Discrete Mathematics and Theoretical Computer Science DMTCS vol. **7**, 2005, 231–254

On the Recognition of Bipolarizable and P_4-simplicial Graphs

Stavros D. Nikolopoulos and Leonidas Palios

Department of Computer Science, University of Ioannina, P.O.Box 1186, GR-45110 Ioannina, Greece
{stavros, palios}@cs.uoi.gr

received Jul 18, 2004, revised Sep 20, 2005, accepted Oct 21, 2005.

The classes of Raspail (also known as Bipolarizable) and P_4-simplicial graphs were introduced by Hoàng and Reed who showed that both classes are perfectly orderable and admit polynomial-time recognition algorithms [16]. In this paper, we consider the recognition problem on these classes of graphs and present algorithms that solve it in $O(nm)$ time. In particular, we prove properties of the graphs investigated and show that we can produce bipolarizable and P_4-simplicial orderings on the vertices of the input graph G, if such orderings exist, working only on P_3s that participate in a P_4 of G. The proposed recognition algorithms are simple, use simple data structures and both require $O(n+m)$ space. Additionally, we show how our recognition algorithms can be augmented to provide certificates, whenever they decide that G is not bipolarizable or P_4-simplicial; the augmentation takes $O(n+m)$ time and space. Finally, we include a diagram on class inclusions and the currently best recognition time complexities for a number of perfectly orderable classes of graphs.

Keywords: Bipolarizable (Raspail) graphs, P_4-simplicial graphs, perfectly orderable graphs, recognition, algorithms, complexity.

1 Introduction

A linear order $\prec$ on the vertices of a graph G is *perfect* if the ordered graph $(G, \prec)$ contains no induced P_4 $abcd$ with $a \prec b$ and $d \prec c$ (such a P_4 is called an *obstruction*). In the early 1980s, Chvátal [4] defined the class of graphs that admit a perfect order and called them *perfectly orderable* graphs.

Chvátal proved that if a graph G admits a perfect order $\prec$, then the greedy coloring algorithm applied to $(G, \prec)$ produces an optimal coloring using only $\omega(G)$ colors, where $\omega(G)$ is the clique number of G. This implies that the perfectly orderable graphs are perfect; a graph G is *perfect* if for each induced subgraph H of G, the chromatic number $\chi(H)$ equals the clique number $\omega(H)$ of the subgraph H. The class of perfect graphs was introduced in the early 1960s by Berge [1], who also conjectured that a graph is perfect if and only if it contains no induced subgraph isomorphic to an odd cycle of length at least five, or to the complement of such an odd cycle. This conjecture, known as the *strong perfect graph conjecture*, has been recently established due to the work of Chudnovsky *et al.* [3].

It is well-known that several interesting problems in graph theory (e.g., coloring, independent set), which are NP-complete in general graphs, have polynomial-time solutions in graphs that admit a perfect

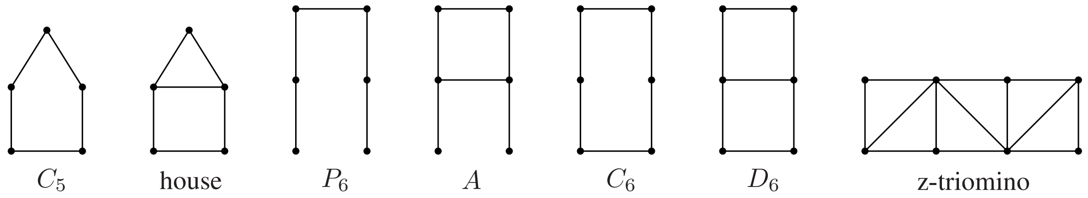


Fig. 1

order [2; 8]; unfortunately, it is NP-complete to decide whether a graph admits a perfect order [22]. Since the recognition of perfectly orderable graphs is NP-complete, we are interested in characterizing graphs which form polynomially recognizable subclasses of perfectly orderable graphs. Many such classes of graphs, with very interesting structural and algorithmic properties, have been defined so far and shown to admit polynomial-time recognitions (see [2; 8]); note however that not all subclasses of perfectly orderable graphs admit polynomial-time recognitions [13].

In 1989, Hoàng and Reed [16] introduced four subclasses of perfectly orderable graphs, namely, the P_4-indifference, Raspail (also known as Bipolarizable), P_4-simplicial, and P_4-comparability graphs; a graph is defined to be

- *P_4-indifference* if it admits a linear order $\prec$ on its vertices such that every P_4 $abcd$ has either ($a \prec b$, $b \prec c$, $c \prec d$) or ($d \prec c$, $c \prec b$, $b \prec a$);
- *Bipolarizable* if it admits a linear order $\prec$ on its vertices such that every P_4 $abcd$ has ($b \prec a$, $c \prec d$);
- *P_4-simplicial* if it admits a linear order $\prec$ such that every P_4 has either a bipolarizable or a P_4-indifference ordering;
- *P_4-comparability* if it admits a linear order $\prec$ on its vertices such that every P_4 $abcd$ has either ($a \prec b$, $c \prec b$, $c \prec d$) or ($b \prec a$, $b \prec c$, $d \prec c$).

Hoàng and Reed proved structural and algorithmic properties for these four classes of perfectly orderable graphs, and provided polynomial-time recognition algorithms and also polynomial-time algorithms for constructing obstruction-free linear orders (i.e., perfect orderings). Note that every linear order $\prec$ on the vertices of a graph yields an acyclic orientation of the edges, where each edge ab is oriented from a to b if and only if $a \prec b$; thus, a graph is bipolarizable if we can assign orientations to its edges so that the wings of every P_4 are oriented towards the endpoints of the P_4. On the other hand, every acyclic orientation gives at least one linear order; for example, the order taken by a topological sorting. Hence, bipolarizable and P_4-simplicial graphs can also be defined in terms of orientations.

Additionally, the class of bipolarizable graphs can be defined in terms of forbidden subgraphs. The minimal set of forbidden subgraphs for this class has been established in [12; 16]; it includes the graphs shown in Figure 1 (we will call the rightmost among them *zig-zag triomino* or *z-triomino* for short) and the *k-wheel* ($k \geq 3$) defined as follows (nearly following the notation of [12]): a *k-wheel* ($k \geq 3$) is the graph formed by a set of $3k$ vertices, namely, $v_0, v_1, \ldots, v_{k-1}$, $r_0, r_1, \ldots, r_{k-1}$, and $s_0, s_1, \ldots, s_{k-1}$, such that

- ▷ the vertices $v_0, v_1, \ldots, v_{k-1}$ form a clique, while each of the vertex sets $\{r_0, r_1, \ldots, r_{k-1}\}$ and $\{s_0, s_1, \ldots, s_{k-1}\}$ is an independent set,
- ▷ for $0 \leq i, j \leq k-1$, v_i is adjacent to r_j except for $j = i+1$,
- ▷ for $0 \leq i, j \leq k-1$, v_i is adjacent to s_j except for $j = i, i+1$, and
- ▷ for $0 \leq i \leq k-1$, r_i is adjacent to s_i but non-adjacent to any s_j for $j \neq i$,

where all integer subscripts are taken modulo k. We note that the 2-wheel is also well defined and coincides with P_6. A figure depicting the k-wheels for $k = 2, 3, 4, 6$ can be found in [12]. Unlike the bipolarizable graphs, the minimal set of forbidden subgraphs for the class of P_4-simplicial graphs, if it can be concisely described, has not been determined.

Another interesting class of graphs is the class of weak bipolarizable graphs: a graph is *weak bipolarizable* if it has no induced subgraph isomorphic to C_k for $k \geq 5$, to the house graph ($\overline{P_5}$), or to the graphs A and D_6 of Figure 1. The class was introduced by Olariu [25] who also presented an $O(n^3)$-time recognition algorithm. Since any C_k, where $k \geq 7$, contains a P_6, the class of weak bipolarizable graphs is a superclass of the bipolarizable graphs, as the name suggests after all (see also [25]).

As mentioned above, the recognition problem on both bipolarizable and P_4-simplicial graphs has been addressed by Hoàng and Reed [16]. Their algorithms are based on detecting whether the input graph G admits a bipolarizable or P_4-simplicial ordering. More precisely, the algorithm for recognizing a bipolarizable graph G constructs an acyclic bipolarizable orientation; for every P_4 $abcd$ in G, it orients ab (resp. cd) towards a (resp. d); if no edge of G receives opposite orientations and the resulting oriented graph is acyclic, it returns "yes", otherwise returns "no". The algorithm runs in $O(n^4)$ time, where n is the number of vertices of the input graph G. Their algorithm for recognizing P_4-simplicial graphs constructs a P_4-simplicial ordering of the vertices of G as follows: it initially sets $H := V(G)$ and then repeatedly chooses a vertex $x \in H$ such that x is not a midpoint of any P_4 $abxc$ (of the graph G) with $b, c \in H$ and sets $H := H - \{x\}$ and $y \prec x$ for all $y \in H$; if $H \neq \emptyset$ and the algorithm fails to choose a vertex $x \in H$, it returns "no". This algorithm takes $O(n^5)$ time.

Recently, Eschen *et al.* [7] described recognition algorithms for several classes of perfectly orderable graphs, e.g., $O(n^{3.376})$-time algorithms for both bipolarizable and P_4-simplicial graphs. In particular, they presented an algorithm for recognizing brittle graphs by direct application of the definition; a graph is *brittle* if and only if its vertices admit a linear order $(v_1, v_2, \ldots, v_n)$ such that each vertex v_i is either not a midpoint or not an endpoint of a P_4 in the subgraph of G induced by $\{v_i, \ldots, v_n\}$ [2; 14]. Their algorithm uses matrix multiplication and runs in $O(n^{3.376})$ time; it computes, for each vertex v_i, the P_4s which contain v_i as a midpoint, and thus, is also used to recognize bipolarizable and P_4-simplicial graphs within the same time bounds.

Additionally, we note that Hoàng and Reed also presented algorithms which solve the recognition problem for P_4-comparability and P_4-indifference graphs which run in $O(n^4)$ and $O(n^6)$ time [16; 17]. Recent results on these problems include $O(nm)$-time [23] and $O(n+m)$-time algorithms [10; 26], respectively, where m is the number of edges of the input graph.

Our objective is to study the recognition problem on the classes of bipolarizable and P_4-simplicial graphs and we present $O(nm)$-time algorithms for each of these problems. Our algorithms rely on properties of these graphs which we establish and which allow us to only work with P_3s of the input graph G which participate in P_4s of G; such P_3s can be computed in $O(nm)$ time by means of the distance trees of the *complement* of the graph rooted at each of its vertices [23]. The proposed recognition algorithms are simple, use simple data structures and both require $O(n + m)$ space. Additionally, we describe how to augment our two recognition algorithms so that they return a certificate whenever they decide that G is not bipolarizable or P_4-simplicial, thus, providing the most natural evidence that the input graph G indeed is not bipolarizable or P_4-simplicial. In particular, for the case of bipolarizable graphs, the augmented algorithm returns a forbidden subgraph contained in G. The augmented algorithms take $O(n + m)$ additional time and $O(n + m)$ space. Finally, we show that the class of weak bipolarizable graphs is a subclass of the class of P_4-simplicial graphs and give class inclusion results and the currently best time complexities for the recognition problem for a number of perfectly orderable classes of graphs.

The paper is structured as follows. In Section 2 we review the terminology that we use throughout the paper and we establish the theoretical framework on which our algorithms are based. The recognition algorithms for bipolarizable and P_4-simplicial graphs are described and analyzed in Sections 3 and 4, respectively. Section 5 gives results on class inclusions for a number of perfectly orderable classes, and Section 6 concludes with a summary of our results and some open problems.

2 Theoretical Framework

We consider finite undirected graphs with no loops or multiple edges. Let G be such a graph; then, $V(G)$ and $E(G)$ denote the set of vertices and of edges of G respectively. The *neighborhood* $N(x)$ of a vertex $x \in V(G)$ is the set of all the vertices of G which are adjacent to x. The *closed neighborhood* of x is defined as $N[x] := \{x\} \cup N(x)$.

The subgraph of G induced by a subset S of G's vertices is denoted by $G[S]$. A subset $A \subseteq V(G)$ of p vertices is a *p-clique*, or *clique*, if it induces a complete subgraph, i.e., $G[A] = K_p$; a single vertex is a 1-clique. An *independent set* is a subset $B \subseteq V(G)$ of vertices no two of which are adjacent; it is also called *stable set*. A subset $H \subseteq V(G)$ of vertices is *homogeneous* if $2 \leq |H| < |V(G)|$ and each vertex $x \in V(G) - H$ sees either all vertices or no vertex in H, i.e., either $H \subseteq N(x)$ or $H \cap N(x) = \emptyset$.

A *path* in a graph G is a sequence of vertices $v_0 v_1 \cdots v_k$ such that $v_{i-1}v_i \in E(G)$ for $i = 1, 2, \ldots, k$; we say that this is a path from v_0 to v_k and that its *length* is k. A path may be *undirected* or *directed* depending on whether G is an undirected or directed graph. A path is called *simple* if none of its vertices occurs more than once; it is called *trivial* if its length is equal to 0. A path (simple path) $v_0 v_1 \cdots v_k$ is called a *cycle* (*simple cycle*) of length $k + 1$ if $v_0 v_k \in E(G)$. An edge connecting two non-consecutive vertices in a simple path (cycle) is called a *chord*; then, a simple path (cycle) $v_0 v_1 \cdots v_k$ of a graph G is *chordless* if G contains no chords of the path (cycle), i.e., $v_i v_j \notin E(G)$ for any two non-consecutive vertices v_i, v_j in the path (cycle). The chordless path (chordless cycle, respectively) on n vertices is commonly denoted by P_n (C_n, respectively). In particular, a chordless path on 4 vertices is denoted by P_4.

Let $abcd$ be an induced P_4 of a graph. 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 contain some endpoint) are called the *wings*. For the P_4 $abcd$, the edge bc is its rib and the edges ab and cd are its wings.

Our bipolarizable graph recognition algorithm relies on the result stated in the following lemma.

Lemma 2.1 *Let G be a graph that contains no induced subgraph isomorphic to a house graph or the graphs A and D_6 of Figure 1. Then, G does not contain a C_4 $abcd$ such that abc and bcd are P_3s participating in P_4s of G.*

Proof: Suppose for contradiction that G contains a C_4 $abcd$ meeting the conditions in the statement of the lemma. We distinguish cases. Suppose first that the P_3 abc participates in the P_4 $abcx$ and that the P_3 bcd participates in the P_4 $bcdy$. Then, $xd \notin E(G)$, otherwise the vertices a, b, c, d, x would induce a house in G. In a similar fashion, $ya \notin E(G)$ either. But then, if $xy \notin E(G)$, then the subgraph induced by a, b, c, d, x, y is isomorphic to the A whereas if $xy \in E(G)$, it is isomorphic to the D_6; a contradiction in either case. The remaining three cases (depending on whether abc participates in a P_4 $xabc$ or $abcx$ and on whether bcd participates in a P_4 $ybcd$ or $bcdy$) are handled similarly. □

We note that Lemma 2.1, as well as the ensuing Lemma 3.1, in fact hold for the class of weak bipolarizable graphs, which is a superclass of the bipolarizable graphs. Lemma 2.1 however holds neither for the class of P_4-simplicial graphs nor for the class of HHD-free graphs [2; 18; 24], since the former class contains the house graph whereas the latter contains the graph A.

Computing all the P_3s participating in P_4s of a graph G: In [23], it has been shown that all the P_3s participating in P_4s of a graph G can be efficiently computed as follows:

Lemma 2.2 *For each vertex v of a graph G, let $T_{\overline{G}}(v)$ be the distance tree of the complement of G rooted at v (0-th level) and let $S_1, S_2, \ldots, S_{k_v}$ be a partition of the vertices in the 2nd level of the tree, where two vertices belong to the same S_i iff they have the same neighbors in the 1st level of $T_{\overline{G}}(v)$. Then, avb is a P_3 participating in a P_4 of G iff $ab \notin E(G)$ and either exactly one of a, b belongs to the 2nd level and the other to the 3rd level of $T_{\overline{G}}(v)$, or both a and b belong to the 2nd level but they are in different sets of the partition $S_1, S_2, \ldots, S_{k_v}$.*

Lemma 2.2 implies that for a graph G on n vertices and m edges, the P_3s participating in P_4s of G can be computed in $O(nm)$ time and $O(n+m)$ space because the number of vertices in all the levels, but the 0th and the 1st, of the distance tree $T_{\overline{G}}(v)$ does not exceed the degree of v in G; thus, considering pairs of vertices located in these levels takes $O(\sum_v deg^2(v)) = O(nm)$ time, where $deg(v)$ denotes the degree of v in the graph G [23].

Since the vertices in the 2nd and 3rd level of $T_{\overline{G}}(v)$ form a subset of the neighborhood of v, we can give a more unified criterion for deciding whether a P_3 avb participates in a P_4 of G by considering the following partition of $N(v)$:

- ▷ the partition of the vertices in the 2nd level of $T_{\overline{G}}(v)$ into $S_1, \ldots, S_{k_v}$ as described above;
- ▷ all the vertices in the 3rd level of $T_{\overline{G}}(v)$ are placed in a set S_{k_v+1};
- ▷ all remaining vertices in $N(v)$ are placed in a set S_0 (no such vertex a forms a P_3 avb participating in P_4s of G for any vertex b of G).

Then, Lemma 2.2 can be equivalently stated as follows:

Lemma 2.3 *For each vertex v of a graph G, let $T_{\overline{G}}(v)$ be as in Lemma 2.2 and let $S_0, S_1, \ldots, S_{k_v}, S_{k_v+1}$ be the partition described above. Then, for any $a, b \in N(v)$ such that $a \in S_i$ and $b \in S_j$, avb is a P_3 participating in a P_4 of G if and only if $ab \notin E(G)$, $i \neq 0$, $j \neq 0$, and $i \neq j$.*

3 Recognition of Bipolarizable Graphs

The definition of bipolarizable graphs implies that they can be efficiently recognized as soon as the wings of all the P_4s have been computed. The method described in [23] for computing all the P_3s participating in P_4s of a given graph does not seem to extend to produce within the same time complexity which edge of the P_3 is the rib and which is the wing of the P_4. However, in the case of bipolarizable graphs, we take advantage of Lemma 2.1 in order to achieve their efficient recognition. Indeed, since the bipolarizable graphs do not contain the house graph, A, or D_6 (see Figure 1), Lemma 2.1 implies the following result.

Lemma 3.1 *Let G be a bipolarizable graph and let abc be a P_3 participating in a P_4 of G. If bcd is another such P_3, then G contains the P_4 $abcd$.*

Proof: If the path $abcd$ is not a P_4 then G must contain the edge ad. But this creates a C_4 meeting the conditions of Lemma 2.1; a contradiction. □

Then, Lemma 3.1 implies the following corollary.

Corollary 3.1 *Let G be a bipolarizable graph and let F be the orientation of G such that the wings of each P_4 of G are oriented towards the endpoints of the P_4 and edges that are not the wings of any P_4 are not oriented. Then, for each edge bc of G for which there exist P_3s abc and bcd participating in P_4s of G, the edges ab and cd (for all such a and d) get oriented towards a and d respectively.*

Proof: Let us consider any such P_3 abc; then, because of the existence of the P_3 bcd, Lemma 3.1 applies, and thus $abcd$ is a P_4 of G. Therefore, the edge ab is oriented towards a in F, and this holds for all such a, and the edge cd is oriented towards d in F and this holds for all such d. □

The algorithm for the recognition of bipolarizable graphs applies Corollary 3.1. The input graph G is assumed to be given in adjacency list representation. The algorithm uses two arrays, an array $M[\,]$ and an array $S[\,]$, of size $2m$ each. The array $M[\,]$ has entries $M[xy]$ and $M[yx]$, for each edge xy of G; the entry $M[xy]$ is equal to 1 if there exist P_3s xyz participating in P_4s of G, and is equal to 0 otherwise. As a result, for an edge xy, both $M[xy]$ and $M[yx]$ are equal to 1 iff there exist P_3s xyz and txy participating in P_4s of G. The array $S[\,]$ too has entries $S[xy]$ and $S[yx]$, for each edge xy of G; the entry $S[xy]$ is equal to the index number of the partition set of $N(y)$ to which x belongs (see Lemma 2.3). As a result, a path xyz is a P_3 participating in P_4s of G iff $S[xy] \neq 0$, $S[zy] \neq 0$, and $S[xy] \neq S[zy]$. In more detail, the algorithm works as follows.

Bipolarizable Graph Recognition Algorithm
Input: an undirected graph G on n vertices and m edges.
Output: a message as to whether G is a bipolarizable graph or not.

1. Initialize the entries of the arrays $M[\,]$ and $S[\,]$ to 0; for each vertex v, sort the records of the neighbors of v in v's adjacency list in increasing vertex index number;
2. Find all the P_3s participating in P_4s of G; for each such P_3 abc, set the entries $M[ab]$ and $M[cb]$ equal to 1, and appropriately update the entries $S[ab]$ and $S[cb]$;
3. `for` each edge uv of G such that $M[uv] = 1$ and $M[vu] = 1$ `do`

(a) traverse the adjacency lists of u and v in lockstep fashion and process the neighbors of v which are not neighbors of u, and the neighbors of u which are not neighbors of v as follows:

(b) `for` each neighbor w of v in G which is not adjacent to u `do`
    `if` $S[uv] \neq 0$ `and` $S[wv] \neq 0$ `and` $S[uv] \neq S[wv]$
    `then` $\{uvw$ *is a* P_3 *in a* P_4 *of* $G\}$
        `if` the edge vw has not yet received an orientation
        `then` orient it towards w;
        `else if` it is oriented towards v
            `then` print that G is not a bipolarizable graph; `exit`;

(c) `for` each neighbor w of u in G which is not adjacent to v `do`
    `if` $S[vu] \neq 0$ `and` $S[wu] \neq 0$ `and` $S[vu] \neq S[wu]$
    `then` $\{vuw$ *is a* P_3 *in a* P_4 *of* $G\}$
        `if` the edge uw has not yet received an orientation
        `then` orient it towards w;
        `else if` it is oriented towards u
            `then` print that G is not a bipolarizable graph; `exit`;

4. Check if the directed subgraph $\overrightarrow{G}$ spanned by the oriented edges contains a directed cycle; if it does not, print that G is a bipolarizable graph; otherwise, print that it is not.

The correctness of the algorithm follows directly from Corollary 3.1. Observe that for any P_4 $abcd$ of G, the edge bc will be considered in Step 3 of the algorithm, and then the edges ab and cd will be assigned the desired orientations.

Time and Space Complexity. Step 1 takes $O(n+m)$ time since the sorted adjacency lists can be obtained through radix sorting an array of all the ordered pairs of adjacent vertices, while Step 2 takes $O(nm)$ time [23]. Steps 3b and 3c take constant time per such vertex w; it is assumed that the orientation of an edge is stored in an array of size m for constant-time access and update. For an edge uv, Steps 3b and 3c may be executed as many as $\Theta(deg(u) + deg(v))$ times, where $deg(u)$ denotes the degree of vertex u in G. Since Step 3a also takes $O(deg(u) + deg(v))$ time, Step 3 takes $O(nm)$ time because

$$\sum_{uv \in E(G)} (deg(u)+deg(v)) = \sum_{uv \in E(G)} deg(u) + \sum_{uv \in E(G)} deg(v) = \sum_{u \in V(G)} deg(u) \cdot deg(u) = O(nm).$$

Step 4 can be executed by constructing the directed graph $\overrightarrow{G}$ and then by applying topological sorting on it; if the topological sorting succeeds then no directed cycle exists, otherwise there exists a directed cycle. From this description, it is clear that Step 4 can be completed in $O(n+m)$ time and space. Since the computation of the P_3s participating in P_4s takes linear space, the total space needed by the recognition algorithm is clearly linear in the size of the input graph G.

Summarizing, we obtain the following theorem.

Theorem 3.1 *Let G be an undirected graph on n vertices and m edges. Then, our algorithm determines whether G is a bipolarizable graph in $O(nm)$ time and $O(n+m)$ space.*

The recognition algorithm can be used to produce a bipolarizable ordering of the vertices of a bipolarizable graph G. The bipolarizable ordering coincides with the topological ordering of the vertices of the directed graph $\overrightarrow{G}$, possibly extended by an arbitrary ordering of any vertices of G which do not participate in $\overrightarrow{G}$.

3.1 Providing a Certificate

The bipolarizable graph recognition algorithm can be easily augmented so that it provides a certificate whenever it decides that the input graph G is not bipolarizable; in particular, the algorithm can be made to return a forbidden subgraph of G in such a case.

As indicated by the algorithm, there are two reasons due to which a given graph G can be found non-bipolarizable. First, *a conflict of orientation* may arise on an edge of G which ends up receiving opposite orientations by different P_4s sharing it. Second, if no conflict has arisen, there may be the case that the directed subgraph $\overrightarrow{G}$ spanned by the oriented edges contains a *directed cycle*. Relating these two cases to the forbidden subgraphs, it is not difficult to see that:

Lemma 3.2 *A conflict of orientation arises while orienting the edges of a graph G if and only if G contains a C_5, a house graph, a P_6, an A, a C_6, or a D_6 (see Figure 1).*

Proof: It is easy to see that if the graph G contains any of the above graphs then a conflict of orientation arises; the bottom horizontal edge of the C_5 and the house, and the top horizontal edge of P_6, A, C_6, D_6 in Figure 1 receive opposite orientations.

Suppose now that an edge ab receives opposite orientations. Then, G contains P_4s $abcd$ and $xyab$; clearly y differs from c and d, x differ from both c and y. If d coincides with x, then the vertices a, b, c, d, y induce a C_5 or a house in G. If d differs from x, then if d, y or c, x are adjacent in G a C_5 or a house is induced, otherwise G contains an induced P_6, A, C_6, or D_6. □

In the following, we will consider graphs that contain no C_5, house, P_6, A, C_6, or D_6. Then, by Lemma 3.2, the wings of every P_4 of G can all be oriented without conflict towards the endpoints of the P_4. The orientation process produces directed and undirected edges; we say that an edge xy is *not* oriented towards x, if it is either undirected or directed towards y. The following lemma will be useful later.

Lemma 3.3 *Let G be a graph which contains no C_5, house, P_6, A, C_6, or D_6, and whose edges have been oriented as described above. Suppose further that G contains a cycle $bcfg$ with a single diagonal bf such that the edge fg is oriented towards f, the edge bg is not oriented towards b, the edge cf is not oriented towards f, and the diagonal bf is undirected. Then, if $ahgf$ is any P_4 of G, the vertices a, b, c, f, g, h induce in G a subgraph as the one shown on the right in Figure 2 and the edge cf is oriented towards c.*

Proof: The cycle $bcfg$ is shown on the left in Figure 2; white arrows indicate potential orientations. It is easy to see that vertex b is adjacent to both a and h in G: if b was adjacent neither to a nor to h, the path $ahgb$ would be a P_4 with its wing bg oriented away from the endpoint b, in contradiction to the potential orientation of bg; if b was adjacent to a but not to h, the vertices a, b, f, g, h would induce a house in G; if b was adjacent to h but not to a, the path $ahbf$ would be a P_4 and its wing bf would have received an orientation. Additionally, $ac \notin E(G)$: if $ac \in E(G)$, the vertices a, c, f, g, h would induce a house or a C_5 in G depending on whether c, h are adjacent or not. Then, $ch \notin E(G)$ as well,

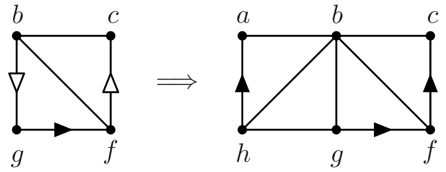


Fig. 2

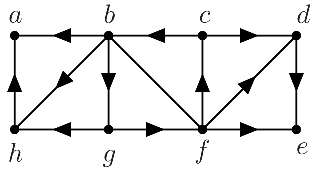


Fig. 3

for otherwise the path $ahcf$ would be a P_4 with its wing cf oriented towards f, in contradiction to the potential orientation of cf. The above adjacencies imply that the path $cfgh$ is a P_4 and thus the edges cf and gh are oriented towards c and h respectively. □

Using Lemma 3.3 we prove the following lemma which is important for locating z-triominoes.

Lemma 3.4 *Let G be a graph which contains no C_5, house, P_6, A, C_6, or D_6. Then, if G contains a z-triomino then the directed graph $\overrightarrow{G}$ exhibits a directed cycle on 4 vertices with a single undirected diagonal. Conversely, if $\overrightarrow{G}$ exhibits a directed cycle $bgfc$ (directed from b to g to f to c to b) with a single undirected diagonal bf, then for any P_4s $ahgf$ and $bcde$ of G, the vertices a, b, c, d, e, f, g, h induce a z-triomino in G.*

Proof: ($\Longrightarrow$) Suppose that G contains a z-triomino. Since G contains no C_5, house, P_6, A, C_6, or D_6, each edge of G will receive at most one orientation (Lemma 3.2). Thus, the edges of each z-triomino of G receive the orientations indicated in Figure 3; that is, for the z-triomino with vertices a, b, c, d, e, f, g, h, a single directed cycle $bcfg$ is formed. It suffices to show that the diagonal bf is not the wing of any P_4 of G and thus receives no orientation. Clearly, it does not get oriented due to a P_4 induced by a subset of the vertices of the z-triomino. Due to symmetry, considering P_4s with vertices in addition to the vertices of the triomino is exhausted to the following three cases (see Figure 3):

(a) *the edge bf participates in a P_4 $xabf$:* Then, $xc \in E(G)$, otherwise the path $xabc$ would be a P_4 whose wing bc would be oriented towards b in $\overrightarrow{G}$. But then, $fcxa$ would be a P_4 with both its wings oriented away from its endpoints; a contradiction.

(b) *the edge bf participates in a P_4 $xhbf$:* Then, again $xc \in E(G)$, otherwise the path $xhbc$ would be a P_4 whose wing bc would be oriented towards b in $\overrightarrow{G}$. And again, $fcxh$ would be a P_4 with both its wings oriented away from its endpoints; a contradiction.

(c) *the edge bf participates in a P_4 $xybf$:* Then, at least one of xc and yc belongs to $E(G)$, otherwise the path $xybc$ would be a P_4 whose wing bc would be oriented towards b. In fact, $yc \in E(G)$, for otherwise the existence of xc would imply that the vertices x, y, b, c, f induce a house graph in G. This in turn implies that $xc \in E(G)$, otherwise the path $xycf$ would be a P_4 whose wing fc would be oriented towards c. The existence of xc implies the existence of xg and the non-existence of xh since otherwise the paths $gfcx$ and $fcxh$ would be "badly" oriented P_4s respectively. But then, the subgraph induced by the vertices b, c, x, g, h is a house; a contradiction.

A contradiction has been obtained in each case, which implies that the edge bf is not the wing of any P_4 of the graph G and thus it does not receive an orientation.

($\Longleftarrow$) Now suppose that the directed graph $\overrightarrow{G}$ exhibits a directed cycle $bgfc$ with a single undirected diagonal bf. Lemma 3.3 applies for this cycle in terms of the edge fg and any P_4 $ahgf$ of G. Lemma 3.3 also applies for the same cycle in terms of the edge bc and any P_4 $bcde$ of G; it is important to observe that the vertices a, h differ from d, e as they have different neighbors among b, c, f, g. As a result, the vertices a, b, c, d, e, f, g, h induce a subgraph containing the edges of the z-triomino shown in Figure 3 and potentially edges connecting a, h to d, e. In fact, d and h are not adjacent, otherwise the path $cdhg$ would be a P_4 whose wing cd would be oriented away from the endpoint c. This implies that neither e, h nor a, d are adjacent, otherwise, there would exist P_4s $cdeh$ and $dahg$, respectively, with their wings cd and gh oriented away from the endpoint. Finally, a, e are not adjacent either; if they were adjacent, the path $haed$ would be a P_4 whose wing ah would be oriented away from the endpoint h. The above adjacencies imply that the vertices a, b, c, d, e, f, g, h induce a z-triomino in G. □

Additionally, we give below a result of Hertz ([12], Lemma 2.4) which we have paraphrased using our formalism and terminology:

Lemma 3.5 *Let $C = v_{k-1}v_{k-2}\cdots v_1v_0$ be a directed cycle on $k \geq 3$ vertices without any directed chord. Let us consider the directed path $v_{n-1}v_{n-2}\cdots v_0$ on $n \leq k$ consecutive vertices of C. Let r_i, s_i $(1 \leq i \leq n-1)$ be any vertices such that the paths $s_ir_iv_iv_{i-1}$ are P_4s of G. Then, the vertices $v_0, v_1, \ldots, v_{n-1}$, $r_1, r_2, \ldots, r_{n-1}$, and $s_1, s_2, \ldots, s_{n-1}$ induce a quasi n-wheel in G.*

(We note that in Lemma 3.5 it is implied that no conflict of orientation has arisen and that the graph G contains all chords connecting the k vertices of the directed cycle C.) A quasi k-wheel is what remains from a k-wheel with vertex set $\{v_0, v_1, \ldots, v_{n-1},\ r_0, r_1, \ldots, r_{n-1},\ s_0, s_1, \ldots, s_{n-1}\}$ when vertices r_0 and s_0 are removed. Hertz used Lemma 3.5 to establish his main theorem; in the proof, he shows the following:

Lemma 3.6 *Let $C = v_{n-1}v_{n-2}\cdots v_1v_0$ be a directed cycle on $n \geq 3$ vertices without any directed chord. Let r_i, s_i $(0 \leq i \leq n-1)$ be any vertices such that the paths $s_ir_iv_iv_{i-1}$ are P_4s of G. Then, the vertices $v_0, v_1, \ldots, v_{n-1}$, $r_0, r_1, \ldots, r_{n-1}$, and $s_0, s_1, \ldots, s_{n-1}$ induce an n-wheel in G.*

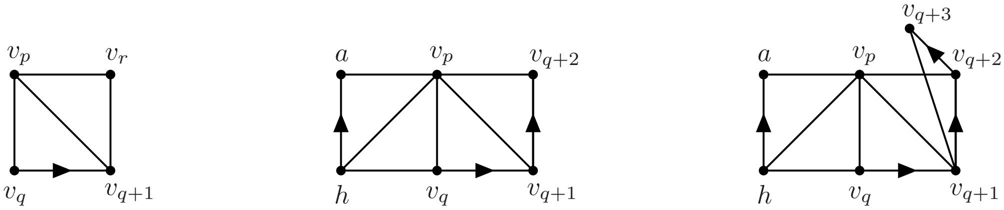
Finally, we can also show the following results.

Lemma 3.7 *Let G be a graph which contains no C_5, house, P_6, A, C_6, or D_6, and whose edges have been oriented as described earlier yielding the directed graph $\overrightarrow{G}$. Then:*

(i) G does not contain a C_4 $bcfg$ such that the edge fg is oriented towards f, the edge bc is not oriented towards c, and the edge cf is not oriented towards f.

(ii) If $\overrightarrow{G}$ contains a directed C_k where $k \geq 5$ (i.e., there are no directed chords), the vertices of the cycle induce a complete graph in G.

Proof: (i) Suppose for contradiction that the graph G contains such a C_4. The orientation of the edge fg implies that G contains an induced P_4, say, $ahgf$, with wing fg. Then, as in the proof of Lemma 3.3, we can show that c is adjacent neither to a nor to h. If $ac \in E(G)$, then G would contain a house or a C_5 depending on whether c, h are adjacent or not. Since a, c are not adjacent then c, h are not adjacent either, otherwise the path $ahcf$ would be a P_4 whose wing cf would be oriented towards f, in contradiction to the potential orientation of cf given in the statement of the lemma. Finally, $bh \in E(G)$ otherwise



(a) (b) (c)

Fig. 4

the path $cbgh$ would be a P_4 whose wing cb would be oriented towards c, again in contradiction to the potential orientation of bc. But then, the vertices b, c, f, g, h induce a house in G; a contradiction.
(ii) Suppose for contradiction that the directed graph $\overrightarrow{G}$ contains a directed C_k on $k \geq 5$ vertices and two of them are not adjacent in G; without loss of generality, we assume that the C_k is $v_0 v_1 \cdots v_{k-1}$ (oriented from v_0 to v_1 and so on to v_{k-1} and back to v_0) and that $v_0 v_t \notin E(G)$ where $1 < t < k-2$. Let

$$q = \max_{0 \leq i < t} \{\, i \mid v_i v_t \notin E(G) \text{ and } v_{i+1} v_t \in E(G)\};$$

we note that q is well defined since $v_0 v_t \notin E(G)$ and $v_{t-1} v_t \in E(G)$.

Next, we show that there exist vertices v_p, v_r such that the vertices v_p, v_q, v_{q+1}, v_r appear in that order around the directed C_k of the graph $\overrightarrow{G}$ and induce in G the cycle $v_p v_q v_{q+1} v_r$ where $v_p v_{q+1} \in E(G)$ and $v_q v_r \notin E(G)$; see Figure 4(a). We consider a chordless path ρ connecting v_t to v_q in the subgraph $G[\{v_t, v_{t+1}, \ldots, v_{k-1}, v_0, v_1, \ldots, v_q\}]$; let us suppose that $\rho = v_t v_{i_1} v_{i_2} \cdots v_{i_\ell} v_q$, where $\ell \geq 1$ since $v_q v_t \notin E(G)$. If $\ell = 1$ then we have the cycle $v_q v_{q+1} v_t v_{i_1}$ where $v_q v_t \notin E(G)$. If $v_{i_1} v_{q+1} \notin E(G)$ then the cycle $v_q v_{q+1} v_t v_{i_1}$ meets the conditions of statement (i) of this lemma for $b = v_{i_1}$, $c = v_t$, $f = v_{q+1}$, and $g = v_q$ (note that the edges $v_{q+1} v_t$, $v_t v_{i_1}$, and $v_{i_1} v_q$ are either chords or they are directed towards v_t, v_{i_1}, and v_q, respectively); since this is impossible according to statement (i), we conclude that $v_{i_1} v_{q+1} \in E(G)$. Then, we have the desired cycle for $v_p = v_{i_1}$ and $v_r = v_t$. Suppose now that $\ell \geq 2$. Let us consider that $v_{i_0} = v_t$ and let $s = \max_{0 \leq j \leq \ell} \{\, j \mid v_{i_j} v_{q+1} \in E(G)\,\}$; s is well defined since the edge $v_{i_0} v_{q+1} = v_t v_{q+1}$ belongs to $E(G)$. Since G contains no C_5, C_6, or P_6, and effectively no C_i for $i \geq 5$, it follows that $s \geq \ell - 1$. But $s \neq \ell - 1$, otherwise the cycle $v_q v_{q+1} v_{i_{\ell-1}} v_{i_\ell}$ would meet the conditions of statement (i) of this lemma for $b = v_{i_\ell}$, $c = v_{i_{\ell-1}}$, $f = v_{q+1}$, and $g = v_q$, and that would be impossible; recall that the path ρ is chordless. Thus, $s = i_\ell$. Then, $v_{q+1} v_{i_{\ell-1}} \in E(G)$; if not, then depending on whether $v_{q+1} v_{i_{\ell-2}} \in E(G)$ or not, G would contain either a house or a C_i for $i \geq 5$, respectively. Therefore, we have the desired cycle for $v_p = v_{i_\ell}$ and $v_r = v_{i_{\ell-1}}$.

The existence of a cycle $v_p v_q v_{q+1} v_r$ such that $v_p v_{q+1} \in E(G)$ and $v_q v_r \notin E(G)$ implies that we can find v_p, v_r such that $v_p \in \{v_{r+1}, v_{r+2}, \ldots, v_{k-1}, v_0, \ldots, v_{q-1}\}$ and the length of the path $v_r v_{r+1} \cdots v_p$ is minimized. The properties of the cycle $v_p v_q v_{q+1} v_r$ imply that Lemma 3.3 applies on it for $b = v_p$, $c = v_r$, $f = v_{q+1}$, and $g = v_q$ (Figure 4(a)); in turn, the lemma implies

(i) that for any P_4 $ahv_q v_{q+1}$ of G, the vertices $a, v_p, v_r, v_{q+1}, v_q, h$ induce a subgraph isomorphic to the one shown on the right in Figure 2, and

(ii) that the edge $v_{q+1} v_r$ is oriented towards v_r.

The latter fact implies that $v_r = v_{q+2}$, since among all the edges of G connecting vertices of the cycle $v_0 v_1 \cdots v_k$ only the edges of the cycle are directed (Figure 4(b)). We distinguish the following cases:

a. $v_p = v_{q+3}$: In this case, the edge $v_{q+2} v_p$ is directed towards v_p and then Lemma 3.3 applies on the cycle $v_p v_q v_{q+1} v_{q+2}$ for $b = v_{q+1}$, $c = v_q$, $f = v_p$, and $g = v_r$, implying that the edge $v_p v_q$ is oriented towards v_q; however, this contradicts the fact that the cycle $v_0 v_1 \cdots v_{k-1}$ is of length $k \geq 5$ with no directed chords.

b. $v_{q+1} v_{q+3} \notin E(G)$: Then, the orientations of the edges $v_q v_{q+1}$, $v_{q+1} v_{q+2}$, and $v_{q+2} v_{q+3}$ imply that $v_q v_{q+3} \in E(G)$, i.e., the vertices $v_q, v_{q+1}, v_{q+2}, v_{q+3}$ induce a C_4 in G; this leads to a contradiction as well since Lemma 3.7 (statement (i)) applies on this C_4.

c. $v_p \neq v_{q+3}$ *and* $v_{q+1} v_{q+3} \in E(G)$: Then, $v_q v_{q+3} \notin E(G)$ for otherwise G would contain the cycle $v_q v_{q+1} v_{q+2} v_{q+3}$, contradicting the minimality of the choice of v_p, v_r. Since $v_q v_{q+3} \notin E(G)$, then for the same reason, $v_p v_{q+3} \notin E(G)$; otherwise, G would contain the cycle $v_p v_q v_{q+1} v_{q+3}$. Thus, the situation is as shown in Figure 4(c) where v_{q+3} may also be adjacent to a, h in G. Since the chord $v_{q+1} v_{q+3}$ is undirected, the path $a v_p v_{q+1} v_{q+3}$ is not a P_4, and thus $a v_{q+3} \in E(G)$. But then, the vertices $a, h, v_q, v_{q+1}, v_{q+3}$ induce either a house or a C_5 depending on whether h, v_{q+3} are adjacent in G or not, a contradiction.

We reached a contradiction in each case, and thus the directed graph $\overrightarrow{G}$ cannot contain a directed C_k, where $k \geq 5$, such that the vertices of the cycle do not induce a complete graph in G. $\square$

The above lemmata suggest the following additions to our bipolarizable graph recognition algorithm so that a forbidden subgraph is returned whenever the input graph G is deemed non-bipolarizable:

A. *a conflict of orientation is found on an edge vw (Steps 3.2, 3.3)*:

A.1 locate P_4s $abvw$ and $vwxy$ of G;

A.2 `if` $a = y$
`then` return as forbidden subgraph the subgraph $G[\{a, b, v, w, x\}]$;
`else` return as forbidden subgraph the subgraph $G[\{a, b, v, w, x, y\}]$;

B. *a directed cycle is detected in the directed graph $\overrightarrow{G}$ (Step 4)*:

B.1 locate a directed cycle with no directed chords;

B.2 `if` this is a directed cycle $bgfc$ with a single undirected diagonal bf
`then` locate P_4s $ahgf$ and $bcde$ of G;
return as forbidden subgraph the subgraph $G[\{a, b, c, d, e, f, g, h\}]$;
`else` $\{$*the vertices of the cycle induce a complete subgraph of* $G\}$
return the k-wheel built around the directed cycle, where k is the length of the cycle;

The correctness of Steps A.1 and A.2 follows from Lemma 3.2 and its proof, while the correctness of Steps B.1 and B.2 follows from Lemmata 3.4, 3.6, and 3.7. In the following, we describe in detail, the components required to efficiently carry out the above computations.

3.1.1 Locating a P_4 $abcd$ when given an edge ab.

We assume that with each edge of G, we store pointers to the vertex records of its endpoints; thus, from the edge ab, we can access a and b in constant time. Then, we find the sought P_4 $abcd$ as follows:

1. store the neighbors of the vertices a and b in an array each for constant time adjacency tests;
2. `for` each edge uw of G `do`
   `if` u is adjacent to b but not to a and w is adjacent neither to a nor to b
   `then` the sought P_4 is the path $abuw$; `return`;
   `if` u is adjacent neither to a nor to b and w is adjacent to b but not to a
   `then` the sought P_4 is the path $abwu$; `return`;

The correctness of the computation is a direct consequence of the adjacencies of the vertices a, b, u, w.

3.1.2 Locating a directed cycle without directed chords.

To locate a directed cycle in the directed subgraph $\overrightarrow{G}$ spanned by the oriented edges of G, we apply depth-first search on $\overrightarrow{G}$ and obtain an ordered list L of the vertices of such a cycle. Then, in order to isolate a directed cycle with no directed chords, we use the following procedure *get_c-d-cycle*(L) on L. The procedure clips portions of the list L until it obtains a directed cycle without any directed chords, as desired. We assume that each record in the list is initially associated with its rank in the list L. These integers will not be updated, and, due to clipping, they will not match the current ranks; nevertheless, they will reflect the order of the vertex records along the directed cycle stored in L at any given time.

get_c-d-cycle(L)
`for` each vertex x in order in the list L `do`
  find the vertex y, if any, such that the edge xy is directed towards x, and y exhibits the smallest "rank" larger than x's;
  `if` no such vertex y exists
  `then` find the vertex z, if any, such that the edge xz is directed towards z, and z exhibits the largest "rank" larger than x's;
    `if` such a vertex z exists
    `then` clip the list L by removing any vertex records between x and z;
  `else` find the vertex z, if any, such that the edge xz is directed towards z, and z exhibits the largest "rank" larger than x's and smaller than y's;
    `if` no such vertex z exists
    `then` clip the list L by removing any vertex records to the left of x and to the right of y;
    `else` clip the list L by maintaining the vertex record for x followed by those between z and y inclusive;

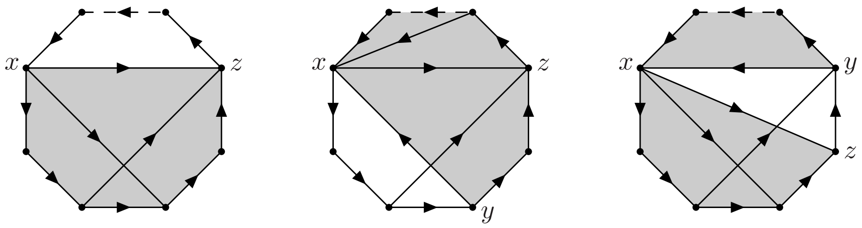


Fig. 5

The vertices that remain in the list L after the procedure *get_c-d-cycle*(L) has completed induce a directed cycle without directed chords in the directed subgraph $\overrightarrow{G}$. The correctness of the computation follows easily: the three cases of clipping are illustrated in Figure 5, where the dashed edge at the top indicates the edge directed from the last vertex of the list L to its first vertex and the shaded regions are the portions of the list L that are removed during the processing of vertex x in each case. It is important to observe that the clipping indeed maintains the ordering of the "ranks" of the vertices along the list L, and that after a vertex x has been processed the resulting list L induces a directed cycle without directed chords incident on x.

3.1.3 Locating a k-wheel.

If the directed cycle in the list L that the procedure *get_c-d-cycle*( ) has located is not a cycle on 4 vertices with a single diagonal but instead is a C_k $(k \geq 3)$ induced by the vertex set $P = \{v_0, v_1, \ldots, v_{k-1}\}$ in order along the C_k, we can obtain the complete k-wheel built around the C_k. If we try to find a P_4 $s_i r_i v_i v_{i-1}$ for each pair v_i, v_{i-1}, this might require $\Omega(k(n+m))$ time, which may be superlinear in $\max\{n, m\}$. Instead, we can obtain a linear time complexity by using an array $A[\,]$ of size n and by working as follows:

1. mark the entries of $A[\,]$ corresponding to the vertices $v_0, v_1, \ldots, v_{k-1}$ of the set P;
   form empty sets $R_0, R_1, \ldots, R_{k-1}$ and $S_0, S_1, \ldots, S_{k-1}$;

2. `for` each vertex $x \notin P$ `do`
   - $p_x \leftarrow$ number of vertices in P which are neighbors of x in G;
   - `if` $p_x = k - 1$
   - `then if` the only vertex in P which is not a neighbor of x in G is $v_{i-1 \bmod k}$
     - `then` insert x in the set R_i;
   - `if` $p_x = k - 2$
   - `then if` the only vertices in P which are not neighbors of x in G are $v_i, v_{i-1 \bmod k}$
     - `then` insert x in the set S_i;

3. let $B[\,]$ be an empty array of size n;
   `for` $i = 0, 1, \ldots, k-1$ `do`
   - mark the entries of $B[\,]$ corresponding to the vertices in the set S_i;
   - `for` each vertex $w \in R_i$ `do`
     - `for` each neighbor z of w in G `do`
       - `if` $B[z]$ is marked $\{z \in S_i\}$
       - `then` $r_i \leftarrow w$; $s_i \leftarrow z$;
         - unmark the entries of $B[\,]$ corresponding to the vertices in S_i;
         - proceed with next value of i;

4. the vertices in P and the collected vertices r_i and s_i induce a k-wheel in G.

It is important to observe that the vertex sets $R_0, R_1, \ldots, R_{k-1}$ and $S_0, S_1, \ldots, S_{k-1}$ are distinct from one another and from P due to their adjacencies to the vertices in P; thus, storing all these sets in linked lists takes $O(n)$ space. Moreover, for each i, the vertices v_i, v_{i-1}, r_i, s_i induce a P_4 $s_i r_i v_i v_{i-1}$ in G; thus, by Lemma 3.6, the process indeed finds a wheel.

3.1.4 Time and Space Complexity.

In light of Section 3.1.1, Step A.1 clearly takes $O(n+m)$ time and space, while Step A.2 takes $O(1)$ time and space. The application of depth-first search in the directed graph $\overrightarrow{G}$ also takes $O(n+m)$ time yielding the list L. Linear time and space is required by the execution of the procedure *get_c-d-cycle*(L) as well: identifying the vertices y and z for each vertex x in L takes time proportional to the degree of x in G, assuming that each directed edge has been marked with its assigned orientation; then, clipping the list can be done in constant time if L is maintained as a doubly connected list. In total, Step B.1 takes $O(n+m)$ time and space.

The time taken by Step B.2 includes $O(1)$ time to test if the located directed cycle is a cycle on 4 vertices with a single undirected diagonal, and either $O(n+m)$ time to locate the P_4s $ahgf$ and $bcde$ or the time to locate the k-wheel. Steps 1and 4of the procedure to locate a k-wheel take $O(n)$ time, Step 2takes $O(n+m)$ time (the number p_x is computed by traversing the adjacency list of x and counting the number of x's neighbors in the set P), while Step 3takes $O\big(n+\sum_i \big(|S_i|+\sum_{w\in R_i} deg(w)\big)\big) = O\big(n+\sum_{x\in V(G)} deg(x)\big) = O(n+m)$ time since the sets R_i and S_i are disjoint; by $deg(x)$ we denote the degree of x in G. In addition to the adjacency list representation of the graph G, the space required by Step B.2 is $O(n)$.

Therefore, we have the following result:

Theorem 3.2 *Let G be an undirected graph on n vertices and m edges. The bipolarizable graph recognition algorithm presented in this section can be augmented to provide a forbidden subgraph in G, whenever it decides that G is not bipolarizable, in $O(n+m)$ time and $O(n+m)$ space.*

4 Recognition of P_4-simplicial Graphs

Our P_4-simplicial graph recognition algorithm relies on the corresponding algorithm of Hoàng and Reed [16]; our contribution is that we restate the main condition on which their algorithm is based in terms of P_3s participating in P_4s of the input graph, and we show how to efficiently take advantage of it in order to achieve an $O(nm)$-time complexity. As described in the introduction, their algorithm works as follows: it initially sets $H := V(G)$ and then it iteratively identifies a vertex x in H such that G does not contain a P_4 of the form $abxc$ with $b, c \in H$, and removes it from H; the graph G is P_4-simplicial iff the above process continues until H becomes the empty set.

It is not difficult to see that the necessary property for a vertex x to be removed from H can be equivalently stated as follows:

Property 4.1 *Let H be the current set of vertices of a given graph G. Then, a vertex x can be removed from H if and only if there does not exist any P_3 bxc participating in a P_4 of G with $b, c \in H$.*

In light of Property 4.1, we can obtain an algorithm for deciding whether a given graph G is P_4-simplicial by keeping count, for each vertex $v \in H$, of the number of P_3s bvc with $b, c \in H$ which participate in P_4s of G, and by removing a vertex x from H whenever the number of such P_3s associated with x is 0. The proposed algorithm implements this idea; it takes advantage of the computation of the P_3s in P_4s of G in $O(nm)$ time, and maintains an array $NumP3[\,]$ of size n, which stores for each vertex v in H the number of P_3s bvc which participate in P_4s of G and have $b, c \in H$. The input graph G is assumed to be given in adjacency list representation. In more detail, the algorithm works as follows.

P_4-simplicial Graph Recognition Algorithm

Input: an undirected graph G on n vertices and m edges.
Output: a message as to whether G is a P_4-simplicial graph or not.

1. Collect all the vertices of G into a set H;
   make a copy $A[v]$ of the adjacency list of each vertex v of G while attaching at each record of the list an additional field *set*;

2. `for` each vertex v of G `do`

   a) compute the partition of the vertices in $N(v)$ into sets $S_0, S_1, \ldots, S_{k_v}, S_{k_v+1}$ as described in Lemma 2.3, and update the fields *set* of the records in the adjacency list $A[v]$ of v;

   b) compute the number of P_3s avb participating in P_4s of G and assign this number to $NumP3[v]$;

3. Collect in a list L the vertices v for which $NumP3[v] = 0$;

4. `while` the list L is not empty `do`

   a) remove a vertex, say, x, from L;

   b) `for` each vertex u adjacent to x in G `do`

      `if` u belongs to H
      `then` traverse the adjacency list $A[u]$ of u and let s_x be the value of the field *set* for x;
         `if` $s_x \neq 0$
         `then` $\{$*there may exist* P_3*s* xuw *participating in* P_4*s of* $G\}$
            `for` each vertex w in the adjacency list $A[u]$ of u `do`
               $s_w \leftarrow$ value of the field *set* for the vertex w;
               `if` $w \in H$ `and` $s_w \neq 0$ `and` $s_w \neq s_x$
               `then` $\{xuw$ *is such a* P_3 *with* $x, u, w \in H\}$
                  $NumP3[u] \leftarrow NumP3[u] - 1$;
            `if` $NumP3[u] = 0$
            `then` insert u in the list L;

   c) remove x from the set H;

5. `if` the set H is empty
   `then` print that G is a P_4-simplicial graph;
   `else` print that G is not a P_4-simplicial graph;

To ensure correct execution, the algorithm maintains the following invariant throughout the execution of Step 4.

Invariant 4.1 *At the beginning of every iteration of the while loop in Step 4 of the algorithm, for each vertex v in H, $NumP3[v]$ is equal to the number of P_3s bvc participating in P_4s of G with $b, c \in H$.*

Proof: The proof proceeds inductively in the number of iterations of the while loop. Step 2 clearly implies that the invariant holds at the begin of the first iteration of the loop, since $H = V(G)$. Suppose now that the invariant holds at the beginning of the i-th iteration, where $i \geq 1$; we will show that it also holds at the beginning of the $(i+1)$-st iteration. Let H_i denote the current value of H at the beginning of the

i-th iteration and let x be the vertex removed from L during the i-th iteration; then, $H_{i+1} = H_i - \{x\}$. Clearly, the vertices v for which $NumP3[v]$ may be changed from the i-th to the $(i+1)$-st iteration are those in $H_{i+1} \cap N(x)$; indeed, for any vertex w in $H_{i+1} - N(x) = H_i - N(x) - \{x\}$, any P_3 awb with $a, b \in H_i$ will have $a, b, w \in H_{i+1}$, since $a, b \neq x$, and thus $NumP3[w]$ will remain unchanged. Now, let us consider a vertex $v \in H_{i+1} \cap N(x)$. At the beginning of the $(i+1)$-st iteration, $NumP3[v]$ must be equal to the number of P_3s avb participating in P_4s of G with $a, b \in H_{i+1}$; this is precisely the value of $NumP3[v]$ at the beginning of the i-th iteration minus the number of P_3s xvz participating in P_4s of G with $z \in H_i$. Step 4b identifies these P_3s and decrements $NumP3[v]$ by 1 for each one of them. Note that $NumP3[v]$ will be decremented exactly once for each P_3 avb: if we assume, without loss of generality, that a is removed from H before b, then $NumP3[v]$ will be decremented during the processing of a; when b is processed, the P_3 avb will not be taken into account, even if v still belongs to H, because $a \notin H$. □

Then, the correctness of the algorithm follows from the correctness of the algorithm of Hoàng and Reed, from Lemma 2.3, Property 4.1, and the fact that at any given time the list L contains precisely those vertices that can be removed from H (a vertex x is inserted in L if and only if $NumP3[x] = 0$, i.e., there does not exist any P_3 bxc participating in a P_4 of G with $b, c \in H$).

Time and Space Complexity. The set H can be implemented by means of an array $M[\,]$ of size n, where $M[v] = 1$ if $v \in H$ and 0 otherwise; in this way, insertion, deletion, and membership queries for any vertex of G can be answered in constant time, while the emptiness of H can be checked in $O(n)$ time. Then, Step 1 takes $O(n+m)$ time, Step 4c takes $O(1)$ time per vertex removed, and Step 5 $O(n)$ time. Step 2 takes $O(nm)$ time [23], while Step 3 takes $O(n)$ time. As a vertex is inserted at most once in the list L, the time complexity of Step 4 is $O\Big(\sum_x \big(1 + \sum_{u \in N(x)} \; deg(u)\big)\Big)$, where $deg(u)$ denotes the degree of u in G. Since $\sum_{u \in N(x)} \; deg(u) = O(m)$, the time complexity of Step 4 is $O(nm)$. The computation of the P_3s participating in P_4s takes linear space, and thus the total space needed by the recognition algorithm is clearly linear in the size of the input graph G.

Summarizing, we obtain the following theorem.

Theorem 4.1 *Let G be an undirected graph on n vertices and m edges. Then, our algorithm determines whether G is a P_4-simplicial graph in $O(nm)$ time and $O(n+m)$ space.*

4.1 Providing a Certificate

As in the case of the bipolarizable graph recognition algorithm, the above algorithm can be made to return a certificate whenever it decides that the input graph G is not P_4-simplicial. In particular, it could return the value of the set H, which would indicate a subgraph of G none of whose vertices can be removed in the sense of Property 4.1. Clearly, this does not require any additional computation time and space.

However, it would be more interesting if the algorithm located a minimal such subset H' of H, that is, a subset H' such that every vertex y of H' forms a P_3 xyz participating in a P_4 of the input graph with $x, z \in H'$. To see the benefits of this, consider for example that the input graph G contained two domino graphs sharing an edge. Since the domino graph is a forbidden subgraph for the class of P_4-simplicial graphs, the algorithm would stop, would report that G is not P_4-simplicial and would return a set H of vertices which would be a superset of the set of vertices of both domino graphs. If however a minimal such set of vertices were returned, then one would be very close to identifying a forbidden

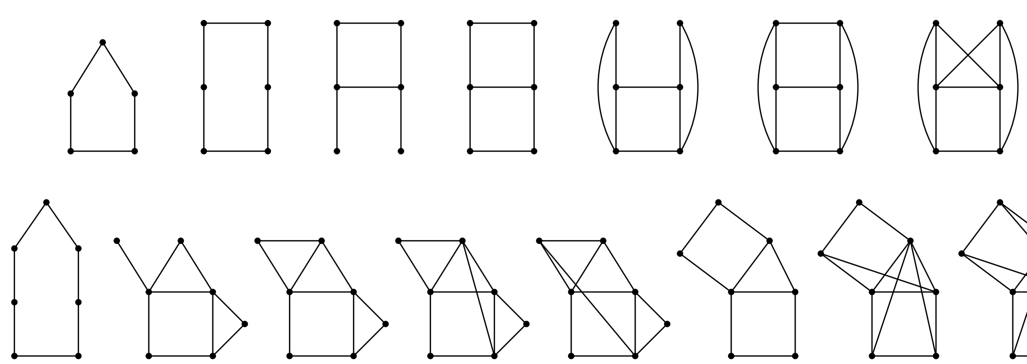


Fig. 6: Some forbidden subgraphs for the class of P_4-simplicial graphs.

subgraph in G. This approach, although very interesting, is hindered in part by the fact that no complete characterization of the P_4-simplicial graphs by forbidden subgraphs is currently available in the literature. As a first attempt towards obtaining such a characterization, we found in an exhaustive fashion that all the forbidden subgraphs for the class of P_4-simplicial graphs on up to 7 vertices are those shown in Figure 6.

5 Class Inclusions and Recognition Time Complexities

Figure 7 shows a diagram of class inclusions for a number of perfectly orderable classes of graphs and the currently best time complexities to recognize members of these classes. For definitions of the classes shown, see [2; 8]; note that the P_4-free and the chordal graphs are also known as co-graphs and triangulated graphs respectively. In the diagram, there exists an arc from a class $\mathcal{A}$ to a class $\mathcal{B}$ if and only if $\mathcal{B}$ is a proper subset of $\mathcal{A}$. Hence, if two classes are not connected by an arc, then each of these classes contains graphs not belonging to the other class (there are such sample graphs for each pair of non-linked classes).

Most of these class inclusions can be found in [2] where a similar diagram with many more graph classes appears; Figure 7 comes from a portion of the diagram in [2] augmented with the introduction of the inclusion relations for the classes of P_4-simplicial, bipolarizable, and P_4-indifference graphs, as described in Lemmata 5.1-5.3. We will show next that the class of weak bipolarizable graphs [25] is a proper subset of the class of P_4-simplicial graphs. In fact, we show a slightly stronger result as established in the following proposition.

Proposition 5.1 *Let G be a weak bipolarizable graph and let v be a vertex of G. Then, G admits a P_4-simplicial order $\prec$ on its vertices such that $v \prec x$ for any vertex x of G other than v.*

Proof: We apply induction on the size of the graph by taking advantage of Theorem 1 of [25] which states that a graph G is weak bipolarizable if and only if every induced subgraph of G is chordal or contains a homogeneous set. For the basis step, it is not difficult to verify that every weak bipolarizable graph on up to 3 vertices admits a P_4-simplicial order as described, since G does not contain any P_4s. Next, we assume that the proposition holds for all weak bipolarizable graphs on up to $k \geq 3$ vertices; we will show that any weak bipolarizable graph on $k+1$ vertices admits a P_4-simplicial order as described. Let G be such a graph. Then, if G is chordal, G admits such a P_4-simplicial order: simply consider the reverse of

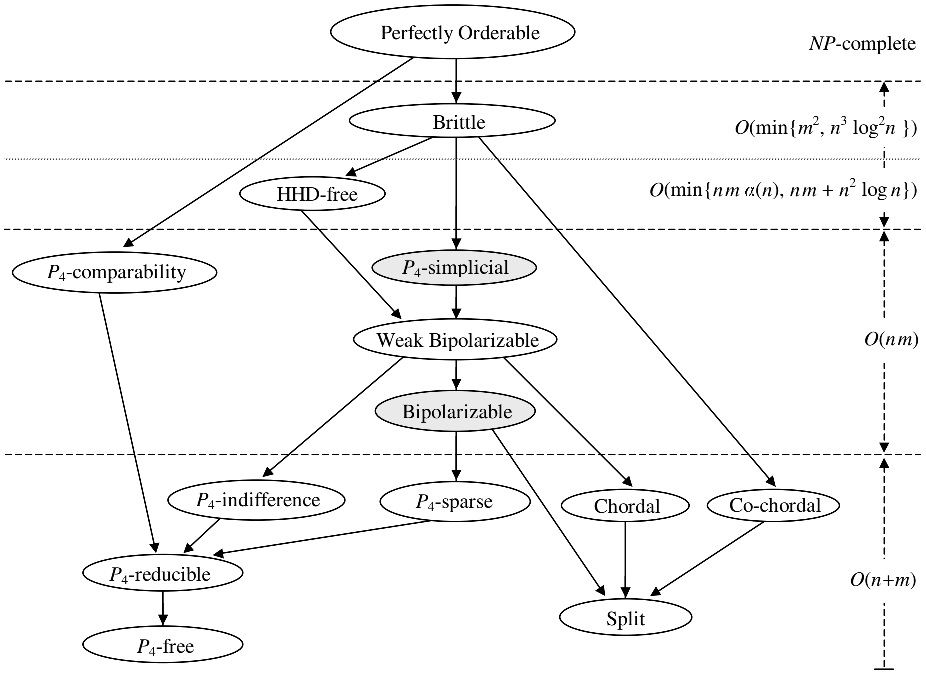


Fig. 7: Class inclusions and recognition time complexities.

the perfect elimination ordering of G's vertices produced by running the LexBFS algorithm on G starting at v [27]; note that if the ordering returned by LexBFS is $(v_1, v_2, \ldots, v_n)$, then v_i is simplicial with respect to the subgraph $G[\{v_i, \ldots, v_n\}]$ and thus cannot form a P_3 $v_j v_i v_\ell$ with $j, \ell > i$ participating in a P_4 of G (see Property 4.1).

If G is not chordal, then from [25] we have that G contains a homogeneous set. Let S be a minimal such homogeneous set and let u be a vertex in S; if $v \in S$, we choose u to be v. Then, the subgraph G_u of G induced by the vertices in $\big(V(G) - V(S)\big) \cup \{u\}$ is also a weak bipolarizable graph which contains v and has at most k vertices, since $|V(S)| \geq 2$. By our inductive hypothesis, G_u admits a P_4-simplicial order $\prec$ such that $v \prec x$ for all vertices $x \in V(G_u) - \{v\}$. On the other hand, the subgraph $G[S]$ is chordal, because it is a subgraph of a weak bipolarizable graph and S was chosen to be minimal [25]. If we replace u in the P_4-simplicial order $\prec$ of the subgraph G_u by the reverse of the ordering of the vertices of S which implements a perfect elimination scheme on S produced by the LexBFS algorithm starting at u, we are guaranteed to obtain a P_4-simplicial ordering of the vertices of G as desired. The inductive proof is complete, and thus every weak bipolarizable graph admits a P_4-simplicial order meeting the conditions of the statement of the proposition. □

We are now ready to prove the inclusion relations of the class of P_4-simplicial graphs.

Lemma 5.1 *The class of P_4-simplicial graphs is a proper subset of the class of brittle graphs and a proper superset of the class of weak bipolarizable graphs.*

Proof: The fact that P_4-*simplicial* $\subseteq$ *Brittle* has been shown in [16]; to see that the subset relation is proper, simply consider the graph D_6 of Figure 1 which is brittle but not P_4-simplicial. On the other hand, Proposition 5.1 establishes that *Weak Bipolarizable* $\subseteq$ P_4-*simplicial*. The proper inclusion follows from the fact that the house graph is P_4-simplicial but not weak bipolarizable. □

Regarding the relation of P_4-simplicial and the HHD-free and co-chordal graphs, note that the graph D_6 of Figure 1 is both HHD-free and co-chordal but is not P_4-simplicial whereas the house graph and P_5 are P_4-simplicial but not HHD-free and not co-chordal respectively.

Lemma 5.2 *The class of bipolarizable graphs is a proper subset of the class of weak bipolarizable graphs and a proper superset of the classes of* P_4*-sparse and split graphs.*

Proof: The fact that *Bipolarizable* $\subset$ *Weak Bipolarizable* has been established in [25]. To establish the relationship of P_4-sparse and bipolarizable graphs, we note that none of the forbidden subgraphs for the class of bipolarizable graphs is P_4-sparse; see Figure 1 and note that a k-wheel of order $k \geq 2$ contains the P_5 $s_0 r_0 v_0 v_{k-1} r_1$. This implies that any graph which is not bipolarizable cannot be P_4-sparse, or conversely that P_4-*sparse* $\subseteq$ *Bipolarizable*. The proper inclusion follows from the fact that a P_5 is bipolarizable but not P_4-sparse.

It is not difficult to see that *Split* $\subseteq$ *Bipolarizable*; the vertex set of a split graph can be partitioned into an independent set and a clique, which implies that any P_4 of a split graph has its midpoints in the clique and its endpoints in the independent set. Thus any ordering of the vertices of a split graph where all the vertices of the clique precede all the vertices of the independent set gives a bipolarizable ordering of the graph. The proper inclusion follows from the fact that C_4 is bipolarizable but not split. □

Lemma 5.3 *The class of* P_4*-indifference graphs is a proper subset of the class of weak bipolarizable graphs and a proper superset of the class of* P_4*-reducible graphs.*

Proof: The fact that P_4-*indifference* $\subset$ *Weak Bipolarizable* follows from the fact that the set of forbidden subgraphs for the class of weak bipolarizable graphs is a proper subset of the set of forbidden subgraphs for the class of P_4-indifference graphs (compare [25] and [15]).

To see that P_4-*reducible* $\subseteq$ P_4-*indifference*, we recall that every vertex of a P_4-reducible belongs to at most one P_4, which implies that the P_4s of a P_4-reducible graph are vertex-disjoint. Thus, we can create a linear order of the vertices of such a graph by concatenating the vertices of each P_4 at a time, in the order they appear along the P_4, and by appending any remaining vertices; then, the resulting ordering is a P_4-indifference ordering of the vertices of the graph. The proper inclusion follows from the fact that the P_5 is a P_4-indifference graph but not P_4-reducible. □

The non-inclusion relation between bipolarizable and co-chordal graphs follows from the counterexamples for the non-inclusion relation of the P_4-simplicial and co-chordal graphs. A non-inclusion relation also holds for the bipolarizable and the chordal graphs (consider a C_4 and the z-triomino) and for the bipolarizable and the P_4-indifference graphs (consider the forbidden subgraphs F_5 of [15] and the z-triomino).

In Figure 7, we have also partitioned the depicted classes of graphs based on the time complexities of the currently best recognition algorithms: see [7; 28] for the $O(\min\{m^2, n^3 \log^2 n\})$-time complexity range, [24] for the $O(\min\{nm\alpha(n), nm + n^2 \log n\})$-time complexity range, [23; 25] for the $O(nm)$-time complexity range, and [10; 26; 19; 20; 5; 27; 9; 11] for the $O(n + m)$-time range. We note that the algorithm of [25] for the recognition of weak bipolarizable graphs has a stated time complexity of $O(n^3)$;

since $O(n+m)$ time suffices to determine whether a graph is chordal and to compute a homogeneous set (by means of modular decomposition [21; 6]), if one exists, the stated time complexity can be seen to be $O(nm)$.

6 Concluding Remarks

We have presented recognition algorithms for the classes of bipolarizable (also known as Raspail) and P_4-simplicial graphs running in $O(nm)$ time, where n and m are the number of vertices and of edges of the input graph. Our proposed algorithms are simple, use simple data structures and require $O(n+m)$ space; the algorithms can also be augmented so that they return a certificate, whenever they decide that the input graph is not bipolarizable or P_4-simplicial, in $O(n+m)$ additional time and space. We have also presented results on class inclusions and recognition time complexities for a number of perfectly orderable classes of graphs.

We leave as an open problem the designing of $o(nm)$-time algorithms for recognizing bipolarizable and/or P_4-simplicial graphs. In light of the linear-time recognition of P_4-indifference graphs [10; 26], it would be worth investigating whether the recognition of P_4-comparability, P_4-simplicial, and bipolarizable graphs is inherently more difficult; it must be noted that the approach used in [10; 26] is different from those used for the recognition of the remaining classes as it reduces in part the problem to the recognition of interval graphs which can be carried out in linear time.

Finally, another interesting open problem is that of obtaining a characterization of the P_4-simplicial graphs by forbidden subgraphs. We note that any forbidden subgraph for the class of P_4-simplicial graphs other than a C_k for $k \geq 5$, an A, and a D_6 contains an induced house (see Figure 6): if it did not, then it would be weak bipolarizable [25], and hence a P_4-simplicial graph due to Lemma 5.1.

References

[1] C. Berge, Färbung von Graphen deren sämtliche bzw. deren ungerade Kreise starr (Zusammenfassung), *Wissenschaftliche Zeitschrift*, Martin Luther Universität Halle-Witterberg, Mathematisch-Naturwissenschaftliche Reihe, 114–115, 1961.

[2] A. Brandstädt, V.B. Le, and J.P. Spinrad, *Graph classes: A survey,* SIAM Monographs on Discrete Mathematics and Applications, 1999.

[3] M. Chudnovsky, N. Robertson, P. Seymour, and R. Thomas, The strong perfect graph theorem, to appear in *Annals of Mathematics*.

[4] V. Chvátal, Perfectly ordered graphs, *Annals of Discrete Math.* **21**, 63–65, 1984.

[5] D.G. Corneil, Y. Perl, and L.K. Stewart, A linear recognition algorithm for cographs, *SIAM J. Comput.* **14**, 926–934, 1985.

[6] E. Dahlhaus, J. Gustedt, and R.M. McConnell, Efficient and practical algorithms for sequential modular decomposition, *J. Algorithms* **41**, 360–387, 2001.

[7] E.M. Eschen, J.L. Johnson, J.P. Spinrad, and R. Sritharan, Recognition of some perfectly orderable graph classes, *Discrete Appl. Math.* **128**, 355–373, 2003.

[8] M.C. Golumbic, *Algorithmic Graph Theory and Perfect Graphs*, Academic Press, Inc., 1980.

[9] M. Habib, R.M. McConnell, C. Paul, and L. Viennot, Lex-BFS and partition refinement, with applications to transitive orientation, interval graph recognition and consecutive ones testing, *Theoret. Comput. Sci.* **234**, 59–84, 2000.

[10] M. Habib, C. Paul, and L. Viennot, Linear time recognition of P_4-indifference graphs, *Discrete Math. and Theoret. Comput. Sci.* **4**, 173–178, 2001.

[11] P.L. Hammer and B. Simeone, The splittance of a graph, *Combinatorica* **1**, 275–284, 1981.

[12] A. Hertz, Bipolarizable graphs, *Discrete Math.* **81**, 25–32, 1990.

[13] C.T. Hoàng, On the complexity of recognizing a class of perfectly orderable graphs, *Discrete Appl. Math.* **66**, 219–226, 1996.

[14] C.T. Hoàng and N. Khouzam, On brittle graphs, *J. Graph Theory* **12**, 391–404, 1988.

[15] C.T. Hoàng, F. Maffray, and M. Noy, A characterization of P_4-indifference graphs, *J. Graph Theory* **31**, 155–162, 1999.

[16] C.T. Hoàng and B.A. Reed, Some classes of perfectly orderable graphs, *J. Graph Theory* **13**, 445–463, 1989.

[17] C.T. Hoàng and B.A. Reed, P_4-comparability graphs, *Discrete Math.* **74**, 173–200, 1989.

[18] C.T. Hoàng and R. Sritharan, Finding houses and holes in graphs, *Theoret. Comput. Sci.* **259**, 233–244, 2001.

[19] B. Jamison and S. Olariu, A linear-time recognition algorithm for P_4-sparse graphs, *SIAM J. Comput.* **21**, 381–407, 1992.

[20] B. Jamison and S. Olariu, A linear-time algorithm to recognize P_4-reducible graphs, *Theoret. Comput. Sci.* **145**, 329–344, 1995.

[21] R.M. McConnell and J. Spinrad, Linear-time modular decomposition and efficient transitive orientation, *Proc. 5th Annual ACM-SIAM Symp. on Discrete Algorithms (SODA'94)*, 536–545, 1994.

[22] M. Middendorf and F. Pfeiffer, On the complexity of recognizing perfectly orderable graphs, *Discrete Math.* **80**, 327–333, 1990.

[23] S.D. Nikolopoulos and L. Palios, Algorithms for P_4-comparability graph recognition and acyclic P_4-transitive orientation, *Algorithmica* **39**, 95–126, 2004.

[24] S.D. Nikolopoulos and L. Palios, Recognizing HHD-free and Welsh-Powell opposition graphs, *Proc. 30th Int. Workshop on Graph-Theoretic Concepts in Computer Science (WG'04)*, LNCS 3353, 105–116, 2004.

[25] S. Olariu, Weak bipolarizable graphs, *Discrete Math.* **74**, 159–171, 1989.

[26] R. Rizzi, On the recognition of P4-indifferent graphs, *Discrete Math.* **239**, 161–169, 2001.

[27] D.J. Rose, R.E. Tarjan, and G.S. Lueker, Algorithmic aspects of vertex elimination on graphs, *SIAM J. Comput.* **5**, 266–283, 1976.

[28] A.A. Schäffer, Recognizing brittle graphs: remarks on a paper of Hoàng and Khouzam, *Discrete Appl. Math.* **31**, 29–35, 1991.