Available online at www.sciencedirect.com
ScienceDirect
Discrete Applied Mathematics 156 (2008) 2469–2479
DISCRETE APPLIED MATHEMATICS
www.elsevier.com/locate/dam

Polarity of chordal graphs

Tınaz Ekim[a,*], Pavol Hell[b], Juraj Stacho[b], Dominique de Werra[a]

[a] *ROSE, Ecole Polytechnique Fédérale de Lausanne, EPFL, Switzerland*
[b] *School of Computing Science, Simon Fraser University, Burnaby, B.C., Canada V5A 1S6*

Received 2 February 2007; received in revised form 8 June 2007; accepted 15 January 2008
Available online 21 March 2008

Abstract

Polar graphs are a common generalization of bipartite, cobipartite, and split graphs. They are defined by the existence of a certain partition of vertices, which is NP-complete to decide for general graphs. It has been recently proved that for cographs, the existence of such a partition can be characterized by finitely many forbidden subgraphs, and hence tested in polynomial time. In this paper we address the question of polarity of chordal graphs, arguing that this is in essence a question of colourability, and hence chordal graphs are a natural restriction. We observe that there is no finite forbidden subgraph characterization of polarity in chordal graphs; nevertheless we present a polynomial time algorithm for polarity of chordal graphs. We focus on a special case of polarity (called monopolarity) which turns out to be the central concept for our algorithms. For the case of monopolar graphs, we illustrate the structure of all minimal obstructions; it turns out that they can all be described by a certain graph grammar, permitting our monopolarity algorithm to be cast as a certifying algorithm.

Keywords: Chordal graph; Polar graph; Polynomial algorithm; Graph grammar; Matrix partition; Generalized colouring; Forbidden subgraph characterization

1. Introduction

It is well known that many colouring (or partition) problems, while NP-complete in general, can be efficiently solved on the class of chordal graphs. This includes classical colouring problems [9], as well as generalized colourings [1], and matrix partitions [5,6,11]. In this paper we consider the case of polar partitions.

A *polar partition* of a graph G is a partition of the vertex set $V(G)$ into two subsets V_r, V_b, such that V_b induces no P_3 and V_r induces no $\overline{P}_3$. Note that a graph has no induced P_3 if and only if it is a *disjoint union* of cliques, with no other edges. (The binary relation 'adjacent or equal' becomes an equivalence relation in this situation.) Thus a partition $V(G) = V_r \cup V_b$ is polar if and only if V_b induces a disjoint union of cliques and V_r induces a complete multipartite graph. (The edges between the two parts are not restricted.)

A graph G is *polar* if it admits a polar partition. As all partitions, a polar partition can be usefully viewed as a *colouring* of the vertices of G: we shall refer to elements of the sets V_r and V_b as *red* (r) and *blue* (b) vertices respectively.

* Corresponding address: Bogazici University, Department of Industrial Engineering, 34342 Bebek-Istanbul, Turkey. Tel.: +90 212 359 66 76; fax: +90 212 265 18 00.
E-mail addresses: tinaz.ekim@epfl.ch, tinaz.ekim@boun.edu.tr (T. Ekim), pavol@cs.sfu.ca (P. Hell), jstacho@cs.sfu.ca (J. Stacho), dominique.dewerra@epfl.ch (D. de Werra).

0166-218X/$ - see front matter
doi:10.1016/j.dam.2008.01.026

It follows from this definition that the complement of a polar graph is also polar and that polar graphs include such well-studied graph classes as split graphs, bipartite graphs, and cobipartite graphs [9]. The problem of polar partitionability is NP-complete in general graphs [3], but has been solved in polynomial time for the class of cographs [2]. As suggested above, a natural class to consider in this context is the class of chordal graphs; we develop the first polynomial time algorithm for polarity of chordal graphs.

A graph is *chordal* if it admits no induced cycle of length greater than three. Equivalently, a graph is chordal if its vertices can be ordered as $v_1, v_2, \ldots, v_n$, so that, for each $i = 1, 2, \ldots, n$, any two neighbours $v_j, v_{j'}$ of v_i with $j \neq j', j > i, j' > i$ are adjacent [9]. Such an ordering is called a *perfect elimination ordering* of G. A perfect elimination ordering of a given graph G can be found in linear time [9,15]. *Linear time* will always mean linear in the number of vertices and edges. If the graph G is not chordal, the algorithm will in the same time bound produce an induced cycle of length greater than three [16], certifying its non-chordality. In a chordal graph with a perfect elimination ordering $v_1, v_2, \ldots, v_n$, we denote by F_i the clique consisting of v_i and all its neighbours $v_j, j > i$. These n cliques F_i can be found, also in linear time, once the perfect elimination ordering is known. It is easy to see [9] that each maximal clique of a chordal graph G is equal to some F_i. Therefore, in what follows, we will be able to test various properties of all maximal cliques by testing all cliques F_i.

While the polarity of cographs can be characterized by finitely many forbidden induced subgraphs, we show that there is no finite forbidden induced subgraph characterization of polar chordal graphs. (For a special class of polar chordal graphs we suggest a simple recursive construction generating all minimal obstructions, see Fig. 4; this will be taken up in [13].)

Polar partitions of chordal graphs take a restricted form. The complete multipartite graph induced by the red vertices has at most one part with more than one vertex. (If two parts have at least two vertices each, we would obtain an induced cycle with four vertices.) In particular, in a polar partition of a chordal graph, the red vertices induce the join of a clique and a stable set. (This means each vertex of the clique is adjacent to each vertex of the stable set.) In what follows, unless explicitly stated otherwise, we shall only consider chordal graphs G. Therefore we amend the definition of a polar partition as follows: a polar partition of a chordal graph G is a partition of its vertices into a red stable set A, a red clique B, and blue cliques $C_1, C_2, \ldots, C_k$, so that A is joined to B by all possible edges, and no edges join a C_i to any C_j with $i \neq j$.

In [4], the authors develop a general framework for similar partitions. Given a symmetric m by m matrix M over $0, 1, *$, they define an *M-partition* of a graph G to be a partition of the vertices of G into sets $V_1, V_2, \ldots, V_m$, such that V_i is a stable set if $M(i, i) = 0$ and a clique if $M(i, i) = 1$, and such that, for $i \neq j$, the set V_i is joined to the set V_j by no edges if $M(i, j) = 0$, and by all possible edges if $M(i, j) = 1$. Asterisks indicate no restriction. It is easy to see that a polar partition into $A, B, C_1, C_2, \ldots, C_k$ fits the definition of an M-partition for a suitable matrix $M = M_k$ with $m = k + 2$. The authors of [5] have investigated the complexity of M-partitions in chordal graphs, and found polynomial time algorithms for many natural matrices M, including the above matrices M_k, for any *fixed* integer k. Surprisingly, there exist matrices M for which the M-partition problem is NP-complete even when restricted to chordal graphs [5]. Let M_∞ denote the obvious infinite generalization of M_k, corresponding to polar partitions $A, B, C_1, C_2, \ldots$ (k is not fixed). Our problem is precisely the problem of M_∞-partitionability of chordal graphs, and our algorithm is one of the very few known non-trivial cases of polynomial M-partition problems with infinite matrix M.

Our main goal is to present a polynomial time algorithm for finding a polar partition of a chordal graph, if one exists — see Section 5. The algorithm is obtained by combining the solutions of a number of special cases investigated in the next two sections.

We mostly follow the standard terminology and notation of [18]. In particular, a *block of* G is a maximal connected subgraph of G without cutpoints. A block consisting of two adjacent vertices is called a *trivial block*, blocks with at least three vertices are called *non-trivial*. A 2-*connected graph* is a graph G that is a non-trivial block of itself.

Given a graph G and a set $U \subseteq V(G)$, we denote by $G - U$ the subgraph of G obtained by removing all vertices in U; if $U = \{u\}$, we simply write $G - u$.

2. Special classes of polar graphs

A few special kinds of polar partitions play a role in our approach. First we state them for general graphs G: a polar partition is called *monopolar* if the red vertices form a stable set, and *unipolar* if the red vertices form a clique. A graph is *monopolar*, or *unipolar*, if it admits a monopolar, respectively unipolar, partition. This definition

of monopolar graphs, more convenient for our purposes, differs from that in [2]; it corresponds to the so-called *stable monopolar* graphs of [2]. Observe that a partition is monopolar if and only if there is no red P_2 and no blue P_3; similarly, a partition is unipolar if and only if there is no red $\overline{P}_2$ and no blue P_3. A *split partition* is a monopolar partition in which the blue vertices form a clique; a graph which admits a split partition is called a *split graph* [9]. Split graphs can be recognized in linear time, even if some vertices are precoloured (preassigned to be in the stable set (precoloured red), or in the clique (precoloured blue)) [11].

For a chordal graph G, a polar partition is monopolar if and only if the red clique B is empty, and unipolar if and only if the red stable set A is empty. A monopolar partition of a chordal graph is a split partition if and only if there is exactly one blue clique.

As noted above, we can obtain all maximal cliques of a chordal graph in linear time. This is sufficient to recognize *unipolar chordal graphs* in polynomial time as well. Indeed, we have the following observation.

Proposition 1 (*Unipolarity of Chordal Graphs*). *A chordal graph G is unipolar if and only if it has a maximal clique U such that $G - U$ is a disjoint union of cliques.*

Proof. If such a clique U exists, we can colour its vertices red and colour all remaining vertices blue. On the other hand, if the vertices are coloured red and blue so that the red vertices form a clique, then any maximal clique containing the red vertices can serve as U. □

Testing whether a graph is a disjoint union of cliques can easily be accomplished by finding its components and checking their edges in linear time. Thus we obtain an $O(n^3)$ unipolarity algorithm for chordal graphs. (For simplicity we express the higher complexities purely in terms of the number of vertices n.) In fact, we can apply the algorithm to test for unipolarity of a graph G in which some vertices have been precoloured. It suffices to consider only those cliques F_i which contain all the red precoloured vertices; and to test sets U consisting of F_i from which all the blue precoloured vertices have been removed. If some U results in $G - U$ being a disjoint union of cliques, we have a unipolar partition observing the precolouring, otherwise such a unipolar partition does not exist.

We remark that unipolar graphs can be recognized in polynomial time even for general graphs [17], and that unipolar chordal graphs can in fact be characterized by a single forbidden induced subgraph, namely $2P_3$ [8]. By contrast, it follows from [7] that recognizing monopolar graphs in general is NP-complete.

In the next section, we shall focus on monopolar partitions in chordal graphs. For now, we take up an important special class of monopolar partitions, which will play a role in Section 5. A *singly monopolar partition* of a chordal graph G is a monopolar partition of G in which each blue clique C_i has at most one red neighbour. (A red neighbour of C_i is a red vertex adjacent to at least one vertex of C_i.) A chordal graph that admits a singly monopolar partition is called *singly monopolar*.

It follows that in any singly monopolar partition of a chordal graph G, each connected component has at most one red vertex; and if a component K has a red vertex v, then $K - v$ is a disjoint union of cliques.

For a connected graph H, we denote by V_H the set of all vertices u such that $H - u$ is a disjoint union of cliques. Of course, the set V_H may be empty.

Proposition 2 (*Single Monopolarity of Chordal Graphs*). *A chordal graph G is singly monopolar if and only if each component K of G has $V_K \neq \emptyset$. All singly monopolar partitions of G are obtained by choosing one vertex of each V_K to be coloured red, except for the components K that are cliques, where we may choose to colour no vertex red.* □

We note that the resulting algorithm can also be easily adapted to allow for precoloured vertices.

There are only a few possible ways a connected graph H can have non-empty V_H. Specifically, if $H - u$ is P_3-free, then either H is P_3-free and hence H is a complete graph, $V_H = V(H)$, and each $H - u$ has just one component; or there is an induced P_3 in H and hence V_H can have at most three vertices. (A similar argument deals with removing vertices u so that $H - u$ is F-free for any fixed F.)

3. Monopolar chordal graphs

The most interesting special case is that of monopolar chordal graphs. Since the disjoint union of two monopolar graphs is again monopolar, we shall focus on connected graphs. For 2-connected graphs we have the following observation.

Proposition 3. *A 2-connected graph G is both chordal and monopolar if and only if it is a split graph.*

Proof. Clearly if G is a split graph, it is also monopolar and chordal. Conversely, if G is a 2-connected chordal graph, and X a stable set such that $G - X$ is a disjoint union of cliques $C_1, C_2, \ldots, C_k$, then we must have $k = 1$, or else X would contain a minimal cutset of G which is not a clique (impossible in chordal graphs), or has one vertex (impossible in 2-connected graphs). Thus G is a split graph. □

For connected chordal graphs that contain cutpoints, we shall analyze the structure of their blocks. It is easy to see that a non-trivial block must have each vertex in a triangle. Thus we obtain the following frequently used observation.

Proposition 4. *In any monopolar partition of a non-trivial block of a chordal graph, each vertex has a blue neighbour.* □

The structure of the blocks will be analyzed using a variant of the *block-cutpoint tree* $T(G)$ of G [18]. We augment the set of *real cutpoints* (which are as usual the vertices whose removal disconnects the graph), by adding *artificial cutpoints*, namely, those vertices belonging to trivial blocks which are not real cutpoints. This will be illustrated (see Fig. 2) further in this section after we present the rules for recognizing monopolar multicolorings. We shall use the term *cutpoint* to mean either a real cutpoint or an artificial cutpoint. The *nodes* of the block-cutpoint tree $T(G)$ are all blocks and all cutpoints of G. In $T(G)$ we have the following edges: a cutpoint is adjacent to all blocks to which it belongs. The tree $T(G)$ is considered rooted at a fixed cutpoint $root$; we write $p(v)$ for the parent of node v, write $ch(v)$ for the set of vertices w such that $p(w) = v$ (the children of v), write $gc(v)$ for the set of vertices w such that $p(p(w)) = v$ (the grandchildren of v), and write $tg(v)$ for the set of all $w \in gc(v)$ such that $p(w)$ is a trivial block (the *trivial* grandchildren). We call $w \in ch(v)$ *trivial child* (respectively *non-trivial child*) of v if w is a trivial (respectively non-trivial) block. Note that the leaves of $T(G)$ are either (artificial) cutpoints or non-trivial blocks.

Instead of just finding one monopolar partition of G, we shall compute a structure that contains all such partitions. A *multicolouring* ℓ of G is a mapping which assigns to each vertex of G a *set* of colours $\ell(v) \subseteq \{r, b\}$ (recall that r stands for red, b for blue), called the *list* of v. For two multicolourings ℓ, ℓ' of G, we shall write $\ell \preccurlyeq \ell'$ if $\ell(v) \subseteq \ell'(v)$ for each $v \in V(G)$. If $\ell(v)$ contains a single colour for each $v \in V(G)$, then ℓ assigns to each vertex a single colour (red or blue), and we view it as a colouring, or partition, of G. If a colouring (partition) ℓ satisfies $\ell \preccurlyeq \ell'$, we say that ℓ is *contained* in ℓ', and ℓ' *contains* ℓ. (Note that ℓ is obtained by choosing one colour from each list $\ell'(v)$.) A multicolouring of G is a *monopolar multicolouring* if it contains a monopolar partition of G. We shall usually write r for the singleton set $\{r\}$ and b for the singleton set $\{b\}$. Note that the *trivial multicolouring* L with $L(v) = \{r, b\}$, for all $v \in V(G)$, is the largest element of the partial order $\preccurlyeq$. To decide if G has a monopolar partition, we only need to decide whether the trivial multicolouring L is monopolar. In the algorithm below, we may start with this multicolouring $\ell_0 = L$, or we may start with a more restrictive multicolouring ℓ_0, if certain vertices have been already precoloured.

We shall compute a final multicolouring which contains all possible monopolar partitions of G contained in the initial multicolouring ℓ_0. We shall construct the lists in a bottom-up fashion in the block-cutpoint tree $T(G)$. We shall focus the computation on the nodes of $T(G)$ which are cutpoints of G, and so proceed from a vertex to its grandparent. At any time in the computation, S will denote the set of already processed vertices.

It is important to remember, when reading the rest of this section, that a partition assigns to each vertex a single colour, and a multicolouring assigns to each vertex a set (list) of colours.

RECOGNIZING MONOPOLAR MULTICOLOURINGS
Input: A chordal graph G with a multicolouring ℓ_0.
Task: Decide if ℓ_0 is a monopolar multicolouring.
Action: Obtain the block-cutpoint tree T of G. Denote by C its set of cutpoints, and root T at some fixed $root \in C$. If some block is not a split graph, then ℓ_0 is not a monopolar multicolouring. Otherwise, starting with the initial multicolouring $\ell \leftarrow \ell_0$, and the initial set $S \leftarrow \emptyset$ (of already processed cutpoints), modify the current ℓ and S as follows.
As long as $S \neq C$, pick a vertex $v \in C \setminus S$ such that $gc(v) \subseteq S$, add v to S, and process its list $\ell(v)$ by performing the following two steps.
- Apply the Rules 1–8, and
- apply the Block Rule.

If the final multicolouring $\ell^* \leftarrow \ell$ has some list $\ell^*(v)$ empty, then ℓ_0 is not a monopolar multicolouring of G; otherwise it is.

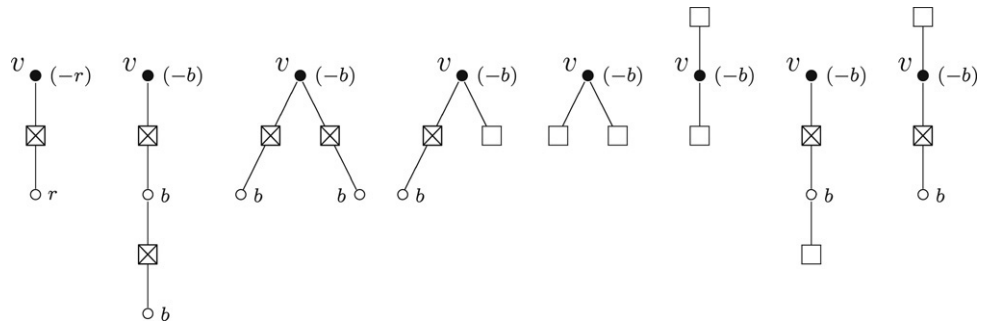

Fig. 1. Rules 1–8.

Recall that C includes the artificial cutpoints (as defined earlier). Note that the condition $gc(v) \subseteq S$ is vacuously satisfied for the artificial cutpoints of T, so they can be added to S at any time.

We first explain the Rules. Recall that v is a cutpoint of G whose grandchildren (if any) have all been already processed. The role of the Rules is to (possibly) reduce the lists $\ell(v)$, eliminating colours that we know can no longer be used. They are depicted in Fig. 1, in left to right order. We use ⊠ to denote trivial blocks, □ for non-trivial blocks, and ∘ for cutpoints.

Rule 1. If some trivial grandchild w of v has $\ell(w) = r$, then remove r from $\ell(v)$.

Rule 2. If some trivial grandchild w of v, and some trivial grandchild w' of w, have $\ell(w) = \ell(w') = b$, then remove b from $\ell(v)$.

Rule 3. If some trivial grandchildren w, w' of v have $\ell(w) = \ell(w') = b$, then remove b from $\ell(v)$.

Rule 4. If some trivial grandchild w of v has $\ell(w) = b$ and v also has a child that is a non-trivial block, then remove b from $\ell(v)$.

Rule 5. If v has two children that are non-trivial blocks, then remove b from $\ell(v)$.

Rule 6. If both the parent of v and some child of v are non-trivial blocks, then remove b from $\ell(v)$.

Rule 7. If some trivial grandchild w of v has $\ell(w) = b$, and w has a child that is a non-trivial block, then remove b from $\ell(v)$.

Rule 8. If some trivial grandchild w of v has $\ell(w) = b$, and the parent of v is a non-trivial block, then remove b from $\ell(v)$.

The block rule. For each non-trivial block $H \in ch(v)$, remove from the current list $\ell(v)$ each colour (r, b) which cannot be the colour of v in a split partition of H contained in ℓ.

These Rules are easy to implement directly. The Block Rule requires us to test whether a graph with some vertices precoloured r or b admits a split partition; an algorithm for this can be found in [11], cf. [10].

Rules 1–8 are justified by the observation that we only remove a colour from $\ell(v)$ when no monopolar partition contained in ℓ can select it from $\ell(v)$, because the vertices labeled r must be P_2-free, and the vertices labeled b must be P_3-free — for Rules 4–8, see Proposition 4.

We have illustrated an example application of the rules in Fig. 2. On the left is the chordal graph to be tested for monopolarity; note that it has one vertex precoloured r. In the middle is its block-cutpoint tree; note that the top vertex is in fact an artificial cutpoint. Since the precoloured vertex is inside a block it is not visible in the block-cutpoint tree. Nevertheless when the Block Rule is applied, it forces the colour of the (unique) cutpoint of that block to be b. Further applications of Rules 7, 1, 5, 1, and 2, in that order, result in the colouring on the right.

Next we show that any monopolar partition of G contained in ℓ_0 is also contained in ℓ^*. Applying this to the trivial multicolouring ℓ_0 in which all lists $\ell_0(v) = \{r, b\}$, we obtain a multicolouring ℓ^* which contains all monopolar partitions of G.

Thus let ℓ_0 denote the input multicolouring of G, let ℓ denote the changing multicolouring during the execution of the algorithm, and let ℓ^* denote the final multicolouring produced by the algorithm. The correctness of the algorithm is justified by the following fact.

Proposition 5. *The final multicolouring ℓ^* contains every monopolar partition contained in ℓ_0.*

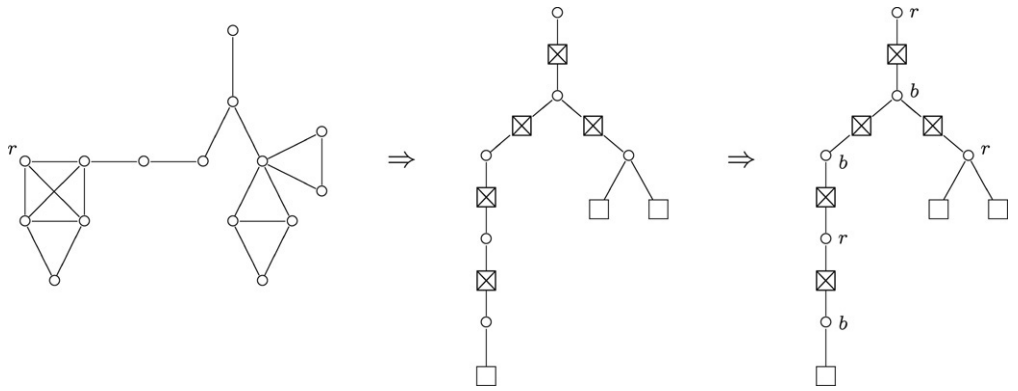

Fig. 2. An illustration of the monopolarity recognition algorithm.

Proof. We shall prove by induction on number of iterations of the algorithm, that this remains true for the changing multicolouring ℓ. It is clearly true for $\ell = \ell_0$, so we assume that this is so for the current ℓ, and perform one iteration of the algorithm. The statement remains valid after the vertex v is processed, since Rules 1–8, as well as the Block Rule are used to eliminate only those colours that could never be used in a monopolar partition. □

In particular, we obtain

Corollary 6. *If some vertex v of G has $\ell^*(v) = \emptyset$, then the multicolouring ℓ_0 is not monopolar; in particular if this happens when $\ell_0 = L$, then G is not a monopolar graph.* □

We shall also use the following fact about the monopolar recognition algorithm.

Proposition 7. *Suppose the execution of the monopolar recognition algorithm has just processed a vertex v, without creating an empty list $\ell(x)$ for any processed vertex x. Then we have*

- *if $r \in \ell(v)$ then $b \in \ell(w)$ for all $w \in tg(v)$,*
- *if $b \in \ell(v)$ then $r \in \ell(w)$ for all $w \in tg(v)$ except possibly one $w = w_0$, and if w_0 is such an exception, then*
  - *$\ell(w_0) = \{b\}$*
  - *$r \in \ell(w')$ for all $w' \in tg(w_0)$,*
  - *both v and w_0 have only trivial blocks as children, and*
  - *if $v \neq root$, then $p(v)$ is a trivial block.*

Proof. Suppose first that $r \in \ell(v)$ but $b \notin \ell(w)$ for some $w \in tg(v)$. Then $\ell(w) = \{r\}$ since otherwise $\ell(w) = \emptyset$ and algorithm would stop at w never reaching v. Therefore r was eliminated from the list of v during its processing (using Rule 1), which leads to a contradiction.

Similarly suppose that $b \in \ell(v)$ but $r \notin \ell(w_0)$ for some $w_0 \in tg(v)$. Then $\ell(w_0) = \{b\}$ and it follows from Rules 4, 7 and 8 that neither v nor w_0 have any non-trivial children or parents. Moreover observing Rules 2 and 3, we can also conclude that neither v nor w_0 have any trivial children with lists $\{b\}$ other than w_0. Therefore it follows that $r \in \ell(w')$ for all $w' \in tg(v)$ different from w_0 and also $r \in \ell(w'')$ for all $w'' \in tg(w_0)$. (Note that the lists of all these nodes must be non-empty since they were already processed.) □

In order to use the recognition algorithm for finding monopolar partitions, it remains to explain how to find a monopolar partition contained in ℓ^*, provided all lists are non-empty. We use a top-down procedure in which we reduce the lists $\ell^*(v)$ to single element lists $\ell_1(v)$, starting from the $root$ and proceeding down in the tree.

EXTRACTING A MONOPOLAR PARTITION
Input: A chordal graph G with a monopolar multicolouring ℓ^*.
Task: Construct a monopolar partition ℓ_1 contained in ℓ^*.
Action: Starting with the initial multicolouring $\ell \leftarrow \ell^*$, *process* vertices from the top down: if all ancestors of v have already been processed, then process v as follows.

- If $\ell(v) = r$, then set $\ell(w) \leftarrow b$ for all trivial grandchildren w of v.
- If $\ell(v) = b$, then set $\ell(w) \leftarrow r$ for all trivial grandchildren w of v which have $r \in \ell(w)$. If there is a trivial grandchild w_0 with $r \notin \ell(w_0)$ then set $\ell(w_0) \leftarrow b$ and set $\ell(w') \leftarrow r$ for all trivial grandchildren w' of w_0; also, mark w_0 as processed.
- If $\ell(v) = \{r, b\}$, then set $\ell(v)$ to either r or b and apply the corresponding step above.
- For each non-trivial child of v, i.e., a non-trivial block H, apply a split partition algorithm on H with respect to the precolouring ℓ, to set the $\ell(w)$ for all children w of H.

After all vertices have been processed, set the final multicolouring $\ell_1 \leftarrow \ell$.

The fact that the colours assigned are always available, as well as the fact that the exceptional grandchild w_0 is unique and has no non-trivial children, follow from Proposition 7. Note that the last bullet of the algorithm is justified by Rules 5 and 6, which allow us to deal with non-trivial children of v independently. The correctness is then implied by the following invariant being maintained by the algorithm.

Proposition 8. *Let v be the next vertex to be processed in the extraction algorithm. Then $\ell(v')$ has exactly one colour, for all ancestors v' of v, and if v has a trivial parent, then $\ell(v) \neq \ell(z)$ where z is the grandparent of v.* □

The invariant allows us to treat the subtree (of the block-cutpoints tree T) rooted at v independently of the remainder of T. It is now easy to see that the operation of the algorithm ensures that the following Corollary is true.

Corollary 9. *The partition ℓ_1 extracted from ℓ^* by the algorithm is monopolar.* □

Thus we have obtained the following result.

Theorem 10 (*Monopolar Chordal Graphs*). *There is a linear time algorithm to decide if a chordal graph is monopolar, and to find a monopolar partition if one exists.*

Proof. Split graphs can be recognized in linear time [9], even if some vertices are precoloured [11]. The block-cutpoint tree can also be constructed in linear time using [14]. Finally, each of the Rules 1–8 can be implemented in $O(\deg(v))$ steps, and the list of every vertex is accessed only a constant number of times during the execution of the algorithm. □

4. Forbidden subgraphs

In the case of cographs, it has been shown in [2] that polarity can be characterized by the absence of a finite set of forbidden induced subgraphs. By contrast, for chordal graphs, there are infinitely many minimal non-polar graphs. For instance it is easy to check (directly or by executing our algorithms) that any of the chordal graphs depicted in Fig. 3 is minimal non-polar; it is also minimal non-monopolar. Interestingly, it can be shown [13] that all chordal minimal non-monopolar graphs can be generated by a simple recursive procedure (which unwinds the operation of the monopolarity recognition algorithm in case it rejects ℓ_0). This turns out to be a particular example of the so-called *hyperedge replacement grammars* [12]. The grammar Γ constructs a tree-like structure (with at most three branches at any node) consisting of very simple blocks (with at most six vertices each). Fig. 3 suggests the idea of a repetition of simple substitutions; a more elaborate example of a graph generated by the grammar is given in Fig. 4. The grammar Γ will be described in detail in [13]. (We remark that this grammar, in fact, generates also all precoloured forbidden subgraphs for monopolarity of chordal graphs.)

Surprisingly, for 2-connected chordal graphs, there is a finite forbidden subgraph characterization of monopolarity.

Proposition 11. *A 2-connected chordal graph G is monopolar if and only if it does not contain any of the four forbidden induced subgraphs, given in Fig. 5.*

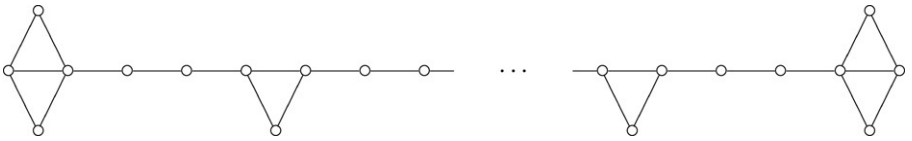

Fig. 3. An infinite family of chordal minimal non-polar graphs.

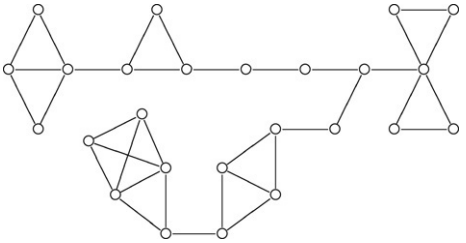

Fig. 4. An example chordal graph generated by the grammar Γ.

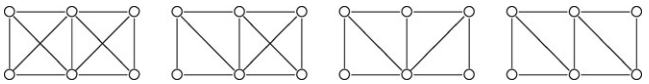

Fig. 5. All 2-connected minimal non-monopolar chordal graphs.

Proof. Clearly, the examples are 2-connected chordal graphs that are minimal non-monopolar. Conversely, by Proposition 3, it suffices to consider a 2-connected chordal graph G which is not a split graph. Hence G contains four vertices inducing a disjoint union of edges xx', yy'. (The other forbidden induced subgraphs for split graphs, not possible for chordal graphs, are cycles of length four and five [9].) By a simple application of Menger's theorem, we conclude that there exist disjoint paths P, P' from x to y and from x' to y'. (Add a vertex x^* adjacent to x, x' and a vertex y^* adjacent to y, y' and apply Menger's theorem to x^*, y^*.) This means we have a cycle xx', P', $y'y$, P. It can now easily be shown that chordality implies that the cycle must contain a six-cycle and hence one of the configurations listed in Fig. 5. □

We note that the grammar Γ allows the monopolarity recognition algorithm to be cast as a *certifying algorithm* — if the algorithm rejects ℓ_0, it generates a minimal non-monopolar subgraph. This could be one of the blocks in Fig. 5, or a subgraph generated by the grammar (such as the one illustrated in Figs. 3 or 4).

5. Polar chordal graphs

We are now ready to bring together the various pieces, and describe our polynomial time algorithm to test for polarity of chordal graphs.

Let M be a clique of a chordal graph G. We shall say that M is *good* if there exists a polar partition of G (with red independent set A, red clique B, and blue cliques $C_1, C_2, \ldots, C_k$), in which $B = M$. (Note that if M is a good clique, then $G - M$ is monopolar.) We shall say that M is *nice* if some polar partition of G (with sets $A, B, C_1, \ldots, C_k$ as above) has $M = B \cup T$ where $T \subseteq C_1$, and if $G - M$ is singly monopolar. Note that in both cases $A \cap M = \emptyset$.

A clique M is *almost good* if it is good or if $M \setminus \{a\}$ is good for some $a \in M$. A clique M is *almost nice* if it is nice or if $M \setminus \{a\}$ is nice for some $a \in M$.

It turns out that if a graph G is polar, there must be an almost good or an almost nice clique that is easy to find.

Proposition 12 (*Polarity of Chordal Graphs*). *Let G be a polar chordal graph which is not monopolar and not unipolar. Then at least one of the following must occur:*

(1) *G contains two non-adjacent vertices u, v such that $N(u) \cap N(v)$ is an almost good clique, or*
(2) *G contains a maximal clique that is almost nice.*

Proof. First we observe that the chordality of G implies that $N(u) \cap N(v)$ is always a clique. Thus suppose none of these cliques, over all pairs of vertices u, v is almost good, and consider a polar partition of G with the parts $A, B, C_1, \ldots, C_k$. Let T be the set of all vertices of $C_1, \ldots, C_k$ that are adjacent to each vertex of B. Since G is not unipolar, the set A has at least two vertices, and any two vertices u, v of A must have a common blue neighbour t, otherwise $N(u) \cap N(v) = B$. We now observe that t is adjacent to every vertex w in B, else we would have the induced four-cycle u, t, v, w without chords. In other words, $t \in T$, and the set T must be non-empty.

We now observe that any two vertices u, v of T are adjacent. Otherwise, the set $N = N(u) \cap N(v)$ would be a clique, as noted above. Moreover u, v would not be in the same blue clique C_i, and so the clique N would consist of red vertices of G. Since A is an independent set, this would mean that either N or some $N \setminus \{a\}, a \in A$, would be equal to B, i.e., N is almost good. Therefore T is included in some blue clique, without loss of generality, the clique C_1. From the definitions, we have $M = T \cup B$, a clique which contains B. We now claim that M is a nice clique, i.e., that $G - M$ is singly monopolar.

If a component K of $G - M$ contained two distinct red vertices, then let u and v be two nearest such vertices. Note that u, v must be in A, hence they are not adjacent. Since K is connected, there is in K a shortest path P from u to v (of length at least two); it follows from the choice of u, v that $P - \{u, v\}$ contains only blue vertices. We may also assume that B is not empty, since otherwise G would be monopolar. Then consider the cycle wPw for any $w \in B$. (Since u, v are in A, they must be adjacent to w). Since G is chordal and P is a shortest path, w must be adjacent to all vertices of P. It now follows that each vertex of $P - \{u, v\}$ is adjacent to all vertices w of B, and hence belongs to T, contradicting the fact that it is in $G - M$. Thus every component K has at most one red vertex. Therefore, every blue clique is adjacent to at most one red vertex.

It remains to observe that M is either a maximal clique of G, or becomes a maximal clique of G by the addition of a single vertex of A. Thus $M \cup \{a\}$ is almost nice. □

We next describe how to test whether a clique B is good.

Proposition 13 (*Good Clique*). *Suppose B is a clique in a chordal graph G. Precolour all vertices v in $G - B$ not completely joined to B by blue. Then B is good if and only if this precolouring can be extended to a monopolar partition of $G - B$.* □

Since our algorithm for monopolarity of chordal graphs allows precoloured vertices, it can be used for this purpose.

In the remainder of this section we describe how to test whether a clique M is nice, assuming that we have a chordal graph G that is not monopolar, not unipolar, and has no non-adjacent vertices u, v for which $N(u) \cap N(v)$ is an almost good clique. Recall that we already have an algorithm to test whether $G - M$ is singly monopolar; if it is not, then M is not a nice clique. Otherwise, we can proceed to colour the vertices of G red and blue, seeking a polar partition of G in which $M = B \cup T$ where $T \subseteq C_1$. We can take advantage of some additional information: the set A must have more than one vertex (since G is not unipolar), and any two vertices of A must have a common blue neighbour (else their common neighbourhood would have been the good clique B). This common blue neighbour must be in M, hence $T = M \setminus B$ must be non-empty.

Let $K_1, K_2, \ldots, K_h$ be the components of $G - M$. Recall that testing if $G - M$ is singly monopolar involves checking that each set V_{K_i} is non-empty. (Recall that V_{K_i} consists of all vertices x for which $K_i - x$ is a union of cliques.) At most one of these vertices may be placed in the red independent set A, while all the remaining vertices of K_i are placed in the blue cliques C_j. If K_i is a clique, then of course $V_{K_i} = V(K_i)$ and any vertex can be coloured red; in this case it is also possible to colour all vertices of K_i blue, i.e., the entire K_i is a blue clique.

It seems harder to identify the clique C_1. We can, however, try all the following possibilities for $C = C_1 - T$:

(1) $C = \emptyset$,
(2) $C = K_i$ for some $i = 1, 2 \ldots, h$,
(3) C is a component (clique) of $K_i - a$, for some $i = 1, 2, \ldots, h$, and some vertex $a \in V_{K_i}$.

In case 1, no component K_i is chosen; in all other cases we call the chosen component K_i the *exceptional* component, and the other components K_j, $j \neq i$ *non-exceptional*.

For each such possible scenario we colour G as follows.

In cases 2 and 3, the exceptional component is coloured by blue in case 2, and coloured by blue except for a, which is coloured red, in case 3. In these cases, we moreover colour all vertices of M non-adjacent to some vertex of C red.

To colour the non-exceptional components we consider the following line of thought, relative to a fixed polar partition of G. If a non-exceptional component K contains a (unique) red vertex, we shall denote it by v_K. Then v_K is adjacent to all vertices of B, while the other vertices of K are not adjacent to any vertex of $M \setminus B$. In fact, no vertex x of K other than v_K can be adjacent to all vertices of B. Otherwise, consider any vertex y in T: since x and y belong to different blue cliques, they must be non-adjacent, and $N(x) \cap N(y)$ must contain only red vertices. It now follows that $N(x) \cap N(y)$ is an almost good clique, which we assumed was not the case. In other words, the neighbourhood of v_K in M strictly contains the neighbourhood of any other vertex x of K in M. We shall call v_K the *maximum vertex* of K.

Consider first a non-exceptional component K which is not a clique. We colour the maximum vertex v_K red, and all other vertices of K blue. If K is a clique, we only colour the vertices v other than v_K blue, leaving the colour of v_K to be decided below.

At this point we shall propagate the colours as follows.

- *Propagation rule* 1. If $v \in K_j$ is red, then all its non-neighbours in M are blue.
- *Propagation rule* 2. If $u \in M$ is blue, then all its neighbours in any non-exceptional component K_j are red.

If a vertex receives both red and blue colours by these rules, then we declare M not nice. Otherwise, we set A to consist of all red vertices not in M, then colour all uncoloured vertices of M red, setting B to consist of all red vertices in M, and colour all remaining vertices not in M blue. It is now clear that A is an independent set and B is a clique, and that each vertex of A is adjacent to each vertex of B, by Propagation Rule 1. Moreover, $(G - A) - B$ is a union of cliques, by Propagation Rule 2 and the colouring of non-neighbours of C in M.

Proposition 14 (*Nice Clique*). *Suppose M is a clique in a chordal graph G. Then M is nice if and only if one of the above scenarios yields a polar partition.* □

The final algorithm for polarity of chordal graphs is summarized below.

RECOGNIZING POLARITY
Input: A chordal graph G.
Task: Decide if G is polar.
Action:
(1) Test if G is unipolar.
(2) Test if G is monopolar.
(3) Test, for any non-adjacent vertices u and v, whether $N(u) \cap N(v)$ is an almost good clique.
(4) Test, for any maximal clique M, whether M is an almost nice clique.
If any test succeeds, G is polar; otherwise it is not polar.

The correctness of the above polarity recognition algorithm follows from Proposition 12. We now analyze its complexity. For simplicity we express everything in terms of just the number n of vertices. Our unipolarity recognition algorithm implicit in Proposition 1 yields complexity $O(n^3)$ for step 1. Our monopolarity recognition algorithm for step 2 has complexity $O(n^2)$ (see Theorem 10). In step 3, we test n^2 times a set for being an almost good clique, each consisting of $O(n)$ tests for being a good clique. According to Proposition 13 this amounts to testing monopolarity; in all, for step 3 we need at most time $O(n^2 \times n \times n^2)$. In step 4 we test at most n sets for being an almost nice clique, each consisting of $O(n)$ tests for being a nice clique. This test for a nice clique is described in detail above Proposition 14, and summarized below. It can be seen that this takes at most time $O(n^3)$ since there are only $O(n)$ possible choices for the clique C (either there are at most three vertices a in V_{K_i} or $K_i - a$ has only one component). Also we note that computing the sets V_{K_i} in time $O(n^3)$ is straightforward.

Hence one can implement the above algorithm in time $O(n^5)$. We have not attempted to make a more careful analysis, nor to improve the efficiency of the algorithm or its implementation. (For an improved version of this algorithm with a slightly better time complexity we refer the reader to [13].)

Theorem 15. *There is a polynomial time algorithm to test the polarity of chordal graphs.* □

As a concluding remark, we mention the following related open problem: given a chordal graph G which is not monopolar (respectively not polar) find the largest monopolar (respectively polar) induced subgraph of G.

TESTING WHETHER A CLIQUE IS NICE
Input: A clique M in a chordal graph G
Task: Decide if M is nice
Action:
- Find the components of $G - M$
- Compute the sets $V_{K_i}, i = 1, \ldots, h$
- For each choice of C (C is empty (1), or equal to some K_i (2) or a component of some $K_i - a, a \in V_{K_i}$ (3)),
  . colour a red (in case 3),
  . colour blue the rest of the exceptional components (in cases 2 and 3),
  . colour red the non-neighbours of any vertex of C in M,
  . colour red the maximum vertex of each non-exceptional component K which is not a clique,
  . colour blue all other vertices of each non-exceptional component K, and
  . apply Propagation Rules 1 and 2 as long as possible. If a vertex receives both colours, M is not nice.

Otherwise, colour all uncoloured vertices in M red, and all uncoloured vertices not in M blue.

Acknowledgements

We are thankful to two anonymous referees for their valuable suggestions that improved the presentation. The first author was supported by Grant 200021-101455/1 and Grant 200020-113405 of the Swiss National Science Foundation. The second and third authors were supported by a Discovery Grant from NSERC.

References

[1] Jason I. Brown, Derek G. Corneil, On generalized graph colorings, J. Graph Theory 11 (1987) 87–99.
[2] T. Ekim, N.V.R. Mahadev, D. de Werra, Polar cographs, Discrete Appl. Math., in press (doi:10.1016/j.dam.2007.08.025).
[3] Z.A. Chernyak, A.A. Chernyak, About recognizing (a, b)-classes of polar graphs, Discrete Math. 62 (1986) 133–138.
[4] T. Feder, P. Hell, S. Klein, R. Motwani, Complexity of graph partition problems, in: 31st Annual ACM STOC, 1999, pp. 464–472.
[5] T. Feder, P. Hell, S. Klein, L.T. Nogueira, F. Protti, List matrix partitions of chordal graphs, Theoret. Comput. Sci. 349 (2005) 52–66.
[6] T. Feder, P. Hell, Matrix partitions of perfect graphs, Discrete Math. 306 (2006) 2450–2460.
[7] A. Farrugia, Vertex-partitioning into fixed additive induced-hereditary properties is NP-hard, Electron. J. Combin. 11 (2004).
[8] A.V. Gagarin, Chordal $(1, \beta)$-polar graphs, Vestsi Nats. Akad. Navuk Belarusi Ser. Fiz.-Mat. Navuk 143 (1999) 115–118.
[9] M.C. Golumbic, Algorithmic Graph Theory and Perfect Graphs, Academic Press, New York, 1980.
[10] P. Hell, J. Nešetřil, Graphs and Homomorphisms, Oxford University Press, 2004.
[11] P. Hell, S. Klein, L.T. Nogueira, F. Protti, Partitioning chordal graphs into independent sets and cliques, Discrete Appl. Math. 141 (2004) 185–194.
[12] G. Rosenberg, Handbook of Graph Grammars and Computing by Graph Transformations, vol. 1, Foundations World Scientific, 1997.
[13] J. Stacho, Complexity of Generalized Colourings of Chordal Graphs, Ph.D. Thesis, Simon Fraser University, 2008.
[14] R.E. Tarjan, Depth first search and linear graph algorithms, SIAM J. Comput. 1 (1972) 146–160.
[15] R.E. Tarjan, M. Yannakakis, Simple linear time algorithms to test chordality of graphs, test acyclicity of hypergraphs, and selectively reduce acyclic hypergraphs, SIAM J. Comput. 13 (1984) 566–579.
[16] R.E. Tarjan, M.Y. Yannakakis, Addendum: Simple linear-time algorithms to test chordality of graphs, test acyclicity of hypergraphs, and selectively reduce acyclic hypergraphs, SIAM J. Comput. 14 (1985) 254–255.
[17] R.I. Tyshkevich, A.A. Chernyak, Algorithms for the canonical decomposition of a graph and recognizing polarity, Izvestia Akad. Nauk BSSR, ser. Fiz. Mat. Nauk. 6 (1985) 16–23 (in Russian).
[18] D. West, Introduction to Graph Theory, Prentice Hall, 1996.