Generalized coloring for tree-like graphs

by

Klaus Jansen Petra Scheffler

No. 319/1992

*Appeared in: Proc. 18th International Workshop on Graph-Theoretic Concepts in Computer Science (WG'92), Lecture Notes in Computer Science 657, (ed. E.W. Mayr), 1993, 50–59.

Generalized coloring for tree-like graphs

Klaus Jansen [*] Petra Scheffler [†]

March 25, 1992

Abstract

We discuss the Precoloring Extension (PrExt) and the List Coloring (LiCol) problems for trees, partial k-trees and cographs in the decision and the construction versions.

Both problems for partial k-trees are solved in linear time, when the number of colors is bounded by a constant and by $O(|V|^{k+2})$-algorithms in general. For trees, we improve this to linear time.

In contrast to that, PrExt and LiCol differ in complexity for cographs. While the first has a linear decision algorithm, the second is shown to be NP-complete.

We give polynomial time algorithms for the corresponding enumeration problems #PrExt and #LiCol on partial k-trees and trees and for #PrExt on cographs.

1 Introduction

This paper was motivated by Biró, Hujter and Tuza [BHT], where the following generalization of the Chromatic Number problem has been con-

[*]Fachbereich Mathematik und Informatik, Universität Trier, Postfach 38 25, W-5500 Trier, Germany. (E-mail: jansen@dm3.uni-trier.de) supported in part by the Deutsche Forschungsgemeinschaft.

[†]Fachbereich Mathematik, TU Berlin, Straße des 17. Juni 136, W-1000 Berlin 12, Germany. (E-mail: scheffle@math.tu-berlin.de) supported by the Koordinierungs- und Aufbau-Initiative für die Forschung in den Ländern Berlin, Brandenburg, Mecklenburg-Vorpommern, Sachsen, Sachsen-Anhalt und Thüringen e. V. under contract 014300/I.

sidered:

Precoloring Extension (PrExt)

Instance: A finite undirected graph $G = (V, E)$, an integer m, and a proper m-coloring of a vertex subset $W \subseteq V$.

Question: Is the coloring extendible to the entire graph G?

This problem arises in practical applications as scheduling or VLSI-layout (cf. [BHT]). It was shown by Biró et. al. that this problem is one of the problems that remain NP-complete even for interval graphs. In a second paper [HT] Hujter and Tuza show also the NP-completeness of PrExt for bipartite graphs and several related classes of graphs. This makes it important to investigate PrExt for graphs that admit efficient algorithms. Biró et. al. give a polynomial-time algorithm for PrExt on partial k-trees restricted to a fixed number of colors (cf. [BHT]). They prove that the time complexity of their algorithm is $O(|V|^{c(k,m)})$, where $c(k, m)$ is a linear function in k and m. We improve the result to a linear-time algorithm. Furthermore, we solve the PrExt problem in polynomial time for partial k-trees also in the general case when the number of colors m is part of the input and may be as large as $O(|V|)$. This was raised as an open problem in [BHT].

Our algorithms for partial k-trees are presented in section 2. They can be adapted to the construction and the enumeration problems, where we ask for a proper coloring extension or for the number of colorings, respectively. Moreover, the same approach is used to solve another problem, the List Coloring problem. This is formulated as follows:

List Coloring (LiCol)

Instance: A finite undirected graph $G = (V, E)$ together with sets of admissible colors S_v for all vertices $v \in V$.

Question: Does there exist a proper coloring of G with colors from $S = \cup_{v \in V(G)} S_v$ such that every vertex v is colored with an admissible color from S_v?

W. l. o. g. we assume that $S = \{1, 2, \ldots, m\}$ for some m and that the admissible colors are always chosen from this set. Hence, this problem is a generalization of the usual Chromatic Number problem, where $S_v = S$

for all vertices $v \in V$. It also generalizes the PREXT problem. Consider a set S_v as admissible color set for a vertex $v \in V$ where $S_v = S$ or $|S_v| = 1$. The LICOL problem was considered earlier in the literature [Viz], and it has several practical applications in scheduling problems (cf. [Jan]).

For example, one application of the LICOL problem is given as a processor assignment problem. Each vertex $v \in V$ corresponds to an operation, the graph $G = (V, E)$ describes an incompatibility graph, and $S = \{1, \ldots, m\}$ is a set of processors. We have an edge $e = \{v, w\}$ between two vertices v and w iff the corresponding operations cannot be assigned to the same processor, and the set S_v is the set of the processors that can execute the operation $v \in V$. Then, the LICOL problem is equal to the problem of finding an assignment $f : V \to \{1, \ldots, m\}$ such that the following two conditions are satisfied:

1. only compatible operations are assigned to the same processor (i. e. $f(v) \neq f(w)$ if $\{v, w\} \in E$), and

2. an assigned processor $f(v)$ can execute the operation v (i. e. $f(v) \in S_v$ for all $v \in V$).

The class of trees is a special case of k-trees with $k = 1$. Our general method from section 2 gives $O(|V|^3)$-algorithms for trees. We improve the algorithms on trees to linear time in section 3.

In section 4 we discuss the class of cographs. Surprisingly, the two problems PREXT and LICOL differ in complexity on this class. We give a linear-time algorithm for PREXT and show NP-completeness for LICOL. This contrasts with the situation on other studied graph classes, where the two problems have the same complexity. They are either both solvable in polynomial time (e. g. for partial k-trees) or both NP-complete (e. g. for interval graphs [BHT] or bipartite graphs [HT]).

2 Partial k-trees

Partial k-trees are the subgraphs of k-trees. A k-tree is a graph that can be reduced to the empty graph by the deletion of vertices of degree k with a completely connected neighbourhood. Partial k-trees are well studied and known also as graphs that have a tree-decomposition of width k. We give a definition of tree-decompositions below.

Many intractable problems are solvable in polynomial time for partial k-trees. The CHROMATIC NUMBER problem (cf. [AP]) is one of them. For the proof of this it is used that the chromatic number of a partial k-tree is at most $k+1$. A similar approach is useful to get a linear algorithm for PrExt with a fixed number of colors.

But the general PrExt problem, where the number of needed colors may be larger than the clique number of the graph, apparently does not fall into the class of problems for that linear-time algorithms are known (see the general results e. g. in [ALS] and [Sch]). Nevertheless, the methods from [Sch] allow us to develop a polynomial-time algorithm for the general PrExt problem.

We use a special kind of tree-decompositions to represent partial k-trees.

Definition 2.1 *A* **nice tree-decomposition** *of width k for a graph $G = (V, E)$ is a pair $(T, \mathcal{X})$, where T is an oriented tree with at most two children per node and $\mathcal{X} = \{X_t : X_t \subseteq V, t \in V(T)\}$ such that:*

(i) $\bigcup_{t \in V(T)} X_t = V(G)$,

(ii) for every $\{u, v\} \in E(G)$, there is a $t \in V(T)$ such that $\{u, v\} \subseteq X_t$,

(iii) $X_i \cap X_k \subseteq X_j$ for all $\{i, j, k\} \subseteq V(T)$ such that j lies on the path between i and k,

(iv) $|X_t| \leq k+1$ for every $t \in V(T)$,

(v) $X_i = X_k = X_j$ if $i \in V(T)$ has two children j and k and

(vi) if j is the only child of i in T then there is a $v \in V(G)$ that either $X_i = X_j \cup \{v\}$ or $X_j = X_i \cup \{v\}$.

This kind of tree-decomposition is very useful to handle partial k-trees efficiently. In fact, it is no restriction compared with the original tree-decomposition as it was given by Robertson and Seymour in [RS]. Indeed, the following fact is easy to prove, see [Sch]:

Lemma 2.1 *Provided a tree-decomposition of width k for a graph G is given, then a nice tree-decomposition of the same width can be constructed in linear time. Its decomposition-tree T has at most $O(|V(G)|)$ nodes.*

It is known, that a tree-decomposition of width k can be constructed in polynomial time for a graph if one exists. The best algorithm for this was published by Lagergren [Lag] and improved by Bodlaender and Kloks

[BK]. It needs $O(|V| \log^2 |V|)$ time. Recently, Bruce Reed announced an $O(|V| \log |V|)$-algorithm [Ree].

Hence, given a graph that has tree-width at most k, we can obtain a nice tree-decomposition with width k in time $O(|V| \log |V|)$. Such a nice tree-decomposition gives a method to reconstruct the graph consecutively by simple operations, starting with small graphs of size at most $k+1$. This is useful to solve the PrExt and LiCol problems.

Similarly to Bern, Lawler and Wong [BLW], we consider **$k+1$-terminal graphs** that are pairs (G, X) consisting of a graph G together with a set of at most $k+1$ terminal nodes $X \subseteq V(G)$. They are constructed by the following four operations:

- *Start:* Take (G, X) with $X = V(G)$. This operation corresponds to the leaves in a decomposition-tree.
- *Forget:* Take (G, X), where $X = Y \setminus \{v\}$ for a given terminal graph (G, Y). This operation corresponds to an edge (s, t) in the decomposition tree T, where s is the only child of t and $Y = X_s = X_t \cup \{v\}$.
- *Introduce:* Take (G, X) where $X = Y \cup \{v\}$ and $V(G) = V(H) \cup \{v\}$ and $E(G) \subseteq E(H) \cup \{\{v, y\} : y \in Y\}$ for a given terminal graph (H, Y). This operation corresponds to an edge (s, t) in the decomposition tree T, where s is the only child of t and $Y = X_s = X_t \setminus \{v\}$.
- *Join:* Take (G, X) where $G = G_1 \cup G_2$ for two given terminal graphs (G_1, X) and (G_2, X) with the same set X of terminal nodes. This operation corresponds to a node with two children in the decomposition tree.

Let $(T, \mathcal{X})$ be a tree-decomposition of a graph G. Then, we get a sequence of terminal graphs (G_t, X_t) constructed according to the four composition operations, starting with small graphs with at most $k+1$ vertices and proceeding in post-order. Observe, that each set of terminal nodes X_t is a separator that separates the graph G_t from the rest of G.

A coloring of a graph G with m colors is a partition $\bar{C} = (C_1, \ldots, C_m)$ of its vertex set $V(G)$ into m independent sets. If the vertex set W_i is precolored by the color number i (for $1 \leq i \leq m$), we have to fulfill the additional conditions $W_i \subseteq C_i$ for all i. Clearly, every proper coloring of the graph G induces a proper coloring on every subgraph of G. On the other hand, we have the following for the subgraphs corresponding to the decomposition tree:

Lemma 2.2 *A proper coloring $\bar{C}_t = (C_{t,1}, \ldots, C_{t,m})$ of G_t is extendible to G, iff the coloring $\bar{H}_t = (H_{t,1}, \ldots, H_{t,m})$ induced on the terminal nodes X_t is extendible to $G \setminus G_t$.*

This is obvious because of the separator property of X_t. We define that two colorings of a terminal graph (G_t, X_t) are equivalent, if they have the same heads, i. e. they agree on their terminal vertices.

Definition 2.2 *The coloring $\bar{H} = (H_1, \ldots, H_m)$ induced on X_t is called the* **head** *of a coloring $\bar{C} = (C_1, \ldots, C_m)$ of (G_t, X_t), if $H_i = C_i \downarrow X_t$ is the restriction of the color class to the set of terminal nodes for all i.*

Two colorings $\bar{C} = (C_1, \ldots, C_m)$ and $\bar{C}' = (C'_1, \ldots, C'_m)$ of a terminal graph (G_t, X_t) are **equivalent**, *if their heads $\bar{H} = (H_1, \ldots, H_m)$ and $\bar{H}' = (H'_1, \ldots, H'_m)$ are equal, i. e. for all i holds $H_i = H'_i$.*

Lemma 2.3 *The number of equivalence classes of m-colorings for a terminal graph (G_t, X_t) is at most $m^{|X_t|}$.*

For our first coloring algorithm we use a dynamic programming approach and color the subgraphs G_t step by step in post-order. This can be done efficiently due to the constant number of equivalence classes. Define the following function for colorings of X_t:

Definition 2.3

$$f(\bar{H}_t) = \begin{cases} \text{TRUE} & \textit{if a proper coloring of } G_t \textit{ with head } \bar{H}_t \textit{ exists} \\ \text{FALSE} & \textit{otherwise} \end{cases}$$

Clearly, the graph G has a proper coloring, iff there exists a coloring of the terminal set X_r of the root r such that $f(\bar{H}_r)$ =TRUE. We compute the function f for all nodes t of the decomposition tree and all possible colorings of X_t. This can be done recursively in post-order, starting at the leaves. Depending on the local structure of the decomposition tree the following cases occur:

- *Start:* $f(\bar{H}_t)$ =TRUE, iff adjacent vertices of X_t are in different color classes $H_{t,i}$ and $H_{t,j}$ and the precoloring is respected. This can be checked in time $O(k^2)$ for graphs with at most $k+1$ vertices.

- *Forget:* $f(\bar{H}_t)$ =TRUE, iff there is at least one possible extension of $\bar{H}_t$ to X_s with $f(\bar{H}_s)$ =TRUE, where s is the child of t. We get a possible extension to X_s by coloring the unique vertex $v \in X_s \setminus X_t$ by any feasible color. So in that case, we have to perform at most $m \cdot m^k$ operations to compute the function f for all colorings of the set X_t.

- *Introduce:* This case is similar to the previous one. But now the new vertex is in the parent set. So we have to check in addition, whether or not the coloring is a proper one in X_t. We have $f(\bar{H}_t)$ =TRUE, iff $\bar{H}_t$ is feasible in X_t and $f(\bar{H}_s)$ =TRUE for the restriction to X_s, where $H_{s,i} = H_{t,i} \downarrow X_s$ for all i. Therefore, we have to check only the precoloring of v and that v has another color than all its neighbours, where v is the only vertex from $X_t \setminus X_s$. This computation needs k operations for every coloring of X_t.

- *Join:* $f(\bar{H}_t)$ =TRUE, iff $f(\bar{H}_s)$ =TRUE and $f(\bar{H}_{s'})$ =TRUE for the two children s and s' of t. We have $H_{s,i} = H_{t,i} = H_{s',i}$ for all i. Clearly, one operation per coloring is enough.

Theorem 2.4 *If the number of colors is bounded by a constant m and the graph is given together with a tree-decomposition of width k,* PREXT *can be solved in linear time.*

Proof:
The above observations show, that the following algorithm PREXT is correct and solves the considered problem. The time complexity is mainly determined by the number of nodes in the decomposition tree. For every node $t \in V(T)$, a constant number of at most $m^{k+1} \cdot k^2$ operations is executed. In total, the algorithm needs linear time $O(|V|)$. □

Algorithm 2.1 PREXT

```
Input: Graph $G$ with a tree-decomposition $(T, \mathcal{X})$ of width $k$, integer $m$,
proper precoloring of $W \subseteq V$.
Output: YES, iff the precoloring can be extented to $V$.
for all nodes $t \in V(T)$ in post-order do
    for all colorings $\bar{H}_t$ of $X_t$ do case
        start : begin
            if $\bar{H}_t$ is feasible on $X_t$ then $f(\bar{H}_t)$ :=TRUE
```

```
          else $f(\bar{H}_t)$ :=FALSE end ;
      forget : begin $f(\bar{H}_t)$ :=FALSE;
          for all extensions $\bar{H}_s$ to the head of the child $s$ do
              if $f(\bar{H}_s)$ :=TRUE then $f(\bar{H}_t)$ :=TRUE endfor end ;
      introduce : begin
          if $\bar{H}_t$ is feasible on $X_t$ and $f(\bar{H}_s)$ =TRUE
          then $f(\bar{H}_t)$ :=TRUE else $f(\bar{H}_t)$ :=FALSE; end ;
      join : begin
          if $f(\bar{H}_s)$ =TRUE and $f(\bar{H}_{s'})$ =TRUE then $f(\bar{H}_t)$ :=TRUE
          else $f(\bar{H}_t)$ :=FALSE; end ; endfor ;
   endfor ;
if there exists at least one coloring of $X_r$ for the root $r$ with $f(\bar{H}_r)$ =TRUE
then return YES else return NO.
```

Looking at the exact bound for the computing time of the above algorithm — $c \cdot m^{k+1} \cdot k^2 \cdot |V|$ — we realize that this is also polynomial in the number of colors m. So, as a simple corollary we get the following:

Theorem 2.5 PrExt *for partial k-trees is decidable in time $O(|V|^{k+2})$.*

Furthermore, it is clear that the construction problem can be solved easily by backtracking. Here we ask for an extension of the given precoloring. For this we store one feasible extension to the child s proving $f(\bar{H}_t)$ =TRUE whenever this holds in a *Forget*-node during the original algorithm. Then, we walk once more through the decomposition tree, now starting at the root r, to get a feasible coloring of G.

Theorem 2.6 *A coloring extending a given precoloring with minimal number of colors for a partial k-tree can be constructed in time $O(|V|^{k+2})$. The time can be reduced to linear, provided a tree-decomposition of the input graph is given and the number of colors is bounded by a constant.*

In the enumeration problem #PrExt we count the number of feasible colorings extending a given precoloring. For that purpose we adapt our algorithm. We consider the unit cost model, i. e. we count the number of operations independent to the sizes of the numbers. Compared with our previous algorithm, we only replace the function f by a function g that counts the number of feasible colorings of the subgraph G_t.

Definition 2.4 *For any coloring $\bar{H}_t$ of X_t the value $g(\bar{H}_t)$ is the number of proper colorings of graph G_t having head $\bar{H}_t$.*

The function g can be computed bottom up in the following way:

- *Start:* Set $g(\bar{H}_t) := 1$, iff adjacent vertices of X_t are in different color classes and the precoloring is respected; otherwise set $g(\bar{H}_t) := 0$.
- *Forget:* Set $g(\bar{H}_t)$ equal to $\sum g(\bar{H}_s)$ where the sum is taken over all possible extensions of $\bar{H}_t$ to X_s.
- *Introduce:* If $\bar{H}_t$ is feasible on X_t, then set $g(\bar{H}_t)$ to $g(\bar{H}_s)$ where $\bar{H}_s$ is the restriction of $\bar{H}_t$ to X_s, with $H_{s,i} = H_{t,i} \downarrow X_s$ for every i. In the other case, set $g(\bar{H}_t) := 0$.
- *Join:* Set $g(\bar{H}_t)$ to the product $g(\bar{H}_s) \cdot g(\bar{H}_{s'})$ where s and s' are the children of t.

We get the number of different colorings of the entire graph by computing $\sum g(\bar{H}_r)$ where the sum is taken over all possible colorings $\bar{H}_r$ of the root r of the decomposition tree. Obviously, this computation does not need more time as the decision procedure PrExt if we use the unit cost measure.

Theorem 2.7 *The number of possible extensions of a precoloring can be counted in $O(|V|^{k+2})$ time on partial k-trees. Provided the number of colors is bounded by a constant and a tree-decomposition of G is given, then linear time is sufficient.*

Finally, in this section we consider the LiCol problem. For this problem only few details have to be changed. Instead of a completely determined precoloring for some vertices, we have a set S_v of colors to choose from for each $v \in V$. We simply adapt the Definition 2.3 of our function f — the meaning of the phrase "feasible coloring" has to be changed. Nevertheless, the computing time does not change. The number of equivalence classes does not increase in the worst case, too. The same holds for the computation of the function g in the enumeration problem. So, we immediately have the following results:

Theorem 2.8 *The problem* LiCol *is decidable in time $O(|V|^{k+2})$ for partial k-trees. There are algorithms with the same time complexity to construct a list coloring (if one exists) and to count the number of feasible list colorings for partial k-trees. If a constant bound for the number of colors and a tree-decomposition are given, then linear time is sufficient.*

3 Trees

A graph $T = (V, E)$ is called a tree if it is connected and has $|E| = |V| - 1$ edges. It is easy to show, that any tree has a tree-decomposition of width 1. Therefore, the above results imply $O(|V|^3)$-algorithms for the PrExt and the LiCol problems restricted to trees. In this section, we improve the running time to linear-time algorithms for both problems.

We assume that the tree is given as an out-tree with a root $r \in V$ that has no predecessor, with a set of children $C(x) \subset V$ for each vertex $x \in V$ and an unique parent for each vertex $x \in V \setminus \{r\}$. A **leaf** of T is a vertex x that has no children. The other vertices are called non-leaves or **inner vertices** of T.

We start with an algorithm for the LiCol problem. We determine two values $unique(x) \in \{\text{TRUE}, \text{FALSE}\}$ and $color(x) \in \{0, \ldots, m\}$ for each vertex $x \in V$. If the color is unique at a vertex x and cannot be changed in a feasible coloring of the subtree rooted at vertex x, then we set $unique(x) := \text{TRUE}$; otherwise we set $unique(x) := \text{FALSE}$. At the beginning the unique colors for vertices with $S_x = \{c\}$ are stored in $color(x)$; for the other vertices we set $color(x) := 0$.

The vertices of the out-tree are traversed in post-order. If a child $y \in C(x)$ has the unique color c at an inner vertex x , then this color cannot be used for x. We store the unique colors of the children in a set $pc(x)$. After that, we determine the values $unique(x)$ and $color(x)$.

Algorithm 3.1 LiCol

```
Input: Tree $T$, feasible color sets $S_x$ with $|S_x| \geq 1$ for all $x \in V(T)$.
Output: YES, iff there exists a feasible list coloring of $T$.
for all $x \in V$ in post-order do $pc(x) := \emptyset$;
    for all $y \in C(x)$ do
        if $unique(y) = \text{TRUE}$ then $pc(x) := pc(x) \cup \{color(y)\}$; endfor ;
    if $|S_x \setminus pc(x)| \geq 2$ then $unique(x) := \text{FALSE}$; $color(x) := 0$;
    if $S_x \setminus pc(x) = \{c\}$ then $unique(x) := \text{TRUE}$; $color(x) := c$;
    if $S_x \setminus pc(x) = \emptyset$ then return NO; stop endfor ;
return YES.
```

To show the correctness of the algorithm, it is sufficient to prove the next lemma.

Lemma 3.1 *For a fixed inner vertex $x \in V$ let $V_x = \{y \in C(x) |$ unique(y) $=$ FALSE $\}$ be the set of its children that are not uniquely colored and let $pc(x)$ be the set of colors that are determined uniquely on the other children from $C(x) \setminus V_x$. Then, every coloring $f(x)$ of vertex x with color $f(x) \in S_x \setminus pc(x)$ can be extended to the children of x.*

Proof:
Since $unique(y) =$ FALSE for each vertex $y \in V_x$, there are at least two feasible colors $f_1(y)$ and $f_2(y)$ in colorings for the subtree rooted at y. Then, given a coloring $f(x)$ of x, at least one of those two colors is unequal to $f(x)$ and can be used. $\square$

The travel in post-order through the out-tree needs $O(|V|)$ time. The computation time for one set $pc(x)$ at a vertex $x \in V$ can be bounded by $O(|C(x)|)$. The decision whether one or two colors are in $S_x \setminus pc(x)$ can be done in $O(|pc(x)| + 2)$ time. Since each vertex in a tree has at most one predecessor and since $|pc(x)| \leq |C(x)|$, the entire algorithm needs at most $O(|V|)$ time. It needs $O(\sum_{x \in V} |S_x|)$ space in the worst case.

A feasible coloring, if it exists, can also be found in linear time. During the algorithm we compute two possible colors $f_1(x)$ and $f_2(x)$ at the vertices with value $number(x) =$ FALSE. We can do this in $O(|pc(x)| + 2)$ steps for any such vertex x. At the end of the first algorithm we have to walk once again through the tree, now in pre-order and compute one possible coloring in $O(|V|)$ time. So, we get:

Theorem 3.2 *The problems* LiCol *and* PrExt *are decidable in linear time for trees. If a feasible coloring exists, it can be found in $O(|V|)$ time.*

In the following, we consider the problem #LiCol to enumerate all feasible colorings of a tree $T = (V, E)$ with given admissible color sets $S_x \subset \{1, \ldots, m\}$. For each vertex $x \in V$ and each color $i \in \{1, \ldots, m\}$, we denote by $nc(x, i)$ the number of such colorings of the subtree of T rooted at x, where the color of x is equal to i. Then, $nc(x) = \sum_{i=1}^{m} nc(x, i)$ is the number of colorings for the subtree rooted at x, and $nc(r)$ is the number of colorings of the entire tree T with root $r \in V$. The following algorithm shows a way to compute these numbers:

Algorithm 3.2 #LiCol

Input: *Tree T, admissible color sets S_x with $\cup_{x \in V} S_x = \{1, \ldots, m\}$.*

```
Output: The number of feasible list colorings of $T$.
for all $x \in V$ in post-order do $nc(x) := 0$;
    for all $i \in \{1, \ldots, m\}$ do $nc(x, i) := 0$;
        if $i \in S_x$ and $x$ is a leaf then $nc(x, i) := 1$;
        if $i \in S_x$ and $x$ is an inner vertex
        then $nc(x, i) := \prod_{y \in C(x)} (nc(y) - nc(y, i))$;
        $nc(x) := nc(x) + nc(x, i)$; endfor ;
    endfor ;
return $nc(r)$.
```

We show that the algorithm is correct. For a leaf the assertion is clear. If we assign a color i to an inner vertex x, then its children must have different colors. Therefore, we compose only colorings of the subtrees at the children $y \in C(x)$ with colors of the children unequal to i. Since the colorings in the subtrees are independent, we get the product of the numbers $nc(y) - nc(y, i)$.

In the computation, we use $m \cdot |V|$ values $nc(x, i)$. After the values $nc(x, i)$ for a vertex are computed, each of them is used at most once for the unique predecessor of x in the tree. Hence, the number of operations is bounded by $O(m \cdot |V|)$, if we consider the unit cost measure.

Theorem 3.3 *The enumeration problems #LiCol and #PrExt for trees are solvable in $O(m \cdot |V|)$ time.*

4 Cographs

In this section, we give a linear-time algorithm for the PrExt problem restricted to cographs. On the other hand, we show that the LiCol problem restricted to cographs is NP-complete. Provided the number of colors m is constant, we give a linear algorithm for LiCol on cographs. After that, we analyse the corresponding enumeration problems.

Definition 4.1 *Take $r \geq 2$ disjoint graphs $G_i = (V_i, E_i)$ with $(1 \leq i \leq r)$ and $V_i \cap V_j = \emptyset$ for $i \neq j$. Their* **union** *$\cup_{i=1}^{r} G_i$ is defined as the graph $(\cup_{i=1}^{r} V_i, \cup_{i=1}^{r} E_i)$. Their* **product** *$\times_{i=1}^{r} G_i$ is obtained by first taking the union of the r graphs, and then adding all edges (v_i, v_j) with $v_i \in V_i$, $v_j \in V_j$ and $1 \leq i \neq j \leq r$.*

Definition 4.2 *The class of cographs is the smallest set of graphs, fulfilling the following rules:*

1. *Every graph $G = (V, E)$ with one vertex and no edges is a cograph.*
2. *If $G_i = (V_i, E_i)$ are cographs with pairwise disjoint vertex sets for $1 \le i \le r$ and $r \ge 2$, then their union $\cup_{i=1}^{r} G_i$ is a cograph.*
3. *If $G_i = (V_i, E_i)$ are cographs with pairwise disjoint vertex sets for $1 \le i \le r$ and $r \ge 2$, then their product $\times_{i=1}^{r} G_i$ is a cograph.*

We associate a corresponding rooted tree $T = (I, F)$ to each cograph $G = (V, E)$. It is called a cotree of G and it reflects the construction of the cograph according to the above definition in the following way: Each non-leaf node in the tree is labeled either with $\cup$ (union-node) or with $\times$ (product-node). It has two or more children. If two non-leaf nodes are connected by an edge, then they have different labels. Each node $x \in I$ of the cotree corresponds to a cograph $G_x = (V_x, E_x)$. A leaf node corresponds to a cograph with one vertex and no edges. An union-node (product-node) corresponds to the union (product) of the cographs, associated with the children of the node. Finally, the entire cograph is given by the cograph associated with the root $r \in I$ of the cotree. In [CPS], it is shown that one can decide in $O(|V| + |E|)$ time, whether a graph is a cograph, and build a corresponding cotree.

Our algorithm for PrExt on cographs has the following structure: At first we find a cotree for the input graph. Then, for each node x of the cotree, we compute a set $pc(x) \subset \{1, \dots, m\}$ of colors used in the precoloring of the cograph corresponding to x. These sets are computed in post-order starting with the leaves. Then, we compute the minimal number of necessary colors to extend the given precoloring of G_x depending on the size $|pc(x)|$ bottom-up in the cotree. Now, we show how to compute the precolor sets $pc(x)$.

Algorithm 4.1 Colorsets

```
Input: Cograph $G$ with a cotree $T$ and a proper precoloring of $W \subseteq V$.
Output: Sets $pc(x)$ of used precolors in $G_x$ for all $x \in V$.
for all $x \in I$ in post-order do case
    leaf : begin $pc(x) := \emptyset$;
        if $x$ is precolored with $c$ then $pc(x) = \{c\}$; end ;
```

```
union : begin $pc(x) = \emptyset$;
    for all children $y \in C(x)$ do $pc(x) = pc(x) \cup pc(y)$;
        endfor ; end ;
product : begin $pc(x) = \emptyset$;
    for all children $y \in C(x)$ do
        if $pc(x) \cap pc(y) \neq \emptyset$ then $pc(x) = pc(x) \cup pc(y)$
        else return "No Extension is possible" stop endfor ;
    end endfor
```

The time to compute the set $pc(x)$ for one union or product node is at most$O(\sum_{y \in C(x)} |pc(y)|)$. Clearly, we can bound the complexity of this algorithm by $O(m \cdot |V|)$. But this gives, if m is large, a quadratic time bound $O(|V|^2)$. We show, that this is not the best possible one can do.

Lemma 4.1 *The algorithm needs at most $O(|V| + |E|)$ time to compute the color sets $pc(x)$ for all nodes of the cotree .*

Proof:
We sum up the computing time needed in the product nodes x to compute the color sets $pc(x)$ and $pc(y)$ for the children $y \in C(x)$ over the cotree. Since union and product nodes in a cotree appear alternately, this sum includes the computing time at all union nodes of T, except, may be, at the root node r and some leaves. So, we add the time needed there in a second step.

In the first step we bound the time to compute the color sets $pc(x)$ and $pc(y)$ for the children $y \in C(x)$ by the number of edges generated by the product at node x plus one. Let G_x be the corresponding cograph to a product node. W. l. o. g. we assume that the cograph G_x consists of $r' \leq r$ cographs of the form $G_{y_i} = \cup_{z \in C(y_i)} G_z$ for $1 \leq i \leq r'$ and of $r - r'$ single vertex graphs $G_{y_{r'+1}}, \ldots, G_{y_r}$. Then, the number of computation steps at node x and its children is bounded by $O(\sum_{1 \leq i \leq r', z \in C(y_i)} |V_z|) + (r - r')$, since no more colors as vertices occur. Since all cographs are disjoint, the sum above is less or equal to $\sum_{y_i \in C(x)} |V_{y_i}|$.

If x has three or more children, the sum $\sum_{y \in C(x)} |V_y|$ is less or equal to the number of edges $|\{(v, v') | v \in V_y, v' \in V_{y'}, y, y' \in C(x), y \neq y'\}|$ generated by the product. If x has only two children, then the sum is less or equal to one plus the number of edges generated by the product at x. Summing up, we get for all product nodes and its children together the computing time $O(|E|)$.

In the case when the root r of the cotree is a union node, we bound the time to compute the set $pc(r)$ by $O(|V|)$. The size of a set $pc(y)$ for a child $y \in C(r)$ is less or equal to the number of vertices $|V_y|$ in the cograph associated with y. Therefore, we bound the computation time for $pc(r)$ by $O(\sum_{y \in C(r)} |V_y|) = O(|V|)$. In addition, we need $O(|V|)$ time at the leaves.
In total, the algorithm needs at most $O(|V| + |E|)$ time. □

After this step of our algorithm, we know the set of precolors $pc(x)$ occuring in its corresponding cograph G_x for each node x of the cotree. If we found at a product node the same precolor in different subtrees, our algorithm has stoped. Hence, we may assume that the precolor sets for all children of a product node are pairwise disjoint.

Now the optimal extension of the given precoloring on $W \subset V$ can be computed in $O(|V|)$ time using the sets $pc(x)$. The minimal number of colors to extend the precoloring to a cograph G_x is denoted by $\chi(x, W)$. These values are computed in post-order.

Lemma 4.2 *Let x be a node of the cotree. Then, the value $\chi(x, W)$ is determined by the following rules:*

(i) If x is a leaf, then $\chi(x, W) = 1$.

(ii) If x is a product node, then $\chi(x, W) = \sum_{y \in C(x)} \chi(y, W)$.

(iii) If x is union node, then $\chi(x, W) = \max(|pc(x)|, max_{y \in C(x)} \chi(y, W))$.

Proof:
The first and second equalities are clear. For the third it is enough to prove it for two children y_1 and y_2. Then, the assertion follows by induction on the number of children. One direction — that $\chi(x, W)$ is greater or equal to $\max(\chi(y_1, W), \chi(y_2, W), |pc(x)|)$ — is also clear. For the other direction, let us define the following variables:

- *The number of equal precolors:* $a = |pc(y_1) \cap pc(y_2)|$.
- *The number of precolors occuring only in G_{y_1}:* $b = |pc(y_1)| - a$.
- *The number of precolors occuring only in G_{y_2}:* $c = |pc(y_2)| - a$.
- *The number of additional colors for G_{y_1}:* $d = \chi(y_1, W) - |pc(y_1)|$.
- *The number of additional colors for G_{y_2}:* $e = \chi(y_2, W) - |pc(y_2)|$.

Clearly, we have $\chi(y_1, W) = a + b + d$ and $\chi(y_2, W) = a + c + e$. We show (iii) by case analysis. The following cases are possible:

1. $b \geq e$ and $c \geq d$: $\chi(x, W) = a + b + c = |pc(x)|$.
2. $b < e$ and $c \geq d$: $\chi(x, W) = a + c + e = \chi(y_2, W)$.
3. $b \geq e$ and $c < d$: $\chi(x, W) = a + b + d = \chi(y_1, W)$.
4. $b < e$ and $c < d$: $\chi(x, W) = a + b + c + max(e - b, d - c)$.
    - $e - b \geq d - c$: $\chi(x, W) = a + c + e = \chi(y_2, W)$.
    - $d - c \geq e - b$: $\chi(x, W) = a + b + d = \chi(y_1, W)$. □

As consequence of both lemmas we get the following result for PrExt.

Theorem 4.3 *The problem* PrExt *restricted to cographs is decidable in linear time* $O(|V| + |E|)$.

Now we extend our algorithm to find an optimal extension of a precoloring. For that we use an additional walk through the cotree in pre-order. Each node $x \in I$ gets an additional set of free colors $fc(x)$. Assuming that $pc(r) = \{1, \ldots, m\}$ and $\chi(r, W) = m' > m$ for the root r, we set $fc(r) = \{m + 1, \ldots, m'\}$. If $\chi(r, W) = m$, the second set $fc(r)$ is empty. Starting with the root, these two sets are divided up for the children. At a product node x, each child $y \in C(x)$ gets its set $pc(y)$ and $\chi(y, W) - |pc(y)|$ free colors of $fc(x)$. At an union node x, additional to the set $pc(y)$, a child gets the same number of further colors, but we can take also colors from $pc(x) \setminus pc(y)$. The computation of the sets $pc(y)$ and $fc(y)$ is done in $O(\chi(y, W))$ time and the computation time of all these sets is bounded by $O(|V| + |E|)$ using the same arguments as in Lemma 4.1.

Theorem 4.4 *The construction of a coloring extending a given partial coloring with minimal number of colors is possible in* $O(|V| + |E|)$ *time for cographs.*

Now we consider the more general LiCol problem with a set of feasible colors S_v for each vertex $v \in V$.

Theorem 4.5 *The* LiCol *problem restricted to cographs is NP-complete.*

Proof:
Clearly, the problem is in **NP**. We give a transformation from the NP-complete 3-Satisfiability problem (3-sat) to this coloring problem (cf. [GJ] for the 3-sat problem).

Let $c_1 \wedge \ldots \wedge c_m$ with $c_i = (y_{i,1} \vee y_{i,2} \vee y_{i,3})$ and $y_{i,j} \in X = \{x_1, \overline{x_1}, \ldots, x_n, \overline{x_n}\}$ be an instance of 3-sat. We define an input cograph $G = (V, E)$ for the LiCol problem corresponding to the instance of 3-sat. It has vertex set $V = \{a_1, \ldots, a_n, b_1, \ldots, b_m\}$ and it is the product of two disjoint graphs G_1 and G_2, where the first graph is $G_1 = (V_1, E_1)$ with $V_1 = \{a_1, \ldots, a_n\}$ and $E_1 = \emptyset$. The second graph is $G_2 = (V_2, E_2)$ with $V_2 = \{b_1, \ldots, b_m\}$ and $E_2 = \emptyset$. The feasible color set for a vertex a_i is $S_{a_i} = \{x_i, \overline{x_i}\}$ and the feasible color set for a vertex b_i is $S_{b_i} = \{y_{i,1}, y_{i,2}, y_{i,3}\}$.

We claim that G has a feasible vertex coloring $f : V \to X$ with $f(v) \in S_v$, iff there is a truth assignment, that verifies all m clauses. Suppose, we have a truth assignment, that verifies the clauses. Then, we construct a feasible coloring in the following way:

$$f(a_i) = \begin{cases} x_i & \text{if } x_i \text{ is FALSE} \\ \overline{x_i} & \text{otherwise} \end{cases}$$

$$f(b_i) = \begin{cases} y_{i,1} & \text{if } y_{i,1} \text{ is TRUE} \\ y_{i,2} & \text{if } y_{i,1} \text{ is FALSE, } y_{i,2} \text{ is TRUE} \\ y_{i,3} & \text{otherwise.} \end{cases}$$

On the other hand, when we have a feasible coloring for G with $f(x) \in S_x$, then there must be a truth assignment that verifies all m clauses: If $f(a_i)$ is equal to $\overline{x_i}$, then we set x_i true, otherwise we choose as truth value for x_i false. Then, the assigned color y_{i,i_j} with $i_j \in \{1, 2, 3\}$ for vertex b_i corresponds to the literal in the clause c_i which gets the truth value true.

$\square$

Since an algorithm for the enumeration problem #LiCol implies an algorithm for LiCol, we get the following result:

Theorem 4.6 *The problem #LiCol is NP-hard for cographs.*

Nevertheless, if the number of different colors in $\cup_{x \in V} S_x$ is bounded by a constant m, LiCol is polynomial time solvable on cographs. We give an algorithm to compute a set of feasible color sets for each node of the cotree. We call a subset $A \subset \{1, \ldots, m\}$ feasible for the cograph $G_x = (V_x, E_x)$

associated to the node x, if there is a coloring $f : V_x \to A$ of the vertices in G_x which uses all colors in A and which satisfies $f(v) \in S_v$ for each vertex $v \in V_x$. We denote with $colset_x$ the set of all feasible sets A for the cograph G_x. Depending on the nodes in the cotree, we get a procedure to compute the sets $colset_x$.

Lemma 4.7 *Let x be a node in the cotree T. Then, $colset_x$ is determined by the following rules:*

(i) *If x is a leaf with corresponding vertex v in the cograph, then $colset_x = \{\{c\} | c \in S_v\}$.*

(ii) *If x is a product node with children $y_1, \ldots, y_r$, then $colset_x = \{A_1 \cup \ldots \cup A_r | A_i \in colset_{y_i}, 1 \leq i \leq r, A_j \cap A_k = \emptyset, 1 \leq j \neq k \leq r\}$.*

(iii) *If x is an union node with children $y_1, \ldots, y_r$, then $colset_x = \{A_1 \cup \ldots \cup A_r | A_i \in colset_{y_i}, 1 \leq i \leq r\}$.*

Notice, that we must compute only the minimal feasible sets A for each node, i. e. such that there is no further feasible set $A' \subset A$, $A' \neq A$. Hence, the number of feasible sets in $colset_x$ is bounded by 2^{m-1}. The computation is done in $O(4^{m-1}\,(r-1)\,m)$ time for each node of the cotree. In total, we need at most $O(4^{m-1}\,m\,|V|)$ steps, because each node despite the root occures only once as a child of another node. As consequence, we get the following result:

Theorem 4.8 *The* LiCol *problem with a constant total number of colors $|\cup_{x \in V} S_x|$ can be solved in linear time $O(|V| + |E|)$ for cographs.*

In the last part of our paper we consider the enumeration problem for the PrExt problem restricted to cographs. For that problem, a generalization of our linear-time algorithm for PrExt on cographs is possible.

First, we apply the algorithm to compute the sets of occuring precolors $pc(x)$ and the coloring number values $\chi(x, W)$ for all nodes in the cotree. Then, for each node x in the cotree and each color i, $1 \leq i \leq m$, we compute the value $nc(x, i)$. It is defined as the number of different colorings of the cograph G_x, where in addition to the precolor set $pc(x)$ further i colors are used. To compute the number of different colorings with m colors for the entire graph, we take the sum $\sum_{i \leq m} nc(r, i)$, where r is the root of the cotree. Clearly, the value $nc(x, i)$ is zero, if i is smaller than the difference between $\chi(x, W)$ and $|pc(x)|$ for some node x.

For simplification, we assume in this part that we have a binary cotree. However, it is easy to see that an arbitrary cotree can be transformed in a cotree with two children per inner node. The following algorithm shows, how we compute the values $nc(x,i)$ efficiently in the cotree.

Algorithm 4.2 #PrExt

```
Input: Cograph G with binary cotree T and a precoloring of $W \subseteq V$.
Output: The number of feasible extensions of the precoloring to $V$.
for all $x \in I$ in post-order do case
    leaf : begin
        if $x$ is precolored then begin $nc(x,0) := 1$;
            for all $i \in \{1,\dots,m\}$ do $nc(x,i) := 0$; endfor ; end ;
        else begin $nc(x,0) := 0$; $nc(x,1) := 1$;
            for all $i \in \{2,\dots,m\}$ do $nc(x,i) := 0$; endfor ; end ; end ;
    product : for all $i \in \{0,\dots,m\}$ do
        $nc(x,i) := \sum_{k=0}^{i} \binom{i}{k} \cdot nc(y_1,k) \cdot nc(y_2,i-k)$; endfor ;
    union : for all $i \in \{0,\dots,m\}$ do $nc(x,i) :=$
  $\sum_{k=0}^{i} \sum_{k'=0}^{k} \sum_{k_1=0}^{c} \sum_{k_2=0}^{b} \binom{i}{k}\binom{k}{k'}\binom{c}{k_1}\binom{b}{k_2} nc(y_1,k+k_1)\, nc(y_2,i-k+k'+k_2)$
        endfor ;
    $nc(x) := \sum_{i=0}^{m} nc(x,i)$;
    endfor ;
return $nc(r)$.
```

In the last case of the algorithm, we have used the values $c = |pc(x) \setminus pc(y_1)|$ and $b = |pc(x) \setminus pc(y_2)|$ as defined in the proof of Lemma 4.2.

It is clear that only $O(m \cdot |V|)$ values $nc(x,i)$ must be computed. The computation of all used binomial cofficients can be done in $O(m^2)$ time. To compute one value $nc(x,i)$ at an inner node we need for a product $O(m)$ time and for an union $O(m^4)$ time steps. This gives a time complexity of $O(m^5 \cdot |V|)$, if we use the unit cost measure.

Theorem 4.9 *The problem #PrExt restricted to cographs can be solved in polynomial time. If the number of colors m is a constant, then linear time is sufficient.*

Acknoledgement We wish to thank Zsolt Tuza for directing our attention to the PrExt problem and Jens Gustedt and Tamal Dey for useful comments that improved the presentation of the paper.

References

[ALS] S. Arnborg, J. Lagergren and D. Seese, Problems easy for tree--decomposable graphs, in: Proc. 15th ICALP, LNCS 317 (Springer, Berlin, 1988), 38–51.

[AP] S. Arnborg and A. Proskurowski, Linear-time algorithms for NP-hard problems on graphs embedded in k-trees, Discr. Appl. Math. 23 (1989) 11–24.

[BLW] M.W.Bern, E.L. Lawler and A.L. Wong, Linear-time computations of subgraphs of decomposable graphs, J. Algorithms 8 (1987), 216–235.

[BK] H.L. Bodlaender and T. Kloks, Better algorithms for the pathwidth and treewidth of graphs, in: Proc. 18th ICALP, LNCS 510 (Springer, Berlin, 1991) 544–555.

[BHT] M. Biró, M. Hujter and Zs. Tuza, Precoloring Extension. I. Interval Graphs, Discrete Math., to appear.

[CPS] D.G. Corneil, Y. Perl and L.K. Stewart, A linear recognition algorithm for cographs, SIAM J. Comput. 4 (1985) 926–934.

[GJ] M.R. Garey and D.S. Johnson, Computers and Intractability: A Guide to the Theory of NP-Completeness (Freeman, San Francisco, 1979).

[HT] M. Hujter and Zs. Tuza, Precoloring Extension. II. Graph Classes Related to Bipartite graphs, J. Graph Theory, to appear.

[Jan] K. Jansen, Ein Zurdnungproblem im Hardware Design, Dissertation, Universität Trier, FB IV - Mathematik und Informatik, Trier (1990).

[Lag] J. Lagergren, Efficient parallel algorithms for tree-decomposition and related problems, in: Proc. 31st Ann. Symp. FOCS (1990) 173–182.

[Ree] B. Reed, Private Communication.

[RS] N. Robertson and P. Seymour, Graph Minors II. Algorithmic aspects of tree-width, J. Algorithms 7 (1986) 309–322.

[Sch] P. Scheffler, Die Baumweite von Graphen als ein Maß für die Kompliziertheit algorithmischer Probleme, Dissertation A, Report RMATH-04/89, AdW der DDR, K.-Weierstraß-Institut für Mathematik, Berlin (1989).

[Viz] V.G. Vizing, Critical graphs with given chromatic class (Russian), Diskret. Analiz 5 (1965) 9–17.

Reports from the group

"Algorithmic Discrete Mathematics"

of the Department of Mathematics, TU Berlin

401/1994 *Rudolf Müller, Andreas S. Schulz:* The Interval Order Polytope of a Digraph, to appear in *Springer Lecture Notes in Computer Science*, Proceedings of IPCO IV

396/1994 *Petra Scheffler:* A Practical Linear Time Algorithm for Disjoint Paths in Graphs with Bounded Tree-width

394/1994 *Jens Gustedt:* The General Two-Path Problem in time $O(m \log n)$, extended abstract

393/1994 *Maurice Queyranne:* A Combinatorial Algorithm for Minimizing Symmetric Submodular Functions

392/1994 *Andreas Parra:* Triangulating Multitolerance Graphs

390/1994 *Karsten Weihe:* Maximum (s,t)–Flows in Planar Networks in $\mathcal{O}(|V| \log |V|)$ Time

386/1994 *Annelie von Arnim, Andreas S. Schulz:* Facets of the Generalized Permutahedron of a Poset, to appear in *Discrete Applied Mathematics*

383/1994 *Karsten Weihe:* Kurzeinführung in C++

377/1994 *Rolf H. Möhring, Matthias Müller-Hannemann, Karsten Weihe:* Using Network Flows for Surface Modeling

376/1994 *Valeska Naumann:* Measuring the Distance to Series-Parallelity by Path Expressions

375/1994 *Christophe Fiorio, Jens Gustedt:* Two Linear Time Union-Find Strategies for Image Processing

374/1994 *Karsten Weihe:* Edge–Disjoint (s,t)–Paths in Undirected Planar graphs in Linear Time

373/1994 *Andreas S. Schulz:* A Note on the Permutahedron of Series–Parallel Posets, appeared in *Discrete Applied Mathematics* 57 (1995), pp. 85-90

371/1994 *Heike Ripphausen–Lipa, Dorothea Wagner, Karsten Weihe:* Efficient Algorithms for Disjoint Paths in Planar Graphs

368/1993 *Stefan Felsner, Rudolf Müller, Lorenz Wernisch:* Optimal Algorithms for Trapezoid Graphs

367/1993 *Dorothea Wagner:* Simple Algorithms for Steiner Trees and Paths Packing Problems in Planar Graphs, erscheint in *"Special Issue on Disjoint Paths" (eds. B. Gerards, A. Schrijver) in CWI Quarterly*

365/1993 *Rolf H. Möhring:* Triangulating Graphs without Asteroidal Triples

359/1993 *Karsten Weihe:* Multicommodity Flows in Even, Planar Networks

358/1993 *Karsten Weihe:* Non-Crossing Path Packings in Planar Graphs with Applications

357/1993 *Heike Ripphausen-Lipa, Dorothea Wagner, Karsten Weihe:* Linear-Time Algorithms for Disjoint Two-Face Paths Problems in Planar Graphs

354/1993 *Dorothea Wagner, Karsten Weihe:* CRoP: A Library of Algorithms for the Channel Routing Problem

351/1993 *Jens Gustedt:* Finiteness Theorems for Graphs and Posets Obtained by Compositions

350/1993 *Jens Gustedt, Angelika Steger:* Testing Hereditary Properties Efficiently

349/1993 *Stefan Felsner:* 3-Interval Irreducible Partially Ordered Sets

348/1993 *Stefan Felsner, Lorenz Wernisch:* Maximum k-Chains in Planar Point Sets: Combinatorial Structure and Algorithms

345/1993 *Paul Molitor, Uwe Sparmann, Dorothea Wagner:* Two-Layer Wiring with Pin Preassignments