Alcom-FT Technical Report Series
ALCOMFT-TR-01-142

Counting H-colorings of Partial k-trees*

Josep Díaz† Maria Serna† Dimitrios M. Thilikos†‡

Abstract

The problem of counting all H-colorings of a graph G of n vertices is considered. While the problem is, in general, #P-complete, we give linear time algorithms that solve the main variants of this problem when the input graph G is a k-tree or, in the case where G is directed, when the underlying graph of G is a k-tree. Our algorithms remain polynomial even in the case where $k = O(\log n)$. That way, we establish the first positive result on counting H-colorings of non trivial graph classes. Moreover, as far as the authors know, it is the first time treewidth is used to solve a counting problem, that is, in its full generality, #P-complete. Our results imply the existence of polynomial time algorithms for a series of problems on partial k-trees such as core checking and chromatic polynomial computation.

1 Introduction

In this paper we consider a series of counting problems associated with various versions of homomorphisms from a graph G to a fixed graph H. Given a graph $H = (G(H), E(V))$, with loops but without multiple edges, for any graph $G = (V(G), E(G))$ an *homomorphism* of G to H is a map $\theta : V(G) \to V(H)$ with the property that $\{v, w\} \in E(G) \Rightarrow \{\theta(v), \theta(w)\} \in E(H)$. We will call such a homomorphism a *H-coloring* of G (for an example, see Figure 1). For H fixed, the *H-coloring problem* asks if there is an H-coloring of G. If H is bipartite or it has a loop, then the H-coloring problem can be trivially solved in polynomial time, but in the case that H is not-bipartite and loopless the problem is known to be NP-complete [HN90] (see also [GHN00] for the complexity of the same problem for bounded degree graphs).

An earlier version of the H-coloring problem is the *exact H-coloring problem*: given a graph G, decide if there is an H-coloring θ of G such that $\theta(V(G)) = V(H)$ and $\theta(E(G)) = E(H)$. This problem appeared in [GJ79] as problem GT-52 and is known to be NP-complete, even

*This research was partially supported by the IST programe of EU under contract number IST-1999-14186 (ALCOM-FT).

†Departament de Llenguatges i Sistemes Informàtics. Universitat Politècnica de Catalunya. Campus Nord Mòdul C6. c/ Jordi Girona Salgado 1-3, 08034, Barcelona, Spain. E-mails: {diaz,mjserna,sedthilk}@lsi.upc.es

‡The research of the third author was supported by the Ministry of Education and Culture of Spain, Grant number MEC-DGES SB98 0K148809.

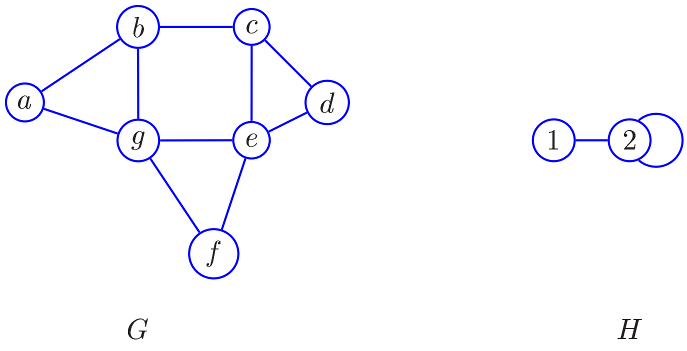

Figure 1: An example of an H-coloring of G is the mapping $\theta(a) = \theta(d) = \theta(f) = 1$ and $\theta(b) = \theta(c) = \theta(e) = 2$.

for the case where H is a triangle [Lev73]. We also consider the intermediate problem of asking whether there is an H-coloring θ of G such that $\theta(V(G)) = V(H)$ and we call it *vertex exact H-coloring*.

Given a graph G, let $\mathcal{H}(G, H)$ denote the set of all H-colorings of G, and let $\mathcal{E}(G, H)$ denote the set of all exact H-colorings of G. Also, we denote as $\mathcal{V}(G, H)$ the set of all vertex H-colorings of G. For a fixed graph H, denote by *#H-coloring* the problem of, given G as input, computing $|\mathcal{H}(G, H)|$. Also, denote by *#EH-coloring* the problem of, given as input G, computing $|\mathcal{E}(G, H)|$ and by *#VEH-coloring* the problem of, given as input G, computing $|\mathcal{V}(G, H)|$. Dyer and Greehill [DG00] have proved that unless every connected component of H is an isolated vertex without a loop, a complete graph with all loops present or a complete unlooped bipartite graph, the #H-coloring problem is #P-complete, even when G has bounded degree. It is easy to see that the same result hold for the #EH-coloring (#VEH-coloring) as the knowledge of $\mathcal{E}(G, H')$ ($\mathcal{V}(G, H')$) for any (induced) subgraph H' of H implies the knowledge of $\mathcal{H}(G, H)$. For further negative results on the approximability of the #H-coloring see [CDF] and [DFJ99]. For positive and negative complexity results concerning special cases of the #H-coloring problem, depending on restrictions on both H and G, see [Edw86, DNS00].

The notion of *treewidth* appears to play a central role in many areas of graph theory. Roughly, a graph has small treewidth if it can be constructed by assembling small graphs together in a tree structure, namely a tree decomposition of small width (see section 2 for the formal definitions). It was first defined by Robertson and Seymour in [RS84] and served as one of the cornerstones of their lengthy proof of the Wagner conjecture, known now as the Graph Minors Theorem (for a survey see [RS85]). Treewidth appears to have interesting applications in algorithmic graph theory. In particular, a wide range of, otherwise intractable, combinatorial problems are polynomially, even linearly, solvable when restricted to graphs with bounded treewidth. Alternatively, graphs with bounded treewidth are called *partial k-trees*. The standard methodology to get such solutions consists of a two step procedure: First to find a tree-decomposition of the input graph, with treewidth bounded by a constant, although possibly not optimal, and second to use some kind of *dynamic programming* taking advantage of the bounded treewidth decomposition of the graph, to get a solution (for a survey, see [Bod97, Arn85]). In particular, such a dynamic programing technique, solving – among others – the H-coloring problem for partial k-trees, has been developed in [TP97]. For the first canonical step, Bodlaender in [Bod96] presented, for any k, a linear time algorithm that, given a graph G, checks whether G is a partial k-tree and, if so, outputs an mimimum width tree decomposition. As this algorithm appears extremely hard to be implemented (see also [BK96]), the first canonical step can be carried out by one of the previously developoed algorithms that are able to output a tree-decomposition of G with treewidth bounded by a constant (linear in k) [Ree92, Lag96, MT92]. In particular, the deterministic algorithm by Reed [Ree92] runs in $O(n \log n)$ time, which, for constant k, gives a 2 approximation to the optimal treewidth-decomposition. Moreover, its implementation is easier than the algorithm in [Bod96]. Consequently, we will assume that any partial k-tree is always accompanied with its linear size tree decomposition of bounded width.

In this paper we present a polynomial time algorithm for the counting problems $\#H$-coloring, $\#EH$-coloring, and $\#VEH$-coloring, in the case that the input graph G has constant treewidth. Although our methodology follows the two cannonical steps, we shall remark that, as far as the authors know, it is the firt time treewidth is used to solve a counting problem, that is, in its full generality, #P-complete. In each case, we present an easy to be implemented algorithm for the second step, asumming the aforementioned results on the existence of fast and implementable algorithms for obtaining a bounded treewidth-decomposition for a partial k-tree G. Finally, we stress out that our results are the first non-trivial positive results on counting H-colorings for the case where H is generic and G is restricted.

In Section 2 we review the basic definitions and some preliminary results on treewith. In Section 3 we present the algorithm solving the $\#H$-coloring problem. In Section 4 we present two variants of the algorithm of Section 3, for solving the $\#VEH$-coloring and the $\#EH$-coloring problem respectively. In the final section, we present some consequences of our previous results for a series of problems on graphs with bounded treewidth.

2 Definitions and basic results

Robertson and Seymour [RS83, RS86] introduced the notion of treewidth in the framework of the theory of graph minors.

Definition 2.1. A *tree decomposition* of a graph G is a pair (X, U) where U is a tree whose vertices we will call *nodes* and $X = (\{X_i \mid i \in V(U)\})$ is a collection of subsets of $V(G)$ such that

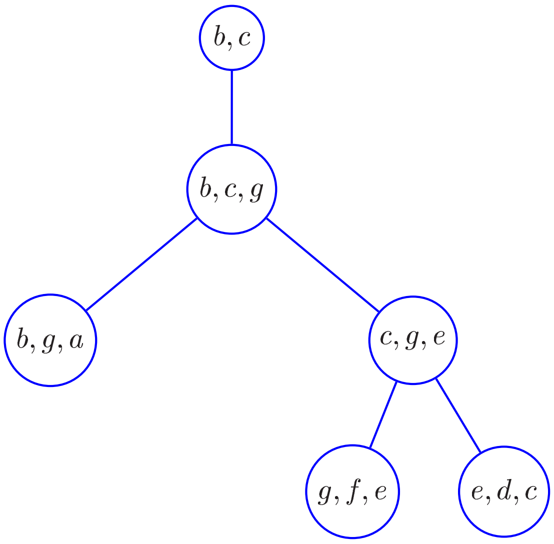

Figure 2: A tree decomposition of G in Figure 1

1. $\bigcup_{i \in V(U)} X_i = V(G)$,

2. for each edge $\{v, w\} \in E(G)$, there is an $i \in V(U)$ such that $v, w \in X_i$, and

3. for each $v \in V(G)$ the set of nodes $\{i \mid v \in X_i\}$ forms a subtree of U.

The *width* of a tree decomposition $(\{X_i \mid i \in V(U)\}, U)$ equals $\max_{i \in V(U)}\{|X_i| - 1\}$. The *treewidth* of a graph G is the minimum width over all tree decompositions of G.

Computing treewidth is an NP-complete problem [ACP93]. In Figure 2, we present an optimal treewidth decomposition of the graph G in Figure 1.

The following extensions of tree decomposition were introduced by Boadlaender and Kloks [Bod96, BK96, Klo94] (see also [Arn85, Bod96]),

Definition 2.2. A *rooted* tree decomposition is a triple $D = (X, U, r)$ in which U is a tree rooted at r and (X, U) is a tree decomposition.

Definition 2.3. Let $D = (X, U, r)$ be a rooted tree decomposition of a graph G where $X = \{X_i \mid i \in V(U)\}$. D is called a *nice* tree decomposition if the following are satisfied:

1. Every node of U has at most two children,

2. if a node i has two children j, h then $X_i = X_j = X_h$,

3. if a node i has one child, then either $|X_i| = |X_j| + 1$ and $X_j \subset X_i$ or $|X_i| = |X_j| - 1$ and $X_i \subset X_j$.

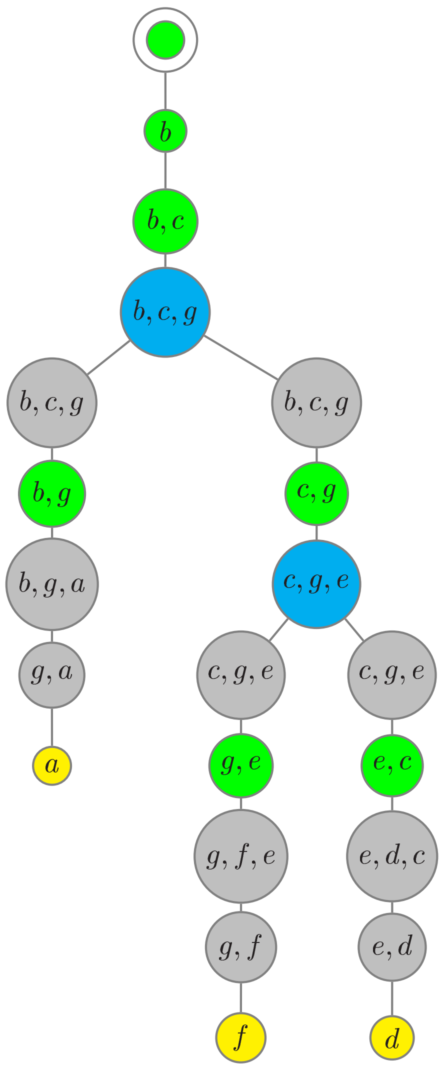

Figure 3: A nice tree decomposition of G

The following lemma can be found in [BK96] (see also [Klo94]).

Lemma 2.1. For any constant $k \geq 1$, given a tree decomposition of a graph G of width $\leq k$ and $O(n)$ nodes, where n is the number of vertices in G, there exists an $O(n)$ algorithm that constructs a nice tree decomposition of G of width less or equal than k and with at most kn nodes.

In Figure 3 we present a nice tree decomposition for the graph G in Figure 1. It has been constructed from the tree decomposition presented in Figure 2.

We observe that a nice tree decomposition $D = (\{X_p \mid p \in V(U)\}, U, r)$ contains nodes of the following four possible types. A node $p \in V(U)$ is called:

Start, if p is a leaf,

Join, if p has two children $q_i, i = 1, 2$,

Forget, if p has only one child q and $|X_p| = |X_q| - 1$,

Introduce, if p has only one child q and $|X_p| = |X_q| + 1$.

Notice that for every start node we may assume that $|X_p| = 1$: the effect of *start* nodes with $|X_p| > 2$ can be obtained by using a *start* node with a set containing one vertex, and then $|X_p| - 1$ *introduce* nodes, which add all the other vertices. For the purposes of this paper we will assume that the root r is a *forget* node with $X_r = \emptyset$.

Taking in mind Lemma 2.1, we will assume that any partial k-tree is given along with a nice tree decomposition of it.

Let $D = (X, U, r)$ be a nice tree decomposition of a graph G. For each node p of D, let U_p be the subtree of U, rooted at node p. We set $V_p = \cup_{v \in V(U_p)} X_v$ and, for any $p \in V(U)$, we define $G_p = G[V_p]$. Notice that $G_r = G$.

3 An algorithm for counting the number of H-colorings

Let H be a fixed graph. Given an input graph G together with a nice tree decomposition $D = (X, U, r)$ of it, we wish to count $|\mathcal{H}(G, H)|$, i.e. to solve the $\#H$-coloring problem for G.

The strategy will be a dynamic programming approach. To optimize the space used in the counting part, we will preprocess the tree. The preprocessing will produce an ordering of the nodes in the nice tree decomposition.

Given a rooted binary tree T, let b be the number of vertices with degree two in T.

Definition 3.1. An ordering $u_1, \dots, u_s$ of $V(T)$ is *stingy* if the following conditions hold.

1. u_1 is a leaf and u_n is the root.
2. For any node u_i, its parent appears at a position $j > i$.
3. For any j the number of nodes in the set $\{u_1, \dots, u_j\}$ whose parent appears at a position $k > j$ is at most $\log b$.

Given a nice tree decomposition $D = (X, U, r)$ it is straightforward to compute a stingy ordering of the vertices of U.

Lemma 3.1. Given a rooted binary tree T a proper ordering of the l nodes in $V(T)$ can be computed using $O(n \log n)$ space in $O(n)$ steps.

The algorithm will do a *traversal* of D following a stingy sequence, computing and tabulating for each $p \in V(U)$ the appropriate information. Before describing it, we need some additional definitions.

Given a nice tree decomposition $D = (X, U, r)$ of a graph G, for each $p \in V(U)$, let $F_p = \{\varphi : X_p \to V(H)\}$. Notice that if G has treewidth k, then for any $p \in V(U)$, $|F_p| \leq h^{k+1}$. Moreover, for the root we have $F_r = \{\varnothing\}$.

The table associated to a node $p \in V(U)$ will have an entry for each $\varphi \in F_p$, holding the value $I_p(\varphi) = |\{\theta \in \mathcal{H}(G_p, H) \mid \theta|_{X_p} = \varphi\}|$ (in general, if θ is a function, we denote $\theta|_S = \{(v, a) \in \theta \mid v \in S\}$). As we always have $\theta|_\emptyset = \varnothing$, we get $|\mathcal{H}(G, H)| = I_r(\varnothing)$.

function count-H(G, H, D, S)

$D = (X, U, r)$ is a nice tree decomposition of G with width k
$n = |V(G)|$, $h = |V(H)|$, $s = |V(U)|$
$S = (p_1, \ldots, p_s)$ is a stingy ordering of $V(U)$

```
begin
    for p := p_1 to p_s do
        if p is a start node, with X_p = {v} then
            for all a ∈ V(H) do I_p((v, a)) := 1
        end
        if p is a introduce node then
            let q be its unique child
            v := X_p − X_q
            S_q := {u ∈ X_q | {u, v} ∈ E(G_p)}
            for all φ ∈ F_q and a ∈ V(H) do
                if ∀_{u∈S_q} {φ(u), a} ∈ E(H) then
                    I_p(φ ∪ {(v, a)}) := I_q(φ)
                else I_p(φ ∪ {(v, a)}) := 0
                end
            end for
            erase the information on node q
        end
        if  is a forget node then
            let q be its unique child
            v := X_q − X_p
            for all φ ∈ F_p do
                I_p(φ) := Σ_{a∈V(H)} I_q(φ ∪ {(v, a)})
            end for
            erase the information on node q
        end
        if p is a join node with children q_1, and q_2 then
            for all φ ∈ F_p do
                I_p(φ) := I_{q_1}(φ) · I_{q_2}(φ)
            end for
            erase the information on nodes q_1 and q_2
        end
    end for
    return I_r(∅)
end
```

Figure 4: An algorithm for counting H-colorings

The function count-H, given in Figure 4, takes as input G, D and a stingy sequence $S = (p_1, \dots, p_s)$ of $V(U)$ and outputs the value $I_r(\varnothing)$.

Theorem 3.1. Given a graph G with $n = |V(G)|$, a nice tree decomposition $D = (X, U, r)$ of G with width k, and a stingy ordering of the nodes in D. Then the algorithm in Figure 4 computes the number of H-colorings of G in $O(nkh^{k+1} \min\{k, h\})$ steps using $O(h^{k+1} \log n)$ additional space, where $h = |V(H)|$.

Proof. Let us first prove the correcteness of the proposed algorithm. The algorithm starts by computing the table for the first leaf, a *start* node, in the stingy sequence. Notice that by condition 2 in Definition 3.1, whenever we process a node $p \in V(U)$ its descendants have been already processed and the associated information stored. So, we have only to show that, for each type of node, our algorithm computes the corresponding values using the correct information stored in the tables of its descendants.

When X_p is a *start* node the computation is correct as G_p is formed by an unique vertex.

If p is an *introduce* node, with $\{v\} = X_p - X_q$, the graph G_p is obtained from G_q adding the new vertex v and the edges in the set $S_q = \{(u, v) \mid u \in S_q\}$, therefore only those extensions that preserve the H-coloring condition survive and therefore counted.

If p is a *forget* node, $X_p = X_q - \{v\}$, then $G_p = G_q$, and therefore, for each $\varphi \in F_p$, we only have to accumulate the number of colorings counted for any extension of φ that is a valid coloring of X_q.

When the computation arrives to a *join* node p, both of its children q_1, q_2 must have been already tabulated. Notice that the graph G_p is obtained as the union of the graphs G_{q_1} and G_{q_2}. Recall that G_{q_1} and G_{q_2} have no edges in common out of those in $X_p = X_{q_1} = X_{q_2}$. In such a case any H-coloring of G_p is the combination of an H-coloring of G_{q_1} with an H-coloring of G_{q_2} with the same images on the vertices of X_p.

When the algorithm arrives to the last node, the root, the only value in its table will be the number of H-coloring of $G_r = G$

Let us analyze now the algorithm's complexity. For any $p \in V(U)$ the number of elements in F_p is at most h^{k+1}. Moreover $|V(U)| = O(kn)$. Therefore, the internal operations computing one entry for the cases of a *start*, *forget*, or *join* node, cost time $O(h)$. In the case, of an *introduce* node, we can implement the computation either in $O(h)$ or $O(k)$ steps depending on whether we represent the set S_q by the set of vertices of G or the set of their images in H. Therefore the time complexity follows.

To compute an upper bound on the required space, we have to take into account that the number of *join* nodes in a nice tree decomposition is at most n. Therefore, condition 3 in Definition 3.1 guarantees that the number of tables maintained at each step is at most $O(\log n)$. As the size of a table is at most h^{k+1}, the space complexity follows. □

Notice that if h is constant, we can allow $k = O(\log n)$ and still have polynomial time. Moreover, for constant treewidth k, we can allow $h = O(n)$ and still have a polynomial time algorithm. As an example of this we have the following result, which is based on the fact that outerplanar graphs have treewidth ≤ 2.

Corollary 3.1. There exists an algorithm that outputs the number of all the homomorphisms between any pair of outerplanar graphs in $O(n^4)$ steps and between any pair of forests in $O(n^3)$ steps.

Taking in mind the algorithm of Bodlaender in [Bod96] we have the following.

Corollary 3.2. Given a graph G if it has treewidth bounded by a constant k, then we can compute $|\mathcal{H}(G,H)|$ in $O(nkh^{k+1}\min\{k,h\}) + O(f(k)n)$ steps.

For constant k and h, our algorithm works in linear time and therefore we have the following.

Corollary 3.3. The $\#H$-coloring problem, with parameters the treewidth of the input and the size of H, is a fixed parameter tractable problem (i.e. belongs to the class FPT).

4 Counting the number of (vertex) exact H-colorings

Let H be a fixed graph. Given a graph G together with a nice tree decomposition $D = (X, U, r)$ of G and a stingy sequence of U, we wish to count $|\mathcal{V}(G,H)|$ and $|\mathcal{E}(G,H)|$, i.e. to solve the $\#VEH$-coloring and the $\#EH$-coloring problems.

The strategy will be similar as in the previous algorithm, a dynamic programming guided by a stingy sequence. The main modification will be the information stored in the tables.

4.1 The $\#VEH$-coloring

The table associated to a node $p \in V(U)$ will have an entry for each pair (φ, R) where $\varphi \in F_p$ and $R \subseteq V(H)$ holding the value

$$I_p(\varphi, R) = |\{\theta \in \mathcal{E}(G_p, J) \mid \theta|_{X_p} = \varphi \text{ and } \theta(V(G_p)) = R\}|.$$

Therefore, $|\mathcal{V}(G,H)| = I_r(\varnothing, V(H))$.

The function count-exact-V-H, given in Figure 5, takes as input G, D and a stingy sequence $S = (p_1, \dots, p_s)$ of $V(U)$ and outputs the value $I_r(\varnothing, V(H))$.

Theorem 4.1. Given a graph G with $n = |V(G)|$, a nice tree decomposition $D = (X, U, r)$ of G with width k, and a stingy ordering of the nodes in D. Then the algorithm in Figure 5 computes the number of vertex exact H-colorings of G in $O(nkh^{k+1}2^h \max\{k, 2^{2h}\})$ time using $O(h^{k+1}2^h \log n)$ additional space, where $h = |V(H)|$.

function count-exact-V-H(G, H, D, S)
$D = (X, U, r)$ is a nice tree decomposition of G with width k
$n = |V(G)|$, $h = |V(H)|$, $s = |V(U)|$
$S = (p_1, \ldots, p_s)$ is a stingy ordering of $V(U)$
begin
    for $p := p_1$ **to** p_s **do**
        if p is a *start node*, with $X_p = \{v\}$ **then**
            for all $R \subseteq V(H)$ **and** $a \in V(H)$ **do**
                if $R = \{a\}$ **then** $I_p((v, a), R) = 1$ **else** $I_p((v, a), R) = 0$
        end
        if p is a *introduce node* **then**
            let q be its unique child
            $v := X_p - X_q$
            $S_q := \{u \in X_q \mid \{u, v\} \in E(G_p)\}$
            for all $\varphi \in F_q$ **and** $R \subseteq V(H)$ **and** $a \in V(H)$ **do**
                if $a \in R$ **and** $\forall u \in S_q\,[(\varphi(u), a) \in E(H)]$ **then**
                    $I_p(\varphi \cup \{(v, a)\}, R) := (I_q(\varphi, R) + I_q(\varphi, R \setminus \{a\}))$
                else $I_p(\varphi \cup \{(v, a)\}, R) := 0$
                end
            end for
            erase the information on node q
        end
        if p is a *forget* node **then**
            let q be its unique child
            $v := X_q - X_p$
            for all $\varphi \in F_q$ **and** $R \subseteq V(H)$ **do**
                $I_p(\varphi, R) := \sum_{a \in V(H)} I_q(\varphi \cup \{(v, a)\}, R)$
            end for
            erase the information on node q
        end
        if p is a *join* node with children q_1 and q_2 **then**
            for all $\varphi \in F_p$ **and** $R \subseteq V(H)$ **do**
                $I_p(\varphi, R) := 0$
                for all R_1 **and** R_2 **with** $R_1 \cup R_2 = R$ **do**
                    $I_p(\varphi, R) := I_p(\varphi, R) + I_{q_1}(\varphi, R_1) \cdot I_{q_2}(\varphi, R_2)$
                end for
            end for
            erase the information on nodes q_1 and q_2
        end
    end for
    return $I_r(\varnothing, V(H))$
end

Figure 5: An algorithm for counting vertex exact H-colorings

Proof. As in de proof of Theorem 3.1, we have to prove the correctness for each type of node. The arguments are similar, we only have to take into account not only the difference between G_p and the subgraphs associated to its descendants, but also the ways in which a given subset of vertices of H can be covered.

The easiest cases are the *start* and *forget* nodes. For the first, only one vertex of H can be covered, and the last will not cover any additional vertex of H.

If p is an *introduce* node, with $\{v\} = X_p - X_q$, the graph G_p is obtained from G_q by adding v and the corresponding edges. The only surviving mapping are those that preserve the H-coloring condition. Furthermore, when we determine the image of v three possible cases arise: the color do not appear in the set R of covered colors, it is new and was not present in the set covered by node q or it is old and appears in the set covered by node q. In the first case the total value is zero, otherwise we have to add all the colorings covering R from the two different possibilities.

When the computation arrives to a *join* node p, both of its children q_1, q_2 must have been already tabulated. The restriction to keep the coloring condition is the same; from the cover part we have to consider all the ways to split a set $R \subseteq V(H)$ into two partial covers one arising from each child. Note that, in the splitting, we can have $R_1 \cap R_2 \neq \emptyset$.

Let us analyze the complexity. For any $p \in V(U)$, the number of elements in F_p is at most h^{k+1}, and the number of possible subsets of $|V(H)|$ is 2^h. Moreover $|V(U)| = O(kn)$. The most costly cases are the *introduce* node, which needs $O(k)$ steps for each entry and a *join* node which needs $O(2^{2h})$ steps for each entry. Therefore, the time complexity follows. The additional space bound follows from the table size and the fact that the stingy sequence restricts the necessary space resources on each step to $O(\log n)$ tables. □

Corollary 4.1. Given a graph G with n vertices and treewidth bounded by a constant k, then we can compute $|\mathcal{V}(G, H)|$ in $O(nkh^{k+1}2^h \max\{k, 2^{2h}\}) + O(f(k)n)$ time, where $h = |V(H)|$.

4.2 The $\#EH$-coloring

The table associated to a node $p \in V(U)$ will have an entry for each pair (φ, J) where $\varphi \in F_p$ and J is a subgraph of H, holding the value

$$I_p(\varphi, J) = |\{\theta \in \mathcal{E}(G_p, J) \mid \theta|_{X_p} = \varphi \text{ and } \theta(V(G_p)) = V(J) \text{ and } \theta(E(G_p)) = E(J)\}|.$$

Therefore, $|\mathcal{E}(G, H)| = I_r(\varnothing, H)$.

The function count-exact-H, given in Figure 6, takes as input G, D and a stingy sequence $S = (p_1, \dots, p_s)$ of $V(U)$ and outputs the value $I_r(\varnothing, H)$. We will use $\mathcal{S}(H)$ to denote the set of subgraphs of H.

Theorem 4.2. Given a graph G with $n = |V(G)|$, a nice tree decomposition $D = (X, U, r)$ of G with width k, and a stingy ordering of the nodes in D. Then the algorithm in Fig-

function count-exact-H(G, H, D, S)
$D = (X, U, r)$ is a nice tree decomposition of G with width k
$n = |V(G)|$, $h = |V(H)|$, $e = |E(H)|$, $s = |V(U)|$
$S = (p_1, \ldots, p_s)$ is a stingy ordering of $V(U)$
begin
  for $p := p_1$ **to** p_s **do**
    if p is a *start node*, with $X_p = \{v\}$ **then**
      for all $J \in \mathcal{S}(H)$ **and** $a \in V(H)$ **do**
        if $V(J) = \{a\}$ **then** $I_p((v, a), J) = 1$ **else** $I_p((v, a), J) = 0$
    end
    if p is a *introduce node* **then**
      let q be its unique child
      $v := X_p - X_q; S_q := \{u \in X_q \mid \{u, v\} \in E(G_p)\};$
      for all $\varphi \in F_q$ **and** $J \in \mathcal{S}(H)$ **and** $a \in V(H)$ **do**
        if $\{\{a, \varphi(u)\} \mid u \in S_q\} \subseteq E(J)$ **then**
          if $\forall \{a, \varphi(v)\} \in E(J)\ \exists u \in X_q : \varphi(u) = a$ **then**
            $I_p(\varphi, J) := I_q(\varphi|_{X_q}, J - \{\varphi(v)\})$
          else $I_p(\varphi, J) := 0$
          end
          for all $A \subseteq \{\{a, \varphi(u)\} \mid u \in S_q\}$ **with** $\forall \{a, b\} \in E(J) - A\ \exists u \in X_q : \varphi(u) = b$ **do**
            $I_p(\varphi, J) := I_p(\varphi, J) + I_q(\varphi|_{X_q}, (V(J), E(J) - A))$
        else $I_p(\varphi, J) := 0$
        end
      end for
      erase the information on node q
    end
    if p is a *forget* node **then**
      let q be its unique child
      $v := X_q - X_p$
      for all $\varphi \in F_p$ **and** $J \in \mathcal{S}(H)$ **do**
        $I_p(\varphi, J) := \sum_{a \in V(J)} I_q(\varphi \cup \{(v, a)\}, J)$
      erase the information on node q
    end
    if p is a *join* node with children q_1 and q_2 **then**
      for all $\varphi \in F_p$ **and** $J \in \mathcal{S}(H)$ **do**
        $I_p(\varphi, J) := 0$
        for all $J_1, J_2 \in \mathcal{S}(H)$ **with** $J_1 \cup J_2 = J$ **and** $\varphi(X_p) \subseteq V(J_1) \cap V(J_2)$ **do**
          $I_p(\varphi, J) := I_p(\varphi, J) + I_{q_1}(\varphi, J_1) \cdot I_{q_2}(\varphi, J_2)$
      end for
      erase the information on nodes q_1 and q_2
    end
  end for
  return $I_r(\varnothing, H)$
end

Figure 6: An algorithm for counting exact H-colorings

ure 6 computes the number of H-colorings of G in $O(nkh^{k+1}2^{h+e}\max\{k, 2^{2(h+e)}\})$ time using $O(h^{k+1}2^{h+e}\log n)$ additional space, where $h = |V(H)|$ and $e = |E(H)|$.

Proof. The arguments are similar to those in the proof of Theorem 4.1. We have to further take into account the ways in which a given subgraph of H can be covered.

The easiest cases are the *start* and *forget* nodes. For the first, only one vertex of H can be covered, and the last will not cover any additional vertex or edge.

If p is an *introduce* node, with $\{v\} = X_p - X_q$, as before only those mapping extensions that preserve the H-coloring condition must survive. Furthermore, when we determine the image of v, it may happen that either the colors of one of the images of the introduced edges are not present in the subgraph or that the colors of all the images are present. In the first case, the number of colorings is zero. We have two types of decomposition: from subgraphs without the image of v or from subgraphs with the image of the same vertex set and less edges. In both cases we must insure that the edges not present are covered by the actual mapping.

When the computation arrives to a *join* node p, we have to consider all the ways to split a subgraph of H into two subgraphs, each covered by a child. Note that in the splitting we can have two subgraphs with non-empty intersection, but both must contain the common image of X_p.

Let us analyze the complexity. For any $p \in V(U)$ the number of elements in F_p is at most h^{k+1}, and the number of possible subgraphs of H, that is is 2^{h+e}. Moreover $|V(U)| = O(kn)$. Again, the most costly cases are the *introduce* node, which needs $O(k + 2^e)$ steps and the *join* node which needs $2^{2(h+e)}$ time. The time complexity follows. The additional space bound follows from the table size and the fact that the stingy sequence restricts the necessary space resources on each step to $O(\log n)$ tables. □

Corollary 4.2. Given a graph G with n vertices and treewidth bounded by a constant k, then we can compute $|\mathcal{E}(G, H)|$ in $O(nkh^{k+1}2^{h+e}\max\{k, 2^{2(h+e)}\} + O(f(k)n)$ time, where $h = |V(H)|$ and $e = |E(H)|$.

Notice that for each of the algorithms given so far, if we retain the information of all the tables, it is possible to use it in order to enumerate all the homomorphisms. The storing of all the tables implies a burden of $O(kn/\log n)$ to the space reported in Theorems 3.1, 4.1, and 4.2. In particular, setting up a suitable bookkeeping of the enumerated homomorphisms, a top-down traversal of the table information can pop-up each of them in $O(kn)$ steps.

Finally, we point out that in both algorithms of this section and for constant h, we can allow $k = O(\log n)$ and still have polynomial time.

4.3 The directed case

Given two directed graphs $\vec{H}$ and $\vec{G}$, an *$\vec{H}$-coloring of $\vec{G}$* is any function $\theta : V(\vec{G}) \to V(\vec{H})$ with the property that $(v, w) \in E(\vec{G}) \Rightarrow (\theta(v), \theta(w)) \in E(\vec{H})$. If $\vec{H}$ is a fixed directed graph, the $\vec{H}$*-coloring problem* asks, given a directed graph $\vec{G}$ as input, whether there is a $\vec{H}$-coloring of $\vec{G}$. We define the $\#\vec{H}$*-coloring problem* as the problem of, given a directed graph $\vec{G}$, counting all the $\vec{H}$-colorings of $\vec{G}$. The $\#E\vec{H}$*-coloring problem* and the $\#VE\vec{H}$*-coloring problem* are defined analogously.

It can be directly verified that all the algorithms and the proofs of this paper work also in the case where G and H are directed. In particular, the corresponding algorithms and proofs can be derived from the existing ones if we consider a nice tree decomposition of the underlying graph of $\vec{G}$ and replace any undirected edge $\{x, y\}$ of G or H by a directed edge (x, y). The times of the corresponding algorithms are the same as those in Theorems 3.1, 4.2, and 4.1. In short, we state the following result.

Theorem 4.3. *The* $\#\vec{H}$*-coloring problem, the* $\#E\vec{H}$*-coloring problem, and the* $\#VE\vec{H}$*-coloring problem are all fixed parameter tractable, with parameters the treewidth of the underlying graph of the input and the number of vertices of* $\vec{H}$*.*

5 Extensions to other results

In this section, we present some consequences of the polynomial time algorithms given in the previous sections. For each problem, either we give a new polynomial-time result or we improve previously known results. Recall that if we are given a graph G, directed or undirected, all of our algorithms should be used in conjuction with one of the algorithms mentioned in the introduction, that in polynomial time, compute the treewidth decomposition of bounded width.

5.1 Coloring problems

Notice that a graph G is H-colorable if $|\mathcal{H}(G, H)| \geq 1$. This implies the following corollary of Theorems 3.1, 4.1, and 4.2 for the corresponding decision version of the problems examined.

Corollary 5.1. *For any graph* H *with* h *vertices and* e *edges, we can solve, for an* n*-vertex partial* k*-tree as input graph,*

- *the* H*-coloring problem in* $O(nkh^{k+1}\min\{k, h\})$ *steps,*
- *the vertex exact* H*-coloring problem in* $O(nkh^{k+1}2^h\max\{k, 2^h\})$ *steps, and*
- *the exact* H*-coloring problem in* $O(nkh^{k+1}2^{h+e}\max\{k, 2^{h+e}\})$ *steps.*

Corollary 5.1 improves the time of the best known algorithm for the H-coloring of partial k-trees. This algorithm runs in $O(nh^{2(k+1)})$ steps and is due to Telle and Proskurowski in [TP97].

We denote as K_c the complete graph with c vertices. As the problem of counting the number of c-colorings of a graph G is equivalent to the $\#K_c$-coloring problem we can conclude to the following.

Corollary 5.2. An algorithm can be constructed to compute the number of c-colorings of a partial k-tree of n vertices in $O(nkc^{k+1}\min\{k,c\})$ steps.

We denote as $\beta_G(c)$ the function mapping c to the number of colorings of G that use exactly c colors, i.e. $\beta_G(c) = |\mathcal{V}(G, K_c)|$.

Lemma 5.1. For any graph G, $\beta_G(c) = |\mathcal{H}(G, K_c)| - \sum_{1\le r\le c-1} \binom{c}{r}\beta_G(r)$.

Proof. Notice that for any $r, 1 \le r \le c$, the number of functions $\phi \in \mathcal{H}(G, K_c)$ where $|\phi(V(G))| \le r$ is equal to $\binom{c}{r}\beta_G(r)$. The sum of these numbers for $r = 1, \ldots, c$ gives $|\mathcal{H}(G, K_c)|$. Therefore $|\mathcal{H}(G, K_c)| = \sum_{1\le r\le c} \binom{c}{r}\beta(r)$ and the lemma follows. $\square$

The formula of Lemma 5.1 and Corollary 5.2 give a way to compute $\beta_G(c)$.

Corollary 5.3. An algorithm can be constructed that, given partial k-tree G with n vertices and for any $c, 1 \le c \le n$, computes $\beta_G(c)$ in $O(nkc^{k+2}\min\{k,c\})$ steps.

Corollary 5.3 can be rewritten as follows.

Corollary 5.4. An algorithm can be constructed that computes the chromatic polynomial of a partial k-tree with n vertices in $O(k^2n^{k+3})$ steps.

In our knowledge, the best algorithm for computing the chromatic polymonial for partial k-trees can be derived from the algorithm of Andrzejak [And98]. This algorithm can compute in $O(n^{2+7\log_2 c})$ steps the Tutte polynomial of a partial k-tree of n vertices, where c is twice the number of partitions of a set with $3(k+1)$ elements. Several researchers have considered the problem of counting the number of proper c-colorings in a graph G. The problem is #P-hard for $c \ge 3$ and maximum degree $\Delta \ge 3$ [Jer95]. In the same paper, it is proved that there exists a Fully Polynomial time Randomized Aproximation Scheme (FPRAS) for the number of colorings in the case when $c \ge 2\Delta + 1$. Recently, Bubley et all. in [BDGJ99] proved that the problem is #P-hard for fixed Δ, but there is a FPRAS for $c = 5$ and $\Delta = 3$. Edwards [Edw86] proved that if $c \ge 3$ and the minimum degree where $\delta \ge \alpha n$, $n = |V(G)|$, the counting problem is #P-complete if $\alpha < \frac{c-2}{c-1}$, but it is in P for $\alpha > \frac{c-2}{c-1}$.

5.2 Problems on cores

Given a graph G, a*core* of G is a subgraph C of G such that G is homomorphic to C, but G fails to be homomorphic to any proper subgraph of C. This notion of core is due to Hell and Nešetřil [HN92]. A graph G is *a core* if G is a core for itself or, equivalently, if $|\mathcal{E}(G, G)| = |\mathcal{H}(G, G)|$. For example, any graph C_{2k+1} is a core (C_{2k+1} is the connected 2-regular graph with $2k + 1$ vertices). It is known, that in general, the problem of deciding whether G is a core is NP-complete [HN92] and in the same paper it is presented a polynomial time algorithm, to decide if G is a core, for the particular case when G has *independence number* $\alpha(G) \le 2$. Finally, we mention that "almost all graphs" are cores [Neš99, Neš00].

We first prove the following.

Corollary 5.5. An algorithm can be constructed that decides, in $O(k^3 n^{k+3})$ steps, whether a partial k-tree with n vertices is a core.

Proof. Notice that it is enough to check whether there exists a subgraph H of G where $|E(H)| = |E(G)| - 1$ and such that $|\mathcal{H}(G,H)| \geq 1$. According to Theorem 3.1, this requires $O(|E(G)|k^2 n^{k+2}) = O(k^3 n^{k+3})$ steps (take in mind that if G is a partial k-tree then $E(G) = O(k|V(G)|)$). □

We mention that the property of being a core is an EMS-property (i.e. involves counting or summing evaluations over sets definable on monadic second order logic) and therefore, the results in [Cou90, ALS91, AF93] imply the existence of a polynomial time algorithm for deciding it. So far, no explicit algorithm has been reported for this problem.

We consider the isomorphism problem on cores. Notice that graphs G and H are isomorphic iff $|\mathcal{E}(G,H)| \cdot |\mathcal{E}(H,G)| > 1$. This check can be done in polynomial time when both G and H are cores and partial k-trees.

Corollary 5.6. An algorithm can be constructed that checks, in $O(k^2 n^{k+2})$ steps, whether two cores G and H of treewidth $\leq k$ are isomorphic, where $n = \max\{|V(G)|, |V(H)|\}$.

In general, checking whether two partial k-trees are isomorphic can be done in $O(n^{k+4.5})$ steps [Bod90]. Corollaries 5.5 and 5.6 improve this time for the case where some of the input graphs is a core.

Using the idea of Corollary 5.6, we can count automorphisms of cores with bounded treewidth.

Corollary 5.7. An algorithm can be constructed that outputs, in $O(k^2 n^{k+2})$ steps, the number of automorphisms of a core G of n vertices and treewidth $\leq k$.

A graph G is called *rigid* when the identity is the only homomorphism in $\mathcal{H}(G,G)$ [HN92, Neš99]. We conclude to the following.

Corollary 5.8. An algorithm can be constructed that checks, in $O(k^2 n^{k+2})$, steps whether a n-vertex partial k-tree is rigid or not.

5.3 Problems on counting independent sets

Notice that, in the case where H is the graph in Figure 1, $|\mathcal{H}(G,H)|$ is the number of independent sets of G. Therefore, we have the following.

Corollary 5.9. An algorithm can be constructed that outputs the number of independent sets of a n-vertex partial k-tree in $O(nk2^{k+1})$ steps.

Notice that the linear algorithm of Corollary 5.9 remains polynomial even if the treewidth of G is bounded by $O(\log n)$.

We could remark that, by specializing the structure of H, we can generate an arbitrary number of counting problems that are, in general, #P-complete. As an example we mention the problem of counting the q-particle Widom-Rowlinson configurations in bounded treewidth graphs [DG00].

Suppose that H is a star with a loop attached on its unique internal vertex (a star is a tree of diameter 2). Then, asking whether $|\mathcal{V}(G,H)| \geq 1$ is equivalent to ask whether G has an independent set of size $\geq k$. Finally, Theorem 4.1 yields the following.

Corollary 5.10. An algorithm can be constructed that for any partial k-tree of n vertices and any r, outputs the number of the independent sets of G with at least r vertices in $O(nk(r+1)^{k+1}2^{r+1}\max\{k, 2^{2(r+1)}\})$ steps.

References

[ACP93] S. Arnborg, D.G. Corneil, and A. Proskurowski. Complexity of finding embeddings in a k-tree. *SIAM Journal on Algebraic and Discrete Mathematics*, 8:277–284, 1993.

[AF93] Karl R. Abrahamson and Michael R. Fellows. Finite automata, bounded treewidth and well-quasiordering. In N. Robertson and P. Seymour, editors, *Proceedings of the AMS Summer Workshop on Graph Minors, Graph Structure Theory, Contemporary Mathematics vol. 147*, pages 539–564. American Mathematical Society, 1993.

[ALS91] Stefan Arnborg, Jens Lagergren, and Detlef Seese. Easy problems for tree-decomposable graphs. *Journal of Algorithms*, 12:308–340, 1991.

[And98] Artur Andrzejak. An algorithm for the Tutte polynomials of graphs of bounded treewidth. *Discrete Mathematics*, 190(1-3):39–54, 1998.

[Arn85] Stefan Arnborg. Efficient algorithms for combinatorial problems on graphs with bounded decomposability – A survey. *BIT*, 25:2–23, 1985.

[BDGJ99] Russ Bubley, Martin Dyer, Catherine Greenhill, and Mark Jerrum. On approximately counting colorings of small degree graphs. *SIAM Journal on Computing*, 29(2):387–400, 1999.

[BK96] H. L. Bodlaender and T. Kloks. Efficient and constructive algorithms for the pathwidth and treewidth of graphs. *Journal of Algorithms*, 21:358–402, 1996.

[Bod90] Hans Leo Bodlaender. Polynomial algorithms for graph isomorphism and chromatic index on partial k-trees. *Journal of Algorithms*, 11:631–643, 1990.

[Bod96] H. L. Bodlaender. A linear time algorithm for finding tree-decompositions of small treewidth. *SIAM Journal on Computing*, 25:1305–1317, 1996.

[Bod97] H. Bodlaender. Treewidth: Algoritmic techniques and results. Technical report, UU-CS-1997-31, University of Utrecht, 1997.

[CDF] Colin Cooper, Martin Dyer, and Alan Frieze. On Marcov chains for randomly H-coloring a graph. *Journal of Algorithms* (to appear).

[Cou90] Bruno Courcelle. The monadic second-order logic of graphs. I. Recognizable sets of finite graphs. *Inform. and Comput.*, 85(1):12–75, 1990.

[DFJ99] Martin Dyer, Alan Frieze, and Mark Jerrum. On counting independent sets in sparse graphs. In *40th Annual Symposium on Foundations of Computer Science*, pages 210–217, 1999.

[DG00] M.E. Dyer and C. S. Greenhill. The complexity of counting graph homomorphisms. In *11th ACM/SIAM Symposium on Discrete Algorithms*, pages 246–255, 2000.

[DNS00] Josep Díaz, Jaroslav Nešetřil, and Maria Serna. H-coloring of large degree graphs. Technical Report No. 2000-465, KAM-DIMATIA Series, Charles University, 2000.

[Edw86] K. Edwards. The complexity of coloring problems on dense graphs. *Theoretical Computer Science*, 16:337–343, 1986.

[GHN00] Anna Galluccio, Pavol Hell, and Jaroslav Nešetřil. The complexity of H-colouring of bounded degree graphs. *Discrete Math.*, 222(1-3):101–109, 2000.

[GJ79] M. R. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness.* Freeman, San Francisco, 1979.

[HN90] P. Hell and J. Nešetřil. On the complexity of H-coloring. *Journal of Combinatorial Theory, series B*, 48:92–110, 1990.

[HN92] P. Hell and J. Nešetřil. The core of a graph. *Discrete Mathematics*, 109:117–126, 1992.

[Jer95] M. Jerrum. A very simple algorithm for stimating the number of k-colorings of a low degree graph. *Random Structures and Algorithms*, 7:157–165, 1995.

[Klo94] Z. Kloks. *Treewidth. Computations and Approximations.*, volume 842. Lecture Notes in Computer Science, Springer-Verlag, 1994.

[Lag96] J. Lagergren. Efficient parallel algorithms for graph with bounded tree-width. *Journal of Algorithms*, 20:20–44, 1996.

[Lev73] L. Levin. Universal sequential search problems. *Problems of Information Transmissions*, 9:265–266, 1973.

[MT92] J. Matousek and R. Thomas. On the complexity of finding iso- and other morphisms for partial k-trees. *Discrete Mathematics*, 108:343–364, 1992.

[Neš99] Jaroslav Nešetřil. Aspects of structural combinatorics (graph homomorphisms and their use). *Taiwanese J. Math.*, 3(4):381–423, 1999.

[Neš00] Jaroslav Nešetřil. Personal communication, 2000.

[Ree92] B. Reed. Finding approximate separators and computing tree-width quickly. In *24th ACM Symposium on Theory of Computing*, pages 221–228, 1992.

[RS83] N. Robertson and P. D. Seymour. Graph minors. I. excluding a forest. *Journal of Combinatorial Theory, series B*, 35:39–61, 1983.

[RS84] Neil Robertson and Paul D. Seymour. Graph minors. III. Planar tree-width. *Journal of Combinatorial Theory Series B*, 36:49–64, 1984.

[RS85] Neil Robertson and Paul D. Seymour. Graph minors — a survey. In I. Anderson, editor, *Surveys in Combinatorics*, pages 153–171. Cambridge Univ. Press, 1985.

[RS86] N. Robertson and P. D. Seymour. Graph minors. II. algorithmic aspects of tree-width. *Journal of Algorithms*, 7:309–322, 1986.

[TP97] J. A. Telle and A. Proskurowski. Algorithms for vertex partitioning problems on partial k-trees. *SIAM Journal on Discrete Mathematics*, 10:529–550, 1997.