Precoloring extension on chordal graphs

Dániel Marx*

17th August 2004

Abstract

In the precoloring extension problem (PrExt) we are given a graph with some of the vertices having a preassigned color and it has to be decided whether this coloring can be extended to a proper k-coloring of the graph. 1-PrExt is the special case where every color is assigned to at most one vertex in the precoloring. Answering an open question of Hujter and Tuza [HT96], we show that the 1-PrExt problem can be solved in polynomial time for chordal graphs.

1 Introduction

In graph vertex coloring we have to assign colors to the vertices such that neighboring vertices receive different colors. Starting with [ERT80] and [Viz76], a generalization of coloring was investigated: in the *list coloring* problem each vertex can receive a color only from its list of available colors. A special case is the *precoloring extension* problem: a subset W of the vertices have a preassigned color and we have to extend this to a proper coloring of the whole graph, using only colors from a color set C. It can be viewed as a special case of list coloring: the list of a precolored vertex consists of a single color, while the list of every other vertex is C. A thorough survey on list coloring, precoloring extension, and list chromatic number can be found in [Tuz97].

Since vertex coloring is the special case when $W = \emptyset$, the precoloring extension problem is NP-complete in every class of graphs where vertex coloring is NP-complete. Therefore we can hope to solve precoloring extension efficiently only on graphs that are easy to color. Biró, Hujter and Tuza [BHT92, HT93, HT96] started a systematic study of precoloring extension in perfect graphs, where coloring can be done in polynomial time. It turns out that for some classes of perfect graphs, e.g., split graphs, complements of bipartite graphs, and cographs, the precoloring extension problem can be solved in polynomial time. On the other hand, for some other classes like bipartite graphs, line graphs of bipartite graphs, and interval graphs, precoloring extension is NP-complete.

The d-PrExt problem is the restriction of the precoloring extension problem where every color is used at most d-times in the precoloring. It is easy to reduce PrExt

*Dept. of Computer Science and Information Theory, Budapest University of Technology and Economics (dmarx@cs.bme.hu). Research is supported in part by grants OTKA 44733, 42559 and 42706 of the Hungarian National Science Fund.

to 1-PrExt: collapse the vertices precolored with the same color to a single vertex. Therefore 1-PrExt is not easier than PrExt on classes of graphs that are closed for this operation. One can also show that 1-PrExt is NP-complete on bipartite graphs [HT93, BJW94].

However, there are cases where 1-PrExt is strictly easier than PrExt. For planar bipartite graphs, if the set of colors C contains only 3 colors, then PrExt is NP-complete [Kra93], while 1-PrExt can be solved in polynomial time [MTW98]. For interval graphs already 2-PrExt is NP-complete [BHT92], but 1-PrExt can be solved in polynomial time [BHT92]. For the special case of unit interval graphs PrExt remains NP-complete [Mar04], and obviously 1-PrExt remains polynomial-time solvable.

Every chordal graph is perfect and interval graphs form a subset of chordal graphs (cf. [Gol80]). Therefore by [BHT92], the 2-PrExt problem is NP-complete for chordal graphs. The complexity of 1-PrExt on chordal graphs is posed by Hujter and Tuza as an open question [HT96]. Here we show that 1-PrExt can be solved in polynomial time also for chordal graphs. The algorithm is a generalization of the method of [BHT92] for interval graphs. As in [BHT92], 1-PrExt is reduced to a network flow problem, but for chordal graphs a more elaborate construction is required than for interval graphs.

The paper is organized as follows. In Section 2 we review some known properties of chordal graphs. In Section 3 we define a set system that will be crucial in the analysis of the algorithm. The algorithm is presented in Section 4. In Section 5 we discuss some connections of the problem with matroid theory.

2 Tree decomposition

A graph is *chordal* if every cycle of length greater than 3 contains at least one chord, i.e., an edge connecting two vertices not adjacent in the cycle. Equivalently, a graph is chordal if and only if it does not contain a cycle of length greater than 3 as an induced subgraph. This section summarizes some well-known properties of chordal graphs. First, chordal graphs can be also characterized as the intersection graphs of subtrees of a tree (see e.g., [Gol80]):

Theorem 2.1. *The following two statements are equivalent:*

1. *$G(V, E)$ is chordal.*
2. *There exists a tree $T(U, F)$ and a subtree $T_v \subseteq T$ for each $v \in V$ such that $u, v \in V$ are neighbors in $G(V, E)$ if and only if $T_u \cap T_v \neq \emptyset$.*

The tree T together with the subtrees T_v is called the *tree decomposition* of G. Given a chordal graph G, a tree decomposition can be found in polynomial time (see [Gol80, RTL76]).

For clarity, we will use the word "vertex" when we refer to the graph $G(V, E)$, and "node" when referring to $T(U, F)$. We assume that T is a rooted tree with some root $r \in U$. For a node $x \in U$, let T^x be the subtree of T rooted at x. Consider those subtrees that contain at least one node of T^x, denote by V_x the corresponding vertices.

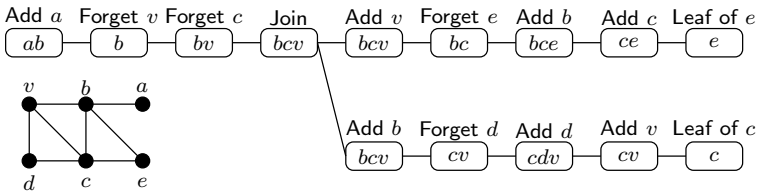

Figure 1: Nice tree decomposition of a chordal graph.

The subgraph of G induced by V_x will be denoted by $G_x = G[V_x]$. For a node $x \in U$ of T, denote by K_x the union of v's where $x \in T_v$. Clearly, the vertices of K_x are in V_x, and they form a clique in G_x, since the corresponding trees intersect in T at node x. An important property of the tree decomposition is the following: for every node $x \in U$, the clique K_x separates $V_x \setminus K_x$ and $V \setminus V_x$. That is, among the vertices of V_x, only the vertices in K_x can be adjacent to $V \setminus V_x$.

Every inclusion-wise maximal clique of a chordal graph G is a clique K_x of the tree decomposition. This is a consequence of the fact that subtrees of a tree satisfy the Helly property (a family of sets is said to satisfy the Helly property if for each pairwise intersecting collection of sets from the family it follows that the sets in the collection have a common element). If K is a clique of G, then its vertices correspond to pairwise intersecting subtrees, hence by the Helly property, these trees have a common node x, implying $K \subseteq K_x$. Since every chordal graph is perfect, the chromatic number of G equals its clique number, and it follows that G is k-colorable if and only if $|K_x| \leq k$ for every node $x \in T$. Clearly, the precoloring can exist only if G is $|C|$-colorable, hence we assume in the following that $|K_x| \leq |C|$ holds for every $x \in T$.

A tree decomposition will be called *nice* [Klo94], if it satisfies the following additional requirements (see Figure 1):

- Every node $x \in U$ has at most two children.
- If $x \in U$ has two children $y, z \in U$, then $K_x = K_y = K_z$.
- If $x \in U$ has only one child $y \in U$, then either $K_x = K_y \cup \{v\}$ or $K_x = K_y \setminus \{v\}$ for some $v \in V$.
- If $x \in U$ has no children, then K_x contains exactly one vertex.

It is easy to see that by splitting the nodes of the tree in an appropriate way, a tree decomposition of G can be transformed into a nice tree decomposition in polynomial time. In such a decomposition, the nodes are of the following types:

- A node x with two children is a *join* node.
- A node x with one child y and $K_x = K_y \setminus \{v\}$ is a *forget* node (of v).
- A node x with one child y and $K_x = K_y \cup \{v\}$ is an *add* node (of v).
- A node x without children is a *leaf* node.

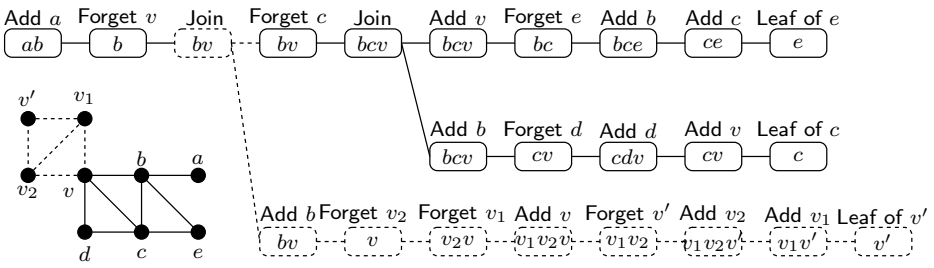

Figure 2: Nice tree decomposition of the graph shown on Figure 1, after adding the vertices v', v_1, v_2 to the graph. Dashed lines show the new parts of the tree decomposition.

A vertex v can have multiple add nodes, but at most one forget node (the vertices in clique K_r of the root r have no forget nodes, but every other vertex has exactly one). For a vertex v, its subtree T_v is the subtree rooted at the forget node of v (if it exists, otherwise at the root) and whose leaves are exactly the add nodes and leaf nodes of v.

Given a graph G and a precolored set of vertices, we slightly modify the graph to obtain an even nicer tree decomposition. For each precolored vertex v, we add a clique K of $|C|-1$ new vertices to the graph, each vertex of K is connected to v; and we also add a new vertex v' that is connected to each vertex of K (but not to v). The precoloring of vertex v is removed and v' becomes a precolored vertex, the color of v is assigned to v'. It is easy to see that this transformation does not change the solvability of the instance: vertices v and v' receive the same color in every $|C|$-coloring of the new graph G' (since they are both connected to the same clique of $|C|-1$ vertices), thus a precoloring extension of G' induces a precoloring extension for G. Although the transformation increases the size of the graph and hence the size of the tree decomposition, it will be very useful, since now we can assume that the nice tree decomposition has the following additional properties:

- If $x \in U$ is the add node of v, then v is not a precolored vertex.
- If $x \in U$ is a join node, then K_x does not contain precolored vertices.

We show how a nice tree decomposition T of G can be modified to obtain a nice tree decomposition T' of G' satisfying these two additional properties. Let v_1, v_2, …, $v_{|C|-1}$ be the neighbors of v' in G'. Let x be an arbitrary node containing vertex v, let $K_x = \{v, w_1, w_2, \ldots, w_t\}$. Insert a new join node between x and its parent, we attach a new branch to the parent of x, which will describe the vertices v', v_1, …, $v_{|C|-1}$. This new branch is a path, containing the following nodes (see Figure 2):

- Leaf node containing v'.
- Add node of v_1, add node of v_2, …, add node of $v_{|C|-1}$.
- Forget node of v'.

- Add node of v.
- Forget node of v_1, forget node of v_2, ..., forget node of $v_{|C|-1}$.
- Add node of w_1, add node of w_2, ..., add node of w_t.

It is clear that this modification results in a nice tree decomposition, and if we perform it for each precolored vertex v, then we obtain a decomposition of G'.

3 System of extensions

Let H be an induced subgraph of G, and let K be a clique of H. We define a set system $\mathscr{S}(H,K)$ over K that will play an important role in the analysis of the algorithm. Denote by $C_H \subseteq C$ those colors that the precoloring assigns to vertices in H. The set system $\mathscr{S}(H,K)$ is defined as follows:

Definition 3.1. *For $S \subseteq K$, the set S is in $\mathscr{S}(H,K)$ if and only if there is a precoloring extension $\psi\colon V(H) \to C$ of subgraph such that*

- *$\psi(v) \in C_H$ for every $v \in S$, and*
- *$\psi(v) \notin C_H$ for every $v \in K \setminus S$.*

Thus the set system $\mathscr{S}(H,K)$ describes all the possible colorings that can appear on K in a precoloring extension of H, but this description only distinguishes between colors in C_H and colors not in C_H. In particular, the precoloring can be extended to H if and only if $\mathscr{S}(H,K)$ is not empty. If H contains no precolored vertices, but it can be colored with $|C|$ colors, then $\mathscr{S}(H,K)$ contains only the empty set.

The following observation bounds the possible size of a set in $\mathscr{S}(H,K)$:

Observation 3.2. *If $S \in \mathscr{S}(H,K)$, then*

$$|K| - |C \setminus C_H| \leq |S| \leq |C_H|$$

Proof. If $S \in \mathscr{S}(H,K)$, then there is a coloring ψ that assigns exactly $|S|$ colors from C_H to the vertices of K. Clearly, in ψ at most $|C_H|$ vertices of the clique K can receive colors from C_H, proving the upper bound. Coloring ψ assigns colors from $C \setminus C_H$ to the vertices in $K \setminus S$, hence $|C \setminus C_H| \geq |K| - |S|$, and the lower bound follows. ■

The definition of this set system might seem somewhat technical, but it precisely captures the information necessary for solving the precoloring extension problem. Let K be a *clique separator* of G, that is, K is a clique such that its removal separates the graph into two or more components. Let $V \setminus K = V_1 \cup V_2$ be a partition of the remaining vertices such that there is no edge between V_1 and V_2 (that is, each of V_1 and V_2 contains one or more connected components of $V \setminus K$). Let $G_1 = G[V_1 \cup K]$ and $G_2 = G[V_2 \cup K]$. Assume that we have already extended the precoloring to G_1 (coloring ψ_1) and to G_2 (coloring ψ_2). If $\psi_1(v) = \psi_2(v)$ for every vertex v of the clique K, then they can be merged to obtain a coloring of G. Therefore G has a precoloring extension if and only if there is a precoloring extension ψ_1 of G_1, and a precoloring

extension ψ_2 of G_2 such that they agree on K. This means that if we have the list of all possible colorings that a precoloring extension of G_1 can assign to K, then to decide if G has a precoloring extension this list is all the information required from the graph G_1. More formally, if we replace G_1 with a graph that has the same list of possible colorings on K, then this does not change the existence of a precoloring extension on G.

However, the following lemma shows that even less information is sufficient: we do not need the list of all possible colorings that can appear on clique K in a coloring of G_1, the set system $\mathscr{S}(G_1, K)$ is enough. More precisely, the set system $\mathscr{S}(G, K)$ can be constructed from $\mathscr{S}(G_1, K)$ and $\mathscr{S}(G_2, K)$, hence these two systems are sufficient to decide whether G has a precoloring extension.

Lemma 3.3. *Let K be a clique separator of $G(V_1 \cup K \cup V_2, E)$ containing no precolored vertices, let $G_1 = G[V_1 \cup K]$ and $G_2 = G[V_2 \cup K]$. A set $S \subseteq K$ is in $\mathscr{S}(G, K)$ if and only if $|S| \geq |K| - |C \setminus C_G|$ and S can be partitioned into disjoint sets $S_1 \in \mathscr{S}(G_1, K)$ and $S_2 \in \mathscr{S}(G_2, K)$.*

Proof. Assume first that $S \in \mathscr{S}(G, K)$ and let ψ be a coloring corresponding to the set S. Observation 3.2 implies that $|S| \geq |K| - |C \setminus C_G|$, as required. Coloring ψ induces a coloring ψ_i of G_i, let $S_i \in \mathscr{S}(G_i, K)$ be the set corresponding to ψ_i $(i = 1, 2)$. Coloring ψ can assign three different types of colors to the vertices in K:

- If $\psi(v) \notin C_G$ (i.e., $\psi(v)$ is not used in the precoloring), then $v \notin S, S_1, S_2$.
- If the precoloring uses $\psi(v)$ in V_1, then $v \in (S \cap S_1) \setminus S_2$. (Since each color is used at most once in the precoloring, $\psi(v)$ cannot appear in V_2 on a precolored vertex.)
- If the precoloring uses $\psi(v)$ in V_2, then $v \in (S \cap S_2) \setminus S_1$.

Note that v cannot be a precolored vertex, hence the precoloring cannot use $\psi(v)$ in K. Therefore S is the disjoint union of S_1 and S_2, as required.

Now assume that S can be partitioned into disjoint sets $S_1 \in \mathscr{S}(G_1, K)$ and $S_2 \in \mathscr{S}(G_2, K)$, let ψ_1 and ψ_2 be the two corresponding colorings. In general, ψ_1 and ψ_2 might be different on K, thus they cannot be combined to obtain a coloring of G. However, with some permutations of colors we modify the two colorings in such a way that they assign the same color to every vertex of K. Let C_1 (resp. C_2) be the colors of the precolored vertices in V_1 (resp. V_2). Notice that both ψ_1 and ψ_2 assign colors from $C \setminus C_1$ to S_2, (since S_1 and S_2 are disjoint). Modify coloring ψ_1: permute the colors of $C \setminus C_1$ such that $\psi_1(v) = \psi_2(v)$ holds for every $v \in S_2$ (this can be done since K is a clique, hence both ψ_1 and ψ_2 assign distinct colors to the vertices in S_2). Since the precolored vertices in V_1 have colors only from C_1, coloring ψ_1 remains a valid precoloring extension for G_1. Similarly, in coloring ψ_2, permute the colors of $C \setminus C_2$ such that $\psi_1(v) = \psi_2(v)$ for every $v \in S_1$. Now we have that ψ_1 and ψ_2 agree on S, there might be differences only on $K \setminus S$. Moreover, ψ_1 uses only colors from $C \setminus C_1$ on $K \setminus S$, and ψ_2 uses colors only from $C \setminus C_2$ on this set. Now select a set $C' \subseteq C \setminus C_G$ such that $|C'| = |K \setminus S|$ (here we use the assumption $|S| \geq |K| - |C \setminus C_G|$, which implies that there are enough colors in $C \setminus C_G$). Permute again the colors of

$C \setminus C_1$ in coloring ψ_1 such that ψ_1 assigns to $K \setminus S$ exactly the colors in C'. Similarly, permute the colors of $C \setminus C_2$ in coloring ψ_2 such that ψ_2 also uses C' on $K \setminus S$. Now the colorings ψ_1 and ψ_2 agree on K, hence we can combine them to obtain a coloring ψ of G. This coloring proves that $S = S_1 \cup S_2$ is in $\mathscr{S}(G, K)$, what we had to show. ■

Lemma 3.3 implies that if we know the set systems $\mathscr{S}(G_1, K)$ and $\mathscr{S}(G_2, K)$, then the set system $\mathscr{S}(G, K)$ can be also determined. This suggests the following algorithm: for each node x of the tree decomposition, determine $\mathscr{S}(G_x, K_x)$. In principle, this can be done in a bottom-up fashion: the set system for node x can be determined from the systems of its children. Unfortunately, the size of $\mathscr{S}(G_x, K_x)$ can be exponential, thus it cannot be constructed explicitly during the algorithm. However, if G is a chordal graph, then these set systems have nice combinatorial structure that allows a compact representation. The main idea of the algorithm in Section 4 is to use network flows to represent the set systems $\mathscr{S}(G_x, K_x)$. In Section 5 we discuss formally what is this nice structure that makes possible the representation with flows: it turns out that if G is chordal and K is a clique of G, then $\mathscr{S}(G, K)$ is the projection of a matroid.

4 The algorithm

In this section we prove the main result of the paper:

Theorem 4.1. 1-PrExt *can be solved in polynomial time for chordal graphs.*

Given an instance of the 1-PrExt problem, we construct a network flow problem that has a feasible flow if and only if there is a solution to 1-PrExt. We use the following variant of the flow problem. The *network* is a directed graph $D(U, A)$, each arc $e \in A$ has an integer capacity $c(e)$. The set of arcs entering (resp. leaving) node v will be denoted by $\delta^-(v)$ (resp. $\delta^+(v)$). The set of *sources* is $S \subseteq U$, and $T \subseteq U$ is the set of *terminals* in the network (we require $S \cap T = \emptyset$). Every source $v \in S$ produces exactly one unit amount of flow, and every terminal $v \in T$ has a capacity $w(v)$, it can consume up to $w(v)$ units. Formally, a *feasible flow* is a function $f\colon A \to \mathbb{Z}^+$ that satisfies $0 \leq f(e) \leq c(e)$ for every arc $e \in A$, and the following holds for every node $v \in U$:

- If $v \in S$, then $\sum_{e \in \delta^-(v)} f(e) - \sum_{e \in \delta^+(v)} f(e) = -1$.
- If $v \in T$, then $0 \leq \sum_{e \in \delta^-(v)} f(e) - \sum_{e \in \delta^+(v)} f(e) \leq w(v)$.
- If $v \in U \setminus (T \cup S)$, then $\sum_{e \in \delta^-(v)} f(e) = \sum_{e \in \delta^+(v)} f(e)$.

Using standard techniques, the existence of a feasible flow can be tested by a maximum flow algorithm. It is sufficient to add two new vertices s and t, an arc with capacity 1 from s to every vertex $v \in S$, and an arc with capacity $w(v)$ to t from every vertex $v \in T$. Clearly, there is a feasible flow in the original network if and only if there is an $\overrightarrow{st}$ flow with value $|S|$ in the modified network. The maximum flow can be determined using at most $|S|$ iterations of the Edmonds-Karp augmenting path algorithm, hence the existence of a feasible flow in a network $D(U, A)$ can be tested in $O(|S||A|)$ time.

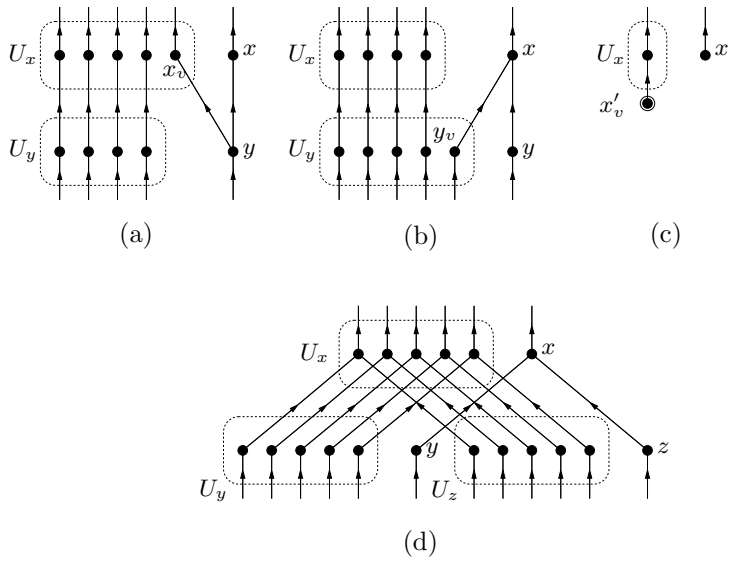

Figure 3: Construction of the network when node x is of the following types: (a) add node for $v \notin W$, (b) forget node for v (either in W or not), (c) leaf node for $v \in W$, (d) join node.

Given a chordal graph $G(V, E)$, its nice tree decomposition $T(U, F)$, $\{T_v \mid v \in V\}$, and the set of precolored vertices $W \subseteq V$, we construct a network as follows. Direct every edge of T towards the root r. For every $v \in V$ and for every $x \in T_v$ add a node x_v to the network. Denote by U_x the $|K_x|$ nodes corresponding to x. If the edge xy is in T_v, then connect $x_v \in U_x$ and $y_v \in U_y$ by an arc. If y is the child of x, then direct this arc from y_v to x_v. These new arcs $\overrightarrow{y_v x_v}$ have capacity 1, while the arcs $\overrightarrow{yx}$ of the tree T have capacity $|C| - |K_y|$ (recall that if the graph is $|C|$-colorable, then $|K_y| \leq |C|$).

For each node $x \in T$, depending on the type of x, we do one of the following:

- If x is an add node of some vertex $v \notin W$, and y is the child of x, then add an arc $\overrightarrow{yx_v}$ to the network.
- If leaf node x contains some vertex $v \in W$, then add a new node x'_v to the network, add an arc $\overrightarrow{x'_v x_v}$ with capacity 1, and set x'_v to be a source.
- If x is a forget node of some vertex v (either in W or not), and y is the child of x, then add an arc $\overrightarrow{y_v x}$ to the network.

For join nodes and for leaf nodes containing vertices outside W we do nothing. Figure 3 sketches the construction for the different types of nodes.

So far there are no terminals in the network. The definition of the network is completed by adding terminals as follows. Here we define not only a single network, but several subnetworks that will be useful in the analysis of the algorithm. For every

node $x \in U$ of the tree T, the network N_x contains only those nodes of the network that correspond to nodes in T^x (recall that T^x is the subtree of T rooted at x). Formally, the network N_x has the node set $T^x \cup \bigcup_{y \in T^x} U_y$, and the sources nodes (if available) corresponding to the leaves of T^x. Moreover, in network N_x the nodes in U_x are set to be terminals with capacity 1, and node x is a terminal with capacity $|C| - |K_x|$. This completes the description of the network N_x.

Notice that there are sources only at the leaf nodes of precolored vertices. Therefore the number of sources in network N_x is the same as the number of precolored vertices in V_x (recall that V_x is the set of those vertices v whose tree T_v has at least one node in T^x, and $G_x = G[V_x]$). We will denote by C_x the set of colors that appear on the precolored vertices of V_x. In network N_x, there are terminals only at x and U_x, these terminals must consume all the flow.

Observation 4.2. *The number of sources in N_x equals the number of precolored vertices in V_x, which is $|C_x|$. Consequently, in every feasible flow of N_x, the amount of flow consumed by the terminals at x and U_x is exactly $|C_x|$.* ■

To prove Theorem 4.1, we show that the precoloring of G can be extended to the whole graph if and only if there is a feasible flow in N_r, where r is the root of T. This gives a polynomial-time algorithm for 1-PrExt in chordal graphs, since constructing network N_r and finding a feasible flow in N_r can be done in polynomial time. The proof of this claim uses induction on the tree decomposition of the graph. For every node $x \in U$ of T, we prove the following more general statement: the network N_x has a feasible flow if and only if the precoloring of G can be extended to G_x.

More precisely, we show that the network N_x represents (in some well-defined sense) the set system $\mathscr{S}(G_x, K_x)$: every feasible flow corresponds to a set in the system. Therefore N_x has no feasible flows if and only if $\mathscr{S}(G_x, K_x)$ is empty, or, equivalently, the precoloring cannot be extended to G_x.

We say that a feasible integer flow of N_x *represents the set* $S \subseteq K_x$ if for every $v \in S$, the terminal at x_v consumes one unit of flow, while for every $v \notin S$, there is no flow entering x_v. The following lemma establishes the connection between the constructed networks and the set systems $\mathscr{S}(G_x, K_x)$. The proof of this lemma completes the proof of Theorem 4.1, as it reduces the 1-PrExt problem to finding a feasible flow in N_r.

Lemma 4.3. *For an arbitrary node $x \in U$ of T, the network N_x has a feasible flow representing a set $S \subseteq K_x$ if and only if $S \in \mathscr{S}(G_x, K_x)$.*

Proof. The lemma is proved for every node x of T by a bottom-up induction on the tree T. After checking the lemma for the leaf nodes, we show that it is true for a node x assuming that it is true for the children of x. The proof is done separately for the different types of nodes. Verifying the lemma in each case is tedious, but it does not require any new ideas. The way the networks are constructed ensures that the set systems represented by the networks have the required properties.

Leaf node. For a leaf node x, the lemma is trivial: if the vertex v in K_x is precolored, then every flow of N_x represents $\{v\}$, otherwise N_x contains no sources, and every flow represents $\emptyset$.

Add node for $v \notin W$. Let x be an add node of $v \notin W$, and let y be the child of x. For every $S \in \mathscr{S}(G_x, K_x)$, it has to be shown that there is a feasible flow of N_x representing the set S. Assume first that $v \notin S$. Since $G_y = G_x \setminus v$ and $K_y = K_x \setminus \{v\}$, it follows that $S \in \mathscr{S}(G_y, K_y)$. Therefore by the induction hypothesis, there is a flow f_y in N_y representing S. We modify this flow to obtain a flow f_x of N_x also representing S. For every $u \in S$, in flow f_y there is one unit of flow consumed by the terminal at y_u. To obtain flow f_x, direct this unit flow towards x_u, and consume it by the terminal at that node. Similarly, in the flow f_y, there is some amount of flow consumed by the terminal at y, direct this flow to x, and consume it by that terminal. By Observation 4.2, the amount of flow consumed at x is exactly $|C_x| - |S|$. Moreover, the lower bound of Observation 3.2 implies that this is at most $|C_x| - |K_x| + |C \setminus C_x| = |C| - |K_x| < |C| - |K_y|$, hence the capacity of the arc $\overrightarrow{yx}$ and the terminal at x is sufficient for the flow. Thus we obtained a feasible flow of N_x, and obviously it represents S.

We proceed similarly if $v \in S$. In this case $S \setminus \{v\} \in \mathscr{S}(G_y, K_y)$, thus N_y has a flow f_y representing $S \setminus \{v\}$. To obtain a flow f_x of N_x representing S, the flow consumed at y_u is directed to x_u, as in the previous paragraph. However, now we do not direct all the flow consumed at y to x, but we direct one unit amount through the arc $\overrightarrow{yx_v}$, and only the rest goes through arc $\overrightarrow{yx}$. Therefore the amount of flow consumed by the terminal at x is one unit less than the flow consumed at y in flow f_y, hence the capacity of the terminal at x is sufficient. Clearly, this results in a flow f_x of N_x representing S, as required. The only thing to verify is that there is at least one unit of flow consumed at y in flow f_y. The flow f_y represents $S \setminus \{v\}$, and by Observation 4.2, the amount of flow consumed in $U_y \cup \{y\}$ is exactly $|C_y|$, hence the flow consumed at y is $|C_y| - |S| + 1$. Since v is not a precolored vertex, we have that $C_y = C_x$. We know that $S \in \mathscr{S}(G_x, K_x)$, therefore by the upper bound of Observation 3.2, $|C_y| - |S| + 1 \geq 1$, hence there is nonzero flow consumed at y in flow f_y.

Now assume that there is a flow f_x in N_x representing $S \subseteq K_x$, it has to be shown that $S \in \mathscr{S}(G_x, K_x)$. Let y be the child of x. Assume first that $v \notin S$, we show that N_y has a flow f_y in N_y representing S. To obtain this f_y, the flow f_x is modified the following way. For every vertex $x_w \in U_x$, where $w \neq v$, if there is flow on the arc $\overrightarrow{y_w x_w}$, then consume it by the terminal at y_w. Similarly, the flow on the arc $\overrightarrow{yx}$ can be consumed by the terminal at y (the capacity of the terminal at y equals the capacity of arc $\overrightarrow{yx}$). It is clear that these modifications result in a feasible flow for N_y that represents S. By the induction hypothesis, this means that $S \in \mathscr{S}(G_y, K_y)$, and there is a corresponding coloring ψ. Since v is the only vertex in $V_x \setminus V_y$, to prove $S \in \mathscr{S}(G_x, K_x)$ it is sufficient to show that coloring ψ can be extended to v in such a way that v receives a color not in C_x. If there is no such extension, then this means that ψ uses every color of $C \setminus C_x$ on the neighbors of v, that is, on the clique K_y. By construction, ψ assigns exactly $|S|$ colors from C_x to the clique K_y, hence if every color of $C \setminus C_x$ is used on K_y, then $|K_y| = |C \setminus C_x| + |S|$. Therefore $|K_x| = |K_y| + 1 = |C \setminus C_x| + |S| + 1$ and the capacity of the terminal at x is $|C| - |K_x| = |C_x| - |S| - 1$. However, in flow f_x of N_x that represents S, exactly $|C_x| - |S|$ unit of flow is consumed at x (Observation 4.2), a contradiction. Thus ψ can be extended to v, and $S \in \mathscr{S}(G_x, K_x)$ follows.

The case $v \in S$ can be handled similarly. If there is a flow f_x in N_x that represents S, then this is only possible if there is flow on the arc $\overrightarrow{yx_v}$. Therefore by restricting the flow to N_y as in the previous paragraph, we can obtain a flow representing $S \setminus \{v\}$. (Notice that the capacity of the terminal at y equals the combined capacity of the terminals at x and x_v, hence it can consume the flow on the arcs $\overrightarrow{yx_v}$ and $\overrightarrow{yx}$.)

By the induction hypothesis, it follows that $S \setminus \{v\} \in \mathscr{S}(G_y, K_y)$, and there is a corresponding coloring ψ. Now it has to be shown that ψ can be extended to vertex v such that v receives a color from C_x. Coloring ψ assigns exactly $|S| - 1$ colors from C_x to K_y. The extension is not possible only in the case if every color of C_x is already used on K_y, that is, if $|C_x| = |S| - 1$. This would imply that in flow f_x of N_x, the amount of flow consumed at U_x is $|S| = |C_x| + 1$. However, by Observation 4.2, this is strictly larger than the number of sources in N_x, a contradiction.

Forget node for v (vertex v is either in W or not). Let x be the forget node of v, and let y be the child of x. Let $S \in \mathscr{S}(G_x, K_x)$, it has to be shown that there is a feasible flow f_x of N_x representing S. Since $G_x = G_y$, if $S \in \mathscr{S}(G_x, K_x)$, then either S or $S \cup \{v\}$ is in $\mathscr{S}(G_y, K_y)$. In the first case, if $S \in \mathscr{S}(G_y, K_y)$, then the flow f_y in N_y that represents S can be extended to a flow in N_x that also represents S. As before, the flow consumed at y_w is directed to x_w, and the flow consumed at y is directed to x. Recall that the capacity of the arc $\overrightarrow{yx}$ equals the capacity of the terminal at y, while the capacity of the terminal at x is strictly greater. Therefore the resulting flow is feasible in N_x, and clearly it represents S. If $S \cup \{v\} \in \mathscr{S}(G_y, K_y)$, then we do the same, but the flow consumed at y_v is directed to x through the arc $\overrightarrow{y_v x}$. The resulting flow is feasible in N_x and represents S.

To prove the other direction, assume that N_x has a flow f_x representing $S \subseteq K_x$. Restrict this flow to N_y, that is, modify the flow such that the terminals at y and U_y consume all the flow. This results in a feasible flow f_y of N_y that represents S or $S \cup \{v\}$. Notice that the terminal at y has the same capacity as the arc $\overrightarrow{yx}$, hence this terminal can consume all the flow going through the arc. Therefore, by the induction hypothesis, either S or $S \cup \{v\}$ is in $\mathscr{S}(G_y, K_y)$ (depending on whether there is flow consumed at y_v or not). In either case, $S \in \mathscr{S}(G_x, K_y)$ follows since $G_x = G_y$ and $K_x = K_y \setminus \{v\}$.

Join node. Let y and z be the two children of the join node x. Let $S \in \mathscr{S}(G_x, K_x)$, by Lemma 3.3 this means that

$$|S| \geq |K_x| - |C \setminus C_x| \tag{1}$$

and S can be partitioned into disjoint sets $S_1 \in \mathscr{S}(G_y, K_y)$ and $S_2 \in \mathscr{S}(G_z, K_z)$. By the induction hypothesis, this implies that there are flows f_y, f_z in N_y and N_z that represent the sets S_1 and S_2, respectively. We combine these two flows to obtain a flow f_x of N_x that represents the set S. If there is flow consumed at a node $y_v \in U_y$ (resp. $z_v \in U_z$) in f_y (resp. f_z), then direct this flow on the arc $\overrightarrow{y_v x_v}$ (resp. $\overrightarrow{z_v x_v}$) to node x_v, and consume it there. The capacity of the terminal at x_v is 1, but the disjointness of S_1 and S_2 implies that at most one unit of flow is directed to x_v. The flow consumed at node y and z is directed to x on the arc $\overrightarrow{yx}$, $\overrightarrow{zx}$, respectively. Since there are exactly $|S|$ units of flow consumed in U_x, therefore $|C_x| - |S|$ units of flow has to be consumed at x. By (1), this is at most $|C_x| - |K_x| + |C \setminus C_x| = |C| - |K_x|$,

thus the capacity of the terminal at x is sufficient for consuming this flow. Therefore we have obtained a flow f_x in network N_x that represents S.

Now assume that N_x has a flow f_x that represents S. Since the terminal at x has capacity at most $|C| - |K_x|$, and by Observation 4.2, the amount of flow consumed in $x \cup U_x$ is $|C_x|$, it follows that

$$|S| \geq |C_x| - (|C| - |K_x|) = |K_x| - |C \setminus C_x|. \tag{2}$$

If flow is consumed at a node $x_v \in U_x$, then the flow arrives to this node either from y_v or from z_v. Define the sets $S_1, S_2 \subseteq K_x$ such that $v \in S_1$ (resp. $v \in S_2$) if there is flow on arc $\overrightarrow{y_v x_v}$ (resp. $\overrightarrow{z_v x_v}$).

Based on the flow f_x of N_x representing S, we create a flow f_y of N_y that represents S_1 and a flow f_z of N_z that represents S_2. The flows f_y and f_z are constructed as follows. For every $y_v \in U_y$, if there is flow going through the arc $\overrightarrow{y_v x_v}$, then consume this flow at y_v, and similarly for the nodes $z_v \in U_z$. The flow on arcs $\overrightarrow{yx}$ and $\overrightarrow{zx}$ are consumed at y and z, respectively (the capacity of nodes x, y, and z are the same $|C| - |K_x| = |C| - |K_y| = |C| - |K_z|$). Clearly, flows f_y and f_z represent S_1 and S_2, respectively. By the induction hypothesis, the flows f_x and f_y imply that $S_1 \in \mathscr{S}(G_y, K_y)$ and $S_2 \in \mathscr{S}(G_z, K_z)$. Furthermore, it is clear that S_1 and S_2 are disjoint, and $S = S_1 \cup S_2$. Therefore by Lemma 3.3 and Inequality (2), this proves that $S \in \mathscr{S}(G_x, K_x)$, as required.

■

To determine the running time of the algorithm, we have to consider two main steps: the construction of the network and the solution of the flow problem. First of all, the transformation introduced at the end of Section 2 can increase the size of the graph by at most a factor of n. Given a chordal graph $G(V, E)$, its tree decomposition can be constructed by first finding a perfect vertex elimination scheme [Gol80, RTL76]. Based on this ordering of the vertices, one can build a tree $T(U, F)$ of size $|V|$, and one subtree for each vertex of the graph. This tree decomposition can be found in time linear in the size of the output, that is, in $O(|V|^2)$ time. Converting $T(U, F)$ to a nice tree decomposition can introduce an increase of factor at most $|V|$, thus it can be done in $O(|V|^3)$ time. The network defined by the algorithm has size linear in the total size of the tree decomposition (size of $T(U, F)$ and the sum of the size of the subtrees), and clearly it can be constructed in linear time. Therefore the constructed network has $O(|V|^3)$ nodes and $O(|V|^3)$ arcs, and the construction takes $O(|V|^3)$ time.

In a network with n nodes and m arcs, the maximum flow can be determined in $O(n^2 m)$ or even in $O(n^3)$ time [AMO93]. Moreover, it can be determined in $O(km)$ time if a flow with value k exists: the Edmonds-Karp algorithm produces such a flow after finding the first at most k augmenting paths (assuming that the capacities are integer). As discussed in the beginning of the section, the existence of a feasible flow can be tested by finding an s-t flow with value $|S|$, hence it can be done in $O(|S| \cdot |V|^3)$ time. By Observation 4.2, this is at most $O(|C| \cdot |V|^3) = O(|V|^4)$. We believe that the running time can be significantly improved by streamlining the construction. However, our aim was only to prove that the problem can be solved in polynomial time, thus we preferred ease of presentation over efficiently.

The algorithm described above determines whether a precoloring extension exists, but does not find a coloring. However, based on the feasible flow of network N_r, one can construct a precoloring extension of the graph. We have seen that the feasible flow of network N_x represents a set $S_x \in \mathscr{S}(G_x, K_x)$. Recursively for each $x \in U$, we compute a coloring ψ_x corresponding to S_x. For the leaf nodes this is trivial. Let x be an add node of vertex v, and let y be the child of x. To obtain ψ_x, coloring ψ_y has to be extended to v: if there is flow on $\overrightarrow{yx_v}$, then v has to receive a color from C_x, otherwise from $C \setminus C_x$. The construction ensures that there is always such a color not already used on the neighbors of v. If x is a forget node with child y, then ψ_x can be selected to be the same as ψ_y. Finally, assume that x is a join node with children y and z. By the way the network was constructed, S_y and S_z are disjoint, $S_x = S_y \cup S_z$, and $S \geq |K_x| - |C \setminus C_x|$. Therefore the method described in the proof of Lemma 3.3 can be used to construct a coloring ψ_x of G_x that corresponds to $S_x \in \mathscr{S}(G_x, K_x)$.

5 Matroidal systems

The main idea of the algorithm in Section 4 is to represent the set system $\mathscr{S}(G, K)$ by a network flow. We have shown that for chordal graphs the set systems $\mathscr{S}(G_x, K_x)$ can be represented by network flows for every subgraph G_x and clique K_x given by the tree decomposition. The reason why these systems can be represented by flows is that they have nice combinatorial structure (the proof is given at the end of the section):

Theorem 5.1. *Let $G(V, E)$ be a chordal graph, and let $W \subseteq V$ be a arbitrary set of precolored vertices such that every color of C is used at most once in the precoloring. If H is an induced subgraph of G, and K is a clique of H, then the set system $\mathscr{S}(H, K)$ is the projection of the basis set of a matroid.*

A set system $\mathscr{B}$ is the basis set of a *matroid*, if it satisfies the following two conditions:

- Every set in $\mathscr{B}$ has the same size.
- For every $B_1, B_2 \in \mathscr{B}$ and $v \in B_1 \setminus B_2$, there is an element $u \in B_2 \setminus B_1$ such that $B_1 \cup \{u\} \setminus \{v\} \in \mathscr{B}$.

If $\mathscr{B}$ is a set system over X, then its *projection* to $Y \subseteq X$ is a set system over Y that contains $Y' \subseteq Y$ if and only if there is a set $B \in \mathscr{B}$ with $B \cap Y = Y'$. The projection of a matroid is always a so-called Δ-matroid [Mur00], hence Theorem 5.1 also says that $\mathscr{S}(G, K)$ is a Δ-matroid. For further notions of matroid theory, the reader is referred to e.g., [Rec89].

In general, if G is not chordal, then $\mathscr{S}(G, K)$ is not necessarily the projection of a matroid. Figure 4 shows a non-chordal graph G with two precolored vertices v_1 and v_2. The graph is not chordal, since vertices v_1, v_4, v_2, v_8 induce a cycle of length 4. If we have only four colors, then G has four precoloring extensions: vertex v_3 can have only color 3 or 4, vertex v_9 can have only color 1 or 2, and setting the color of these two vertices forces a unique coloring for the rest of the graph. For example, if coloring ψ assigns color 3 to v_3, and color 1 to v_9, then $\psi(v_9) = 1$,

$\psi(v_2) = 2$, $\psi(v_4) = 4$ imply $\psi(v_6) = 3$; furthermore $\psi(v_1) = 1$, $\psi(v_2) = 2$, $\psi(v_6) = 3$ imply $\psi(v_8) = 4$; and finally $\psi(v_5) = 1$ and $\psi(v_7) = 2$ follow in a similar fashion. Therefore the clique $K = \{v_5, v_6, v_7\}$ receives one of the four colorings $(3, 1, 4)$, $(1, 3, 2)$, $(4, 1, 3)$, $(1, 4, 2)$ in every precoloring extension. Since $C_G = \{1, 2\}$, it follows that $\mathscr{S}(G, K) = \{\{v_6\}, \{v_5, v_7\}\}$, which cannot be the projection of a matroid (for example, it is not even a Δ-matroid).

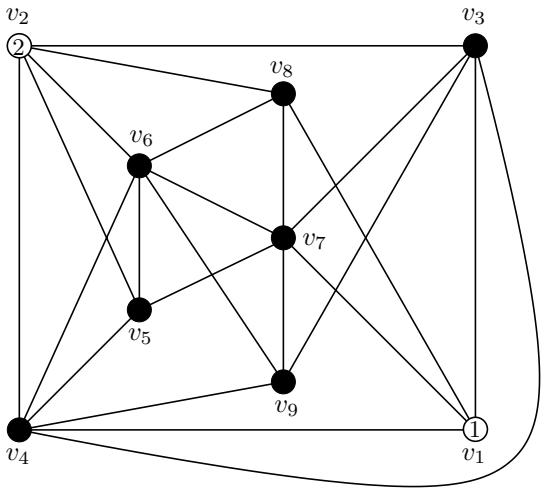

Figure 4: A non-chordal graph G and a clique $K = \{v_5, v_6, v_7\}$ such that $\mathscr{S}(G, K)$ is not the projection of a matroid ($|C| = 4$).

The proof of Theorem 5.1 uses the following result of matroid theory. In a directed graph $D(U, A)$, we say that $Y \subseteq U$ can be *linked* onto $X \subseteq U$, if $|X| = |Y|$ and there are $|X|$ pairwise node disjoint paths from the nodes in X to the nodes in Y. The sets X and Y do not have to be disjoint, and the zero-length path consisting of a single node is also allowed. Hence X can be linked onto X in particular. The following theorem states that the graph G together with a set $X \subseteq U$ induces a matroid on the vertices of the graph (see e.g., [Rec89]):

Theorem 5.2. *If $D(U, A)$ is a directed graph and $X \subseteq U$ is a fixed subset of nodes, then those subsets $U' \subseteq U$ that can be linked onto X form the bases of a matroid M over U.*

Considering the line graph of the directed graph, one can state an arc disjoint version of Theorem 5.2:

Theorem 5.3. *If $D(U, A)$ is a directed graph, $s \in U$ is a fixed vertex and r is a positive integer, then those r-element subsets $A' \subseteq A$ whose arcs can be reached from s by r pairwise arc disjoint paths form the bases of a matroid M over A.* ■

To prove Theorem 5.1, we use the fact that $\mathscr{S}(G_x, K_x)$ can be represented by the network N_x (Lemma 4.3). Then Theorem 5.3 is used to show that the set system represented by a network is the projection of a matroid.

Proof (of Theorem 5.1). Clearly, it is sufficient to consider only the case when $H = G$, since every induced subgraph of a chordal graph is also chordal. Moreover, it can be assumed that K is a maximal (non-extendable) clique: if $K_1 \subseteq K_2$ are two cliques, then $\mathscr{S}(G, K_1)$ is the projection of $\mathscr{S}(G, K_2)$. Therefore if $\mathscr{S}(G, K_2)$ is the projection of the basis set of a matroid, then this also follows for $\mathscr{S}(G, K_1)$. We have seen in Section 2 that given a tree decomposition $T(U, F)$, $\{T_v\}_{v \in V(G)}$ of the chordal graph G, every maximal clique of G is a clique K_x for some $x \in U$. Furthermore, since the choice of the root node of T is arbitrary, it can be assumed that x is the root, thus we have $G = G_x$ and $\mathscr{S}(G, K) = \mathscr{S}(G_x, K_x)$.

By Lemma 4.3, the sets in $\mathscr{S}(G_x, K_x)$ are exactly the sets represented by the feasible flows of the network N_x. Now, as described at the beginning of Section 4, add two new nodes s, t to the network, add an arc with unit capacity from s to every source, and for every terminal x, add an arc from x to t that has capacity equal to the capacity of x. Furthermore, replace every arc e having capacity $c(e)$ with $c(e)$ parallel arcs of unit capacity, clearly this does not change the problem. Call the resulting network N'_x. By Observation 4.2, the number of sources in N_x is $r = |C_x|$, hence every feasible flow of N_x corresponds to an $\overrightarrow{st}$ flow with value r in N'_x. Since every arc has unit capacity in N'_x, an integral $\overrightarrow{st}$ flow with value r corresponds to r arc disjoint paths from s to t. Now consider the matroid M given by Lemma 5.3. Denote by A_t the arcs incident to t, and let matroid M_t be the restriction of matroid M to A_t. Let $A'_t \subseteq A_t$ be those arcs of A_t that originate from some node $x_v \in U_x$ (and not from x). We claim that $\mathscr{S}(G, K_x)$ is isomorphic to the projection of M_t to A'_t (vertex $v \in K_x$ maps to arc $\overrightarrow{x_v t}$). By Lemma 4.3, if $S \in \mathscr{S}(G, K_x)$, then there is a feasible flow in N_x where flow is consumed only by those terminals of U_x that correspond to the elements in S. Based on this flow, one can find r arc disjoint $\overrightarrow{st}$ paths in N'_x, and it follows that the matroid M_t has a base whose intersection with A'_t is exactly S, hence S is in the projection of M_t to A'_t. It is easy to show the other direction as well: if S is in the projection of M_t, then there is a feasible flow of N_x where only the terminals corresponding to S consume flow in U_x. Thus by Lemma 4.3, $S \in \mathscr{S}(G, K_x)$, as required. ■

Acknowledgments

I'm grateful to Katalin Friedl for her useful comments on the paper.

References

[AMO93] Ravindra K. Ahuja, Thomas L. Magnanti, and James B. Orlin. *Network flows*. Prentice Hall Inc., Englewood Cliffs, NJ, 1993. Theory, algorithms, and applications. **p.**

[BHT92] M. Biró, M. Hujter, and Zs. Tuza. Precoloring extension. I. Interval graphs. *Discrete Math.*, 100(1-3):267–279, 1992. **p.**

[BJW94] Hans L. Bodlaender, Klaus Jansen, and Gerhard J. Woeginger. Scheduling with incompatible jobs. *Discrete Appl. Math.*, 55(3):219–232, 1994. **p.**

[ERT80] Paul Erdős, Arthur L. Rubin, and Herbert Taylor. Choosability in graphs. In *Proceedings of the West Coast Conference on Combinatorics, Graph Theory and Computing (Humboldt State Univ., Arcata, Calif., 1979)*, pages 125–157, Winnipeg, Man., 1980. Utilitas Math. **p.**

[Gol80] Martin Charles Golumbic. *Algorithmic graph theory and perfect graphs*. Academic Press, New York, 1980. **p.**

[HT93] M. Hujter and Zs. Tuza. Precoloring extension. II. Graph classes related to bipartite graphs. *Acta Mathematica Universitatis Comenianae*, 62(1):1–11, 1993. **p.**

[HT96] M. Hujter and Zs. Tuza. Precoloring extension. III. Classes of perfect graphs. *Combin. Probab. Comput.*, 5(1):35–56, 1996. **p.**

[Klo94] Ton Kloks. *Treewidth*, volume 842 of *Lecture Notes in Computer Science*. Springer-Verlag, Berlin, 1994. Computations and approximations. **p.**

[Kra93] J. Kratochvíl. Precoloring extension with fixed color bound. *Acta Mathematica Universitatis Comenianae*, 62(2):139–153, 1993. **p.**

[Mar04] Dániel Marx. Precoloring extension on unit interval graphs, 2004. Submitted. **p.**

[MTW98] B. Mohar, Zs. Tuza, and G. Woeginger, 1998. Manuscript. **p.**

[Mur00] Kazuo Murota. *Matrices and matroids for systems analysis*, volume 20 of *Algorithms and Combinatorics*. Springer-Verlag, Berlin, 2000. **p.**

[Rec89] András Recski. *Matroid theory and its applications in electric network theory and statics*, volume 6 of *Algorithms and Combinatorics*. Springer-Verlag, Berlin, New York and Akadémiai Kiadó, Budapest, 1989. **p.**

[RTL76] Donald J. Rose, R. Endre Tarjan, and George S. Lueker. Algorithmic aspects of vertex elimination on graphs. *SIAM J. Comput.*, 5(2):266–283, 1976. **p.**

[Tuz97] Zs. Tuza. Graph colorings with local constraints—a survey. *Discuss. Math. Graph Theory*, 17(2):161–228, 1997. **p.**

[Viz76] V. G. Vizing. Coloring the vertices of a graph in prescribed colors. *Diskret. Analiz*, (29 Metody Diskret. Anal. v Teorii Kodov i Shem):3–10, 101, 1976. **p.**