant time with the use of an auxiliary array, and the P_4-component of an edge is determined in constant time by means of the pointer to the component head record (these pointers are updated during unioning). The Step 1(iii) of the algorithm takes time $O(\deg(v))$ for each edge in the 2nd level of the tree, where by $\deg(v)$ we denote the degree of the vertex v; this implies a total of $O(m\deg(v))$ time for the Step 1(iii) for the tree T_v. Now, the time required for all the P_4-component union operations during the processing of all the BFS-trees is $O(m\log m)$; there cannot be more than $m-1$ such operations (we start with m P_4-components and we may end up with only one), and each one of them takes time linear in the size of the smallest of the two components that are unioned. Finally, checking whether a non-trivial P_4-component is acyclic takes $O(1+m_i)$, where m_i is the number of edges of the component. Thus, the total time taken by Step 2 is $O\big(\sum_i(1+m_i)\big) = O(m)$, since there are at most m P_4-components and $\sum_i m_i = m$. Thus, the overall time complexity is $O\Big(\sum_v\big(1+m+m\deg(v)\big) + m\log m + m\Big) = O\big(n+m^2\big)$, since $\sum_v \deg(v) = 2m$.

The space complexity is linear in the size of the graph G; the information stored in order to help processing each BFS-tree is constant per vertex, and the handling of the P_4-components requires one record per edge and one record per component. Thus, the space required is $O(n+m)$.

Theorem 1. *It can be decided whether a simple graph on n vertices and m edges is a P_4-comparability graph in $O\big(n+m^2\big)$ time and $O(n+m)$ space.*

Remark. It must be noted that there are simpler ways of producing the P_4s of a graph in $O(n + m^2)$ time. However, such approaches require $\Theta(n^2)$ space. For example, Raschle and Simon note that a P_4 is uniquely determined by its wings [11]; this implies that the P_4s can be determined by considering all $\Theta(m^2)$ pairs of edges and by checking if the edges in each such pair are the wings of a P_4. In order not to exceed the $O(m^2)$ time complexity, the information on whether two vertices are adjacent should be available in constant time, something that necessitates a $\Theta(n^2)$-space adjacency matrix.

4 Acyclic P_4-Transitive Orientation

Although each of the P_4-components of the given graph G which is produced by the recognition algorithm is acyclic, directed cycles may arise when all the P_4-components are placed together; obviously, these cycles will include edges from more than one P_4-component. Appropriate inversion of the orientation of some of the components will yield the desired acyclic P_4-transitive orientation.

Our algorithm to compute the acyclic P_4-transitive orientation of a P_4-comparability graph relies on the processing of the P_4-components of the given graph G and focuses on edges incident upon the vertices of the non-trivial P_4-component which is currently being processed. It assigns orientations in a greedy fashion, and avoids both the contraction step and the recursive call of the orientation algorithms of Hoàng and Reed [6], and Raschle and Simon [11]. More specifically, the algorithm works as follows:

Orientation Algorithm.

1. We apply the recognition algorithm of the previous section on the given graph G, which produces the P_4-components of G and an acyclic P_4-transitive orientation of each component.
2. We sort the non-trivial P_4-components of G by increasing number of vertices; let $\mathcal{C}_1, \mathcal{C}_2, \ldots, \mathcal{C}_h$ be the resulting ordered sequence. We associate with each $\mathcal{C}_i$ a `mark` and a `counter` field which are initialized to 0.
3. For each P_4-component $\mathcal{C}_i$ $(1 \leq i < h)$ in order, we do:
   By going through the vertices in $V(\mathcal{C}_i)$, we collect the edges which are incident upon a vertex in $V(\mathcal{C}_i)$ and belong to a P_4-component $\mathcal{C}_j$ where $j > i$. Then, for each such edge e, we increment the `counter` field associated with the P_4-component to which e belongs. Next, we go through the collected edges once more. This time, for such an edge e, we check whether the P_4-component to which e belongs has its `mark` field equal to 0 and its `counter` field equal to the total number of edges of the component; if yes, then we set the `mark` field of the component to 1, and, in case e is not oriented towards its endpoint in $V(\mathcal{C}_i)$, we flip the component's orientation (by updating a corresponding boolean variable). After that, we set the `counter` field of the component to which e belongs to 0; in this way, the `counter` fields of all the non-trivial P_4-components are equal to 0 every time a P_4-component starts getting processed in Step 3.

4. We orient the edges which belong to the trivial P_4-components: this can be easily done by topologically sorting the vertices of G using only the oriented edges of the non-trivial components, and orienting the remaining edges in accordance with the topological order of their incident vertices.

Note that in Step 3 we process all the non-trivial P_4-components of the given graph G except for the largest one. This implies that the vertex set $V(\mathcal{C}_i)$ of each P_4-component $\mathcal{C}_i$ $(1 \leq i < h)$ that we process is a proper subset of the vertex set V of G; if $V(\mathcal{C}_i) = V$, then $V(\mathcal{C}_h) = V$ as well, which implies that $\mathcal{C}_i = \mathcal{C}_h$ (Lemma 6), a contradiction. Thus, the discussion in Section 2 regarding the P_4-components of type A and type B applies to each such $\mathcal{C}_i$. Moreover, according to Lemma 10, the P_4-components whose `mark` field is set to 1 in Step 3 are components which are of type B with respect to the currently processed component $\mathcal{C}_i$. Each edge of these components has exactly one endpoint in $V(\mathcal{C}_i)$ (see Lemma 9, statement (i)), so that it is valid to try to orient such an edge towards that endpoint. Furthermore, Lemma 9 (statement (ii)) implies that if such an edge gets oriented towards its endpoint which belongs to $V(\mathcal{C}_i)$, then so do all the edges of the same P_4-component. In the case that the set R in the partition of the vertices in $V - V(\mathcal{C}_i)$ (as described in Section 2) is empty, there are no P_4-components of type B with respect to $\mathcal{C}_i$. While processing $\mathcal{C}_i$, our algorithm updates the `counter` fields of the components that contain an edge incident upon a vertex in $V(\mathcal{C}_i)$, finds that none of these components ends up having its `counter` field equal to the number of its edges, and thus does nothing further.

The orientation algorithm does not compute the sets R, P, and Q with respect to the currently processed P_4-component $\mathcal{C}_i$. These sets can be computed in $O(n)$ time for each $\mathcal{C}_i$ as follows. We use an array with one entry per vertex of the graph G; we initialize the array entries corresponding to vertices in $V(\mathcal{C}_i)$ to 0 and all the remaining ones to -1. Let v_1 and v_2 be an arbitrary midpoint and an arbitrary endpoint of a P_4 in $\mathcal{C}_i$. We go through the vertices adjacent to v_1 and if the vertex does not belong to $V(\mathcal{C}_i)$, we set the corresponding entry to 1. Next, we go through the vertices adjacent to v_2; this time, if the vertex does not belong to $V(\mathcal{C}_i)$, we increment the corresponding entry. Then, the vertices in $\mathcal{C}_i$, R, and Q are the vertices whose corresponding array entries are equal to 0, 1, and -1 respectively, while the remaining vertices belong to P and their corresponding entries are larger than 1; recall that every vertex in $V - V(\mathcal{C}_i)$ which is adjacent to an endpoint of a P_4 of $\mathcal{C}_i$ is also adjacent to any midpoint.

Correctness of the Algorithm. The acyclicity of the directed graph produced by our orientation algorithm relies on the following two lemmata (proofs can be found in [9]).

Lemma 12. *Let $\mathcal{C}_1, \mathcal{C}_2, \ldots, \mathcal{C}_h$ be the sequence of the non-trivial P_4-components of the given graph G ordered by increasing vertex number. Consider the set $S_i = \{\mathcal{C}_j \mid j < i$ and $\mathcal{C}_i$ is of type B with respect to $\mathcal{C}_j\}$ and suppose that $S_i \neq \emptyset$. If $\hat{\imath} = \min\{j \mid \mathcal{C}_j \in S_i\}$, then our algorithm orients the edges of the component $\mathcal{C}_i$ towards their endpoint which belongs to $V(\mathcal{C}_{\hat{\imath}})$.*

Lemma 13. *Let $\mathcal{C}_1$, $\mathcal{C}_2$, …, $\mathcal{C}_h$ be the non-trivial P_4-components of a graph G ordered by increasing vertex number and suppose that each component has received an acyclic P_4-transitive orientation. Consider the set $S_i = \{\mathcal{C}_j \mid j < i$ and $\mathcal{C}_i$ is of type B with respect to $\mathcal{C}_j\}$. If the edges of each P_4-component $\mathcal{C}_i$ such that $S_i \neq \emptyset$ get oriented towards their endpoint which belongs to $V(\mathcal{C}_{\hat{\imath}})$, where $\hat{\imath} = \min\{j \,|\, \mathcal{C}_j \in S_i\}$, then the resulting directed subgraph of G spanned by the edges of the $\mathcal{C}_i$s $(1 \leq i \leq h)$ does not contain a directed cycle.*

Theorem 2. *When applied to a P_4-comparability graph, our orientation algorithm produces an acyclic P_4-transitive orientation.*

Proof: The application of the recognition algorithm in Step 1 of the orientation algorithm and the fact that thereafter the inversion of the orientation of an edge causes the inversion of the orientation of all the edges in the same P_4-component imply that the resulting orientation is P_4-transitive. The proof of the theorem will be complete if we show that it is also acyclic. Since the edges of the trivial P_4-components do not introduce cycles given that they are oriented according to a topological sorting of the vertices of the graph, it suffices to show that the directed subgraph of G spanned by the edges of the non-trivial P_4-components of G, which results after the last execution of Step 3, is acyclic. This follows directly from Lemmata 12 and 13. □

Time and Space Complexity. As described in the previous section, Step 1 of the algorithm can be completed in $O(n + m^2)$ time. Step 2 takes $O(m \log m)$ time, since there are $O(m)$ non-trivial P_4-components. Since the degree of a vertex of the graph does not exceed $n - 1$, the total number of edges processed while processing the P_4-component $\mathcal{C}_i$ in Step 3 is $O(n\,|V(\mathcal{C}_i)|)$, where $|V(\mathcal{C}_i)|$ is the cardinality of the vertex set of $\mathcal{C}_i$. This upper bound is $O\big(n\,(|E(\mathcal{C}_i)| + 1)\big) = O(n\,|E(\mathcal{C}_i)|)$, because the component $\mathcal{C}_i$ is connected (Lemma 1, statement (ii)) and hence $|V(\mathcal{C}_i)| \leq |E(\mathcal{C}_i)| + 1$. The time to process each such edge is $O(1)$, thus implying a total of $O(n\,|E(\mathcal{C}_i)|)$ time for the execution of Step 3 for the component $\mathcal{C}_i$; since an edge of the graph belongs to one P_4-component and a component is processed only once, the overall time for all the executions of Step 3 is $O(nm)$. Finally, Step 4 takes $O(n + m)$ time.

Summarizing, the time complexity of the orientation algorithm is $O(n+m^2)$. It is interesting to note that the time complexity is dominated by the time to execute Step 1; the remaining steps take a total of $O(nm)$ time. Therefore, an $o(n+m^2)$-time algorithm to recognize a P_4-comparability graph and to compute its P_4-components will imply an $o(n + m^2)$-time algorithm for the acyclic P_4-transitive orientation of a P_4-comparability graph. The space complexity is linear in the size of the given graph G.

From the above discussion, we obtain the following theorem.

Theorem 3. *Let G be a P_4-comparability graph on n vertices and m edges. Then, an acyclic P_4-transitive orientation of G can be computed in $O(n + m^2)$ time and $O(n + m)$ space.*

5 Concluding Remarks

In this paper, we presented an $O(n + m^2)$-time and linear space algorithm to recognize whether a graph of n vertices and m edges is a P_4-comparability graph. We also described an algorithm to compute an acyclic P_4-transitive orientation of a P_4-comparability graph which runs in $O(n + m^2)$ time and linear space as well. Both algorithms exhibit the currently best time and space complexities to the best of our knowledge, are simple enough to be easily used in practice, are non-recursive, and admit efficient parallelization.

The obvious open question is whether the P_4-comparability graphs can be recognized and oriented in $o(n + m^2)$ time. Note that a better time complexity for the recognition problem — assuming that the recognition process determines the P_4-components as well — will imply a better time complexity for our orientation algorithm.

References

1. L. Babel and S. Olariu, A new characterization of P_4-connected graphs, *Proc. 22nd International Workshop on Graph-theoretic concepts in Computer Science (WG '96)* (F. d' Amore, P. G. Franciosa, and A. Marchetti-Spaccamela, eds.), LNCS 1197, 1996, 17–30. 321
2. C. M. H. de Figueiredo, J. Gimbel, C. P. Mello, and J. L. Szwarcfiter, Even and odd pairs in comparability and in P_4-comparability graphs, *Discrete Appl. Math.* **91** (1999), 293–297. 321
3. P. C. Gilmore and A. J. Hoffman, A characterization of comparability graphs and of interval graphs, *Canad. J. Math.* **16** (1964), 539–548. 320
4. M. C. Golumbic, *Algorithmic graph theory and perfect graphs*, Academic Press, Inc., New York, 1980. 320, 321
5. C. T. Hoàng and B. A. Reed, Some classes of perfectly orderable graphs, *J. Graph Theory* **13** (1989), 445–463. 320, 321
6. C. T. Hoàng and B. A. Reed, P_4-comparability graphs, *Discrete Math.* **74** (1989), 173–200. 320, 321, 322, 328
7. R. M. McConnell and J. Spinrad, Linear-time modular decomposition and efficient transitive orientation of comparability graphs, *Proc. 5th Annual ACM-SIAM Symposium on Discrete Algorithms* (1994), 536–545. 321
8. R. M. McConnell and J. Spinrad, Linear-time transitive orientation, *Proc. 8th Annual ACM-SIAM Symposium on Discrete Algorithms* (1997), 19–25. 321
9. S. D. Nikolopoulos and L. Palios, Recognition and orientation algorithms for P_4-comparability graphs, *Technical Report 23-00*, 2000, University of Ioannina, Ioannina, Greece. 322, 324, 327, 329
10. S. D. Nikolopoulos and L. Palios, Parallel algorithms for P_4-comparability graphs, *Technical Report 20-01*, 2001, University of Ioannina, Ioannina, Greece. 321
11. T. Raschle and K. Simon, On the P_4-components of graphs, *Discrete Appl. Math.* **100** (2000), 215–235. 321, 323, 324, 328
12. J. P. Spinrad, On comparability and permutation graphs, *SIAM J. on Comput.* **14** (1985), 658–670. 321