DIMACS Series in Discrete Mathematics
and Theoretical Computer Science

Recent results on Parameterized H-Coloring

Josep Díaz, Maria Serna, and Dimitrios M. Thilikos

ABSTRACT. We survey recent results on the complexity of several versions of the H-coloring and the list-H-coloring problems that are amenable to parameterization.

1. Introduction

The notion of homomorphism between graphs is a natural algebraic characteristic which has been used to study the structural properties of several combinatorial problems on graphs. Recall that given two graphs $G = (V(G), E(G))$ and $G' = (V(G'), E(G'))$ an *homomorphism* of G into G' is a map $\theta : V(G) \to V(G')$ with the property that $\{v, w\} \in E(G) \Rightarrow \{\theta(v), \theta(w)\} \in E(G')$. For a fixed graph H, possibly with loops but without multiple edges, we say that a graph G has an *H-coloring* if there exists a homomorphism θ from G to H. In Figure 1 we have an example of an H-coloring of a graph. Given any graph G as input, the *H-coloring problem* asks whether there exists an H-coloring of G. Notice that in the particular case when H is the complete graph K_c, the H-coloring problem is the problem of deciding if G is c-colorable. The complexity of the H-coloring problem is well known (see [**42**] for a survey). Hell and Nešetřil [**31**] proved a dichotomy theorem stating that if H is bipartite or it has a loop, then the H-coloring problem can be trivially solved in polynomial time, otherwise the H-coloring problem is NP-complete. In the particular case where G has bounded degree such a dichotomy result seems difficult [**28**].

An interesting extension of the H-coloring problem is *list H-coloring*. Given a graph G and, for every $v \in V(G)$, a set $L(v) \subseteq V(H)$, the *list H-coloring* problem asks whether there is an H-coloring χ of G such that for every $u \in V(G)$ we have $\chi(u) \in L(u)$. Notice that when every list $L(u) = V(H)$, the list H-coloring is the H-coloring problem. Moreover, the list K_c-coloring is the *list-coloring problem* (see for [**18, 38**]). From the previous remarks, we know that if the H-coloring problem is NP-complete, the list H-coloring problem is also NP-complete. However, the dichotomy for the list H-coloring problem is different and harder to achieve. The

2000 *Mathematics Subject Classification.* Primary 05C15; Secondary 68W99.

Key words and phrases. H-coloring, parameterized complexity, FPT, counting, bounded treewidth.

The work of all the authors was supported by the Future Emerging Technologies Program of the EU under contract number IST-99-14186 (ALCOM-FT) and by the spanish CYCIT projects TIC-2000-1970-CE and TIC1999-0754-C03 (MALLBA). The work of the third author was also supported by the Ministry of Education and Culture of Spain (Resolución 31/7/00 - BOE 16/8/00).

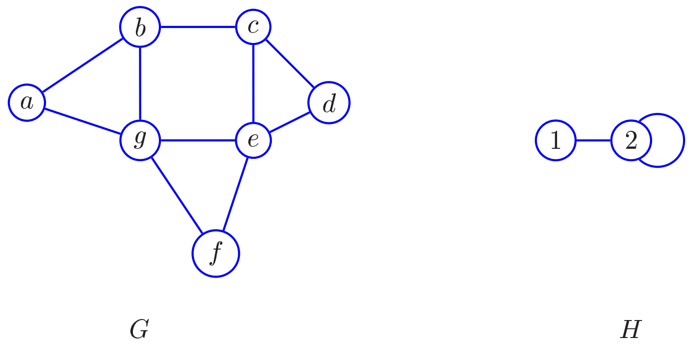


FIGURE 1. An example of an H-coloring of G. The homomorphism is given by: $\theta(a) = \theta(d) = \theta(f) = 1$ and $\theta(b) = \theta(c) = \theta(e) = \theta(g) = 2$

first result showed that when H is a *reflexive* (every vertex is looped) interval graph, the list H-coloring problem can be solved in polynomial time, otherwise the problem is NP-complete [**25**]. The second result states that if H is *irreflexive* (no vertex is looped) and the complement of H is a circular arc graph of clique covering number two, the H-coloring problem can be solved in polynomial time, otherwise the problem is NP-complete [**26**]. The third and last result establishes the dichotomy: when H is a *general* (with looped and unlooped vertices) bi-arc graph, the list H-coloring problem can be solved in polynomial time, otherwise the problem is NP-complete [**27**].

An interesting issue is *counting* the number of solutions to the above mentioned problems. Most counting versions of problems, whose decision versions are NP-complete, are known to be #P-complete. Moreover, the search for efficient approximations to counting problems, known to be #P-complete, has become one of the exciting areas of research in the last decade [**35**]. It should be noted that many counting problems on a graph G can be restated as counting the number of homomorphisms from G to a particular fixed graph H. For instance, the classical problem of counting the number of proper c-colorings of a given graph G is equivalent to counting the number of K_c-colorings of G. As a second example, the problem of counting the number of independent sets corresponds to the problem of counting the number of H-colorings, where H consists of a single edge with a loop in one of its two vertices (see Figure 1).

Given an input graph G, the *#H-coloring problem* is the problem of evaluating the number of H-colorings in G. In the same manner, given the input $(G, \{L(v)\})$, the *list #H-coloring problem* is the problem of evaluating the number of list H-colorings of $(G, \{L(v)\})$.

The following dichotomy result is due to Dyer and Greenhill: the $\#H$-coloring problem is #P-complete if H has a connected component which is not a complete reflexive graph or a complete irreflexive bipartite graph, otherwise it is in P. This result holds even in the case when G has degree bounded by a suitable large constant [**23**].

Regarding the complexity of the problem of counting list H-colorings, it is not difficult to prove that the number of list H-colorings can be computed in polynomial time whenever H is a reflexive complete graph or an irreflexive complete bipartite graph. This observation together with the #P-completenes result for the #H-coloring problem implies the following dichotomy result [**16, 32**]: the list #H-coloring problem is #P-complete if H has a connected component which is not a complete reflexive graph or a complete irreflexive bipartite graph, otherwise the counting problem is in P.

The aim of this paper is to survey the use of ideas from parameterized complexity, to further explore the complexity of the H-coloring and the list H-coloring problems. In defining a parameterization, the issue is not whether a problem is hard, but what makes the problem hard or easy to compute. To study the *structural hardness* of a difficult problem, the approach is to split the input into two parts: a *difficult* part (the non-parameterizided) and an *easy* part (the parameterized), where we impose some restrictions. For several problems, it is known that the parameterization of the input does not break the NP-completeness barrier. A classical example is the *coloring problem* parameterized by the number k of colors that can be used. It is well known that the problem is NP-complete, for $k \geq 3$. On the other hand, problems like the *maximum independent set* or the *minimum vertex cover* of a graph become polynomially solvable when we parameterize them by the size of the independent set or by the size of the vertex cover. Even in the cases where a parameterization of a NP-complete problem leads to a polynomial time algorithm, there are different upper bounds of the running time for the best known algorithm. For instance, the running time could be $O(n^{f(k)})$ or it could be $O(f(k)n^{\alpha})$, where $f(k)$ is a function of the parameter k, and α is a positive constant. A parameterized problem is said to be *Fixed Parameter Tractable* if there exist an $O(f(k)n^{\alpha})$-algorithm that solves the problem. The class FPT is the class of all fixed parameter tractable problems. In a series of papers on parameterized complexity, Downey and Fellows defined a parameterized complexity hierarchy, the W-*hierarchy*, that falls between FPT and W[P], $\mathsf{FPT} \subseteq \mathsf{W}[1] \subseteq \mathsf{W}[2] \subseteq \cdots \subseteq \mathsf{W}[\mathsf{P}]$, along with suitable notions of reducibility and completeness (see [**21**] for a nice exposition of parameterized complexity). It is an open problem whether the inclusion in the hierarchy is strict. Moreover, there is evidence that if a problem is complete for some level of the W-hierarchy, then it is not expected to have an $O(f(k)n^{\alpha})$-algorithm. For example, the parameterized *vertex cover problem* belongs to FPT [**4**], while the the parameterized *independent set problem* is known to be W[1]-complete [**19**].

We survey the parameterized complexity of different variants of the H-coloring and the list H-coloring problems, by considering two different types of parameterizations. The first approach is to set as parameter the *treewidth* of the input graph. Intuitively, the parameter of treewidth is a measure of the global connectivity of a graph (see the formal definitions in Section 2). We survey recent results proving that several extensions and variants of the H-coloring problem, can be solved in time linear to the size of the input graph. We stress that the algorithms are easy to implement and remain polynomial even in the general case where we consider H to be a part of the input.

The second approach is to parameterize the problems by restricting the number of vertices in G that are mapped into a specific subset of vertices of H. We present characterizations of H that allow the classification of the decisional parameterized problems as in FPT, in P, or as NP-complete.

Notice that the counting versions, the list #H-coloring and the #H-coloring, are functional problems and therefore cannot fit into the current framework of the parameterized complexity theory. To study the complexity of the counting versions of the parameterized problems we define the class #PPT (#P Parameter Tractable problems) as the class of parameterized counting problems, which can be solved by an algorithm within time $O(f(k)n^{\alpha})$, where f is some function of the parameter k, and α is a positive integer. We investigate characterizations for which the parameterized counting problems remain #P-complete, as well as conditions for which the parameterization classifies the problem in the classes P or #PPT. Notice that the solution to the list H-coloring and the H-coloring problems, can be considered as a particular case of their counting versions, therefore, when possible, we describe algorithms for solving the counting versions of the problems and we state their decision versions as corollaries to those algorithms.

In the last section, we present some comments and conjectures on the parameterized approach for the counting and decisional versions of the H-coloring problem, and propose some alternative lines of research.

Throughout the paper, we use some standard notation. For the input graph G, we set $n = |V(G)|$ and $m = |E(G)|$. For the fixed graph H, $h = |V(H)|$ and $e = |E(H)|$. For a given graph G and a vertex subset $S \subseteq V(G)$, let $G[S]$ be the subgraph induced by S, and let $G - S$ denote the subgraph $G[V(G) - S]$. As usual K_k denotes a complete graph on k vertices and $K_{k,l}$ a complete bibartite graph with parts containing k and l vertices. As usual for a functiuon θ we use $\theta|_S$ to denote its restrictioin to the set S.

2. Parameterizing by treewidth

Treewidth was first defined by Halin in [**30**] and was reintroduced independently in [**3**] and [**48**]. It plays a key role in many proofs in structural graph theory [**49, 44**] and served as one of the cornerstone concepts for the lengthy proof of the Wagner's conjecture developed by Robertson and Seymour in their Graph Minor Series (see [**47**] for a survey). Formally,

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

(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.

In Figure 2, we present a tree decomposition with optimal width of the graph G given in Figure 1.

As a graph parameter, treewidth has many algorithmic applications. A wide range of combinatorial optimization problems are polynomially solvable when restricted to graphs with bounded treewidth. Unfortunately, computing the treewidth

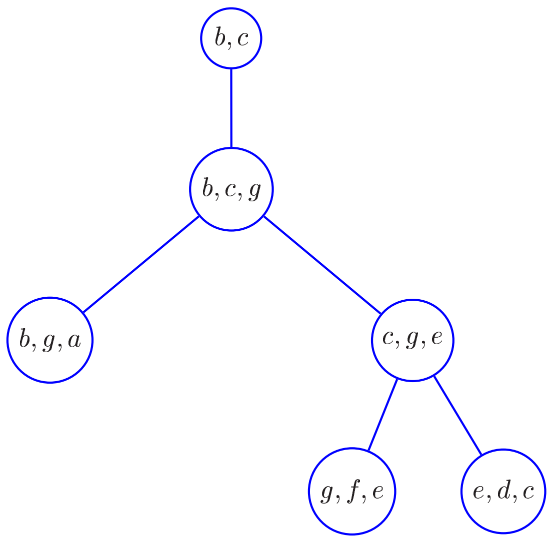


FIGURE 2. A tree decomposition of the graph G in Figure 1

of a graph is a NP-complete problem [**3**]. However, we can fix our attention only to graphs where the treewidth is bounded by a constant k. Such graphs are alternatively called *partial k-trees*. For any constant k, Bodlaender [**6**] presented a linear time algorithm that, given a graph G, checks whether G is a partial k-tree and, if so, outputs a minimum width tree decomposition.

The canonical methodology to get polynomial solutions to difficult problems, consists of a two step procedure: First find a constant width tree-decomposition of the input graph, not necessarily optimal. Then, use *dynamic programming* to get a solution, taking advantage of the bounded treewidth decomposition of the graph (see [**2, 5**]). For the purpose of solving the first canonical step, the algorithm of Bodlaender [**6**] does not seem to be feasible to implement, as it uses as a subroutine the algorithm in [**7**], which involves an enormous hidden constant. However, we can make use of earlier algorithms that output a tree decomposition of G, with width bounded by a linear function of k [**46, 39, 41**]. In particular, the deterministic algorithm by Reed [**46**] runs in $O(n \log n)$ time and, for constant k, it either returns that the treewidth of G is more than k, or it constructs a tree decomposition of width $4k$ or less. This last algorithm allows us to assume that any partial k-tree is always given together with a tree decomposition of constant width.

2.1. The main algorithm. For some problems on partial k-trees, where k is a fixed constant, Courcelle [**13**] associated the existence of a polynomial time algorithm with their expressibility by Monadic Second Order Logic, see also [**12**]. As a consequence of these results, it is possible to construct a polynomial time algorithm solving the H-coloring and the $\#H$-coloring problem for partial k-trees, when k and the size of H are fixed constants. However, the results of Courcelle do not provide implementable algorithms because of the very large hidden constants in their complexity.

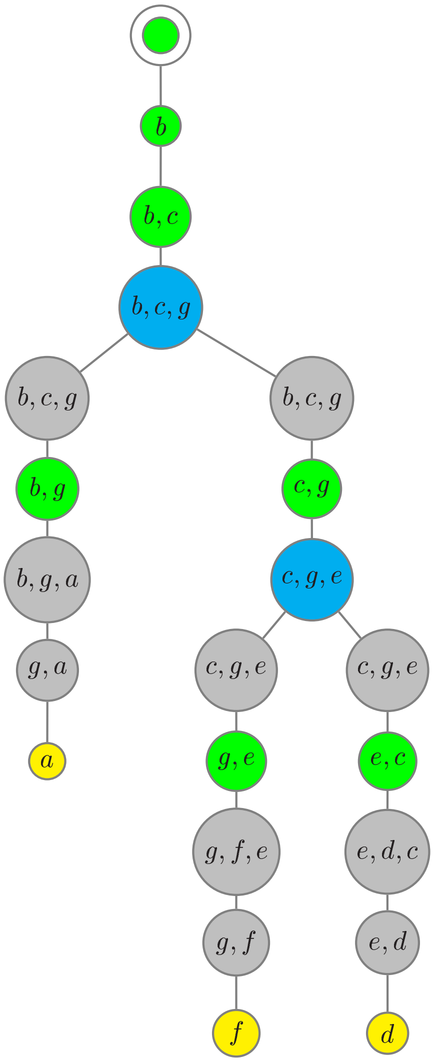


FIGURE 3. A nice tree decomposition of the graph G in Figure 1

The H-coloring problem is one of the many problems, where the dynamic programming technique has yield a polynomial-time solution for partial k-trees [**50**]. In terms of parameterized complexity this means that, for any H, the H-coloring problem parameterized by the treewidth of the input graph is a problem in FPT. In the remaining of the section, we present a polynomial time algorithm for counting H-colorings, in the case that the input graph G has constant treewidth and we survey its consequences.

Although the methodology follows the two cannonical steps, we shall remark that we present an easy to be implemented algorithm for the second step, asumming the aforementioned result on the existence of fast and implementable algorithms for obtaining a bounded width decomposition for a partial k-tree. Moreover, the algorithm remains polynomial even when there are no restricions on the size of H. Once more, this is the first positive result on counting H-colorings, when H is generic and only the input graph is restricted.

Instead of working with the general definition of tree decomposition, we shall consider a more refined version of decomposition, the *nice tree decomposition* defined in [**6, 7, 36**], which simplifies the exploration of the tree structure.

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.

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 and 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 $X_i \subset X_j$.
(4) if a node i is a leaf, then $|X_i| = 1$.

Without lost of generality, we can assume that the root r has $X_r = \emptyset$. In Figure 3 we present a nice tree decomposition for the graph G in Figure 1. It is known that for any constant $k \geq 1$, given a tree decomposition of a graph G of width at most k and $O(n)$ nodes, there exists a linear time algorithm that constructs a nice tree decomposition of G, with $O(n)$ nodes and width at most k [**7**].

Given a nice tree decomposition $D = (X, U, r)$ of G, for any $p \in X$ of D, let U_p denote 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 $G_p = G[V_p]$. Therefore we associate to each node a subgraph, notice that $G_r = G$. A key observation is the fact that a nice tree decomposition $D = (\{X_p \mid p \in V(U)\}, U, r)$ contains four possible node types: *Start node*, if the node is a leaf; *Join node*, if the node has two children $q_i, i = 1, 2$; *Forget node*, if the node p has only one child q and $|X_p| = |X_q| - 1$; and *Introduce node*, if the node p has only one child q and $|X_p| = |X_q| + 1$. Observe that the root r is a *forget* node.

The optimization of the space used in the counting part of the algorithm is attained by the construction of a special topological ordering of $V(U)$, called *stingy*. This ordering has the property that 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 $\log n$. The stingy ordering can be computed in linear time, performing an adequate tree traversal.

Given a nice tree decomposition $D = (X, U, r)$ of a graph G, for each $p \in V(U)$, define the set $F_p = \{\varphi : X_p \to V(H)\}$. 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 $F_r = \{\varnothing\}$, where $\varnothing$ represents the empty function. The table associated to a node $p \in V(U)$ will have an entry for each $\varphi \in F_p$, holding the value $I_p(\varphi) = |\{\theta \mid \theta \text{ is an } H\text{-colorings of } G_p \text{ with } \theta|_{X_p} = \varphi\}|$. As we always have $\theta|_\emptyset = \varnothing$, we get that the number of H-colorings of G is equal to $I_r(\varnothing)$.

THEOREM 2.1 ([**15**]). Given a partial k-tree G, together with a nice tree decomposition $D = (X, U, r)$ with width k, and a stingy ordering of the nodes in D, the algorithm $\mathsf{Count\text{-}H}$ computes the number of H-colorings of G in $O(nh^{k+1} \min\{k, h\})$ steps, using $O(h^{k+1} \log n)$ additional space.

If we use an $O(f(k)n)$ algorithm to solve the first canonical step, we have the following corollary to Theorem 2.1.

```
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, ..., p_s = r) 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 p 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

COROLLARY 2.1. Let k be fixed. The $\#H$-coloring problem, for graphs with treewidth bounded by k, can be solved in $O(n(h^{k+1}\min\{k,h\}) + f(k)))$ steps.

2.2. Other versions of H-coloring. 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 [**29**] as problem GT-52 and it is known to be NP-complete, even for the case where H is a triangle [**40**]. An intermediate problem is the one 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*. The *list exact H-coloring* and the *list vertex exact H-coloring* are defined in the obvious way and we call their counting versions *list exact $\#H$-coloring* and *list vertex exact $\#H$-coloring*. Algorithm count-H can be adapted to solve all these counting variations of the $\#H$-coloring problem.

THEOREM 2.2 ([**15**]). The exact $\#H$-coloring and the vertex exact $\#H$-coloring problems, parameterized by the treewidth of the input graph, are all in #PPT.

Theorem 2.2 yields corollaries analogous to Corollary 2.1 in the obvious way.

2.3. Counting list H-colorings. It is easy to adapt the algorithm count-H to obtain an algorithm for the list $\#H$-coloring problem for graphs of bounded treewidth. The unique changes are the redefinition of the set F_p as $\{\varphi : X_p \rightarrow V(H)$ and $\forall_{x \in X_p} \varphi(x) \in L(x)\}$, and the replacement of the requirement $a \in V(H)$ by $a \in L(v)$, in the treatment of an *start* node, in function count-H. Applying similar changes to the algorithms used to prove Theorem 2.2 we can rewrite them in their "list" versions with the same complexity bounds (for further details see [**15**]).

2.4. The decision version of H-coloring problems. Notice that a graph G is H-colorable if the number of H-colorings of G is non zero. This implies that the algorithms involved in the Theorems 2.1, and 2.2 can solve the corresponding decision version of the problems examined with the same time bounds, as well as their "list" extensions. In other words the decision version of all the variations of the H-coloring introduced in this section, belong in FPT. We mention that for the case of H-coloring of partial k-trees, the result described improves the best known algorithm, which runs in $O(nh^{2(k+1)})$ steps and is due to Telle and Proskurowski in [**50**].

2.5. The directed case. Given two directed graphs $\vec{H}$ and $\vec{G}$, an *$\vec{H}$-coloring of $\vec{G}$* is any function $\theta : V(\vec{G}) \rightarrow V(\vec{H})$ 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 whether, a given directed graph $\vec{G}$ has an $\vec{H}$-coloring. We define the *$\#\vec{H}$-coloring problem* as the problem of, given a directed graph $\vec{G}$, counting all the $\vec{H}$-colorings of $\vec{G}$. In an analougous way, we can extend the definition for the list, exact and vertex exact variations.

It is easy to adapt the algorithms of Theorems 2.1, and 2.2 as well as their "list" extensions for the directed case. The complexity of the corresponding algorithms is the same (for details, see [**15**]).

2.6. Enumeration. Notice that for each of the algorithms described 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 $O(n/\log n)$ to the space reported in Theorems 2.1 and 2.2. 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 $O(n)$ steps.

2.7. The general homomorphism problem. Consider the *homomorphism* problem of deciding, given two input graphs G and H, whether there is an homomorphism from G to H. ¿From Theorem 2.1 we have that the homomorphism problem can be solved in polynomial time if we parameterize it by the treewidth of G. Therefore, the H-coloring problem for partial k-trees remains polynomially solvable even if we consider H to be a part of the input. Similar observations hold for the corresponding list and counting versions.

Notice that if we parameterize the homomorphism problem by considering the treewidth of H, instead of by the treewidth of G, we do not affect the generality of the dichotomy theorem of Hell and Nešetřil [**42**] as, for any k, graphs with treewidth k can be bipartite or non-bipartite. However treewidth related restrictions, for a target graph H, the so called *bounded treewidth duality*, lead to a polynomial solution to the H-coloring problem [**33**], even for the case when H is directed. Similar results are known for the general algebraic homomorphism problem: given two finite relational structures A and B, is there an homomorphisms $H : A \to B$?. This generic formulation include the graph homomorphism problem as well as *constraint satisfaction* problems and *conjunctive-query containment* problems [**37**].

2.8. Coloring and chromatic polynomial. Several researchers have considered the problem of counting the number of proper c-colorings in a graph G. The problem is #P-hard for $c \geq 3$ and maximum degree Δ of G at least 3 [**34**]. 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 $c \geq 2\Delta + 1$. Bubley et al. [**9**] proved that the problem is $\#P$-hard for fixed Δ, but there is a FPRAS for $c = 5$ and $\Delta = 3$. Edwards [**24**] proved that if $c \geq 3$ and the minimum degree where $\delta \geq \alpha n$, the counting problem is #P-complete if $\alpha < \frac{c-2}{c-1}$, but it is in P for $\alpha > \frac{c-2}{c-1}$. Recall that the problem of counting the number of c-colorings of a graph G is the $\#K_c$-coloring problem. Therefore, we conclude that

Corollary 2.2. An algorithm can be constructed to compute the number of c-colorings of a partial k-tree graph, in $O(nc^{k+1} \min\{k, c\})$ steps.

Andrzejak [**1**] has given an $O(n^{2+7\log_2 c})$-time algorithm to compute the Tutte polynomial of a partial k-tree graph on n vertices, where c is twice the number of partitions of a set with $3(k+1)$ elements. Andrzejak's algorithm gave the best procedure to compute the chromatic polymonial for a partial k-tree. As a consequence of the results on counting H-colorings for partial k-trees, the following result is proved in [**15**].

Corollary 2.3. The chromatic polynomial of a partial k-tree can be constructed in $O(kn^{k+3})$ steps.

Notice the the goal of the previous result is not the evaluation of the chromatic polynomial, for which better bouns can be obtained through the evaluation of the Tutte polynomial of a bounded treewidth graph [**43**].

2.9. Other extensions. Recall that in the case where H is an edge with only one looped verex, as in Figure 1, the $\#H$-coloring problem is equivalent to the problem of counting independet sets. Using our previous results, we get

Corollary 2.4. The number of independent sets of a partial k-tree can be obtained in $O(n2^{k+1})$ steps.

It has been observed that, by specializing the structure of H, we can generate an arbitrary number of counting problems that are, in general, #P-complete (see [**23, 22**]). This includes, for example, the problem of counting the q-particle Widom-Rowlinson configurations in bounded treewidth graphs. For further applications of the results described in this section see [**15**].

FIGURE 5. The graphs (H, C, K) for the parameterized independent set and vertex cover as parameterized colorings (the big vertices represent the labeled vertices of H)

Finally, notice that, for the counting and decision version of the problems, Theorems 2.1 and 2.2 provide polynomial time algorithms even in the case where $k = O(\log n)$.

3. Parameterizing H-coloring by image restriction

Let us introduce the second type of *parameterization* for the H-coloring problems. Let $C = \{a_1, \ldots, a_r\}$ be a set of r vertices in $V(H)$ and let $K = (k_1, \ldots, k_r)$ be an r-tuple of non negative integers, where each k_i is associated to a_i. The triple (H, C, K) is called a *partial weighted assignment* on H and a mapping $\chi : V(G) \to V(H)$ is an (H, C, K)*-coloring* of G if χ is an H-coloring of G such that, for all $a_i \in C$, $|\chi^{-1}(a_i)| = k_i$. In this section, we let k denote $\sum_{i=1,\ldots,r} k_i$, for any given partial weighted assignment (H, C, K). We say that a partial weighted assignment (H, C, K) is a *weighted extension* of a graph F if $H - C = F$. We call a partial weighted assignment (H, C, K) *positive* if all the integers in K are positive.

For fixed H, C and K, given an input graph G, the (H, C, K)*-coloring* problem asks whether there exists an (H, C, K)-coloring of G. Notice, that the (H, C, K)-coloring can express well known parameterized problems, like the the *independent set* and the *vertex cover* problems (see Figure 5).

The new parameterization can be extended in a straightforward manner to define the list (H, C, K)-coloring problem. We define the $\#(H, C, K)$*-coloring* as the problem of counting the number of (H, C, K)-colorings of an input graph G, and the *list* $\#(H, C, K)$*-coloring* as the problem of counting the number of list (H, C, K)-colorings of an input $(G, \{L(v)\})$.

3.1. Easy cases.

In this subsection, we survey results from [**16**], giving necessary conditions for the $\#(H, C, K)$-coloring problem and the list $\#(H, C, K)$-coloring, to be solvable in linear time. A sufficient condition, for the list $\#(H, C, K)$-coloring problem to be solved in linear time, is that the set C of parameterized vertices forms a vertex cover. The core of the proof is the Algorithm Count-List-Colorings in Figure 6, which given a graph G, counts the number of list (H, C, K)-coloring, using $O(kn + f(k, h))$ steps, for some function f.

THEOREM 3.1. If (H, C, K) is a partial weighted assignment, where $H - C$ has no edges, there exists an $O(kn + f(k, h))$ time algorithm that solves the list $\#(H, C, K)$-coloring problem.

Algorithm Count-List-Colorings(G, L, H, C, K).
Input: Two graphs G, H, a function $L : V(G) \to 2^{V(H)}$,
and a partial weighted assignment
(H, C, K) on H such that $E(H - C) = \emptyset$.
Output: The number of list (H, C, K)-colorings of G.

1: Let R_1 be the set of vertices in G of degree $> k$.
2: If $|R_1| > k$ or $|E(G)| > kn$ then return 0.
3: Set $G' = G - R_1$.
4: Let R_2 be the non isolated vertices in G' and
let R_3 be the isolated vertices in G'.
5: If $|R_2| > k^2 + k$ then return 0.
6: Setup a partition $\mathcal{R} = (P_1, \ldots, P_m), m \leq q \cdot 2^h$ of R_3 where
q is the number of different neighborhoods of vertices in R_3 and
$\forall_{v,u \in R_3}\ (\exists_{1 \leq i \leq m}\ \{v, u\} \in P_i \Leftrightarrow (N_G(v) = N_G(u) \wedge (L(v) = L(u))))$.
7: Let $Q = \emptyset$.
8: For $i = 1, \ldots, m$,
if $|P_i| \leq k + 1$, then set $F_i = P_i$,
otherwise let F_i be any subset of P_i where $|F_i| = k + 1$.
Set $Q = Q \cup F_i$ and $P_i = P_i - F_i$.
9: Let $\mathcal{H}$ be the set of all the list (H, C, K)-colorings of $G[R_1 \cup R_2 \cup Q]$.
10: Set $\eta = 0$.
11: For any $\chi \in \mathcal{H}$, do
Set $\beta = 1$.
For any nonempty $P_i \in \mathcal{R}$, do
Set $\sigma = \sum_{a \in S} |\chi^{-1}(a) \cap P_i|$, $\tau = \sum_{a \in C} |\chi^{-1}(a) \cap P_i|$
Set $\beta = \beta \cdot \binom{|P_i|}{\tau} \frac{\tau!}{\prod_{a \in C} |\chi^{-1}(a) \cap P_i|!} \binom{|P_i| - \tau}{\sigma} \sigma!\, \sigma^{|P_i| - \tau - \sigma}$.
Set $\eta = \eta + \beta$.
12: return η.
13: End.

FIGURE 6. Algorithm Count-List-Colorings

Recall that any H-coloring can be seen as a list H-coloring in which the list for each vertex is $V(H)$. Therefore, as a corollary to the previous lemma, we get a necessary condition for the $\#(H, C, K)$-coloring problem to be in the class #PPT,

COROLLARY 3.1. If (H, C, K) is a partial weighted assignment, where $H - C$ has no edges, then there exists an $O(kn + f(k, h))$ time algorithm that solves the $\#(H, C, K)$-coloring problem.

It is easy to observe that, when $E(H - C) = \emptyset$, if an (H, C, K)-coloring of G exists then G has a vertex cover of size k. Therefore G must have bounded treewidth, and both Theorem 3.1 and Corollary 3.1 could be obtained as a variation of Algorithm Count-H. However both results improve the complexity bound, as they do not require the computation of a tree decomposition.

3.2. Cases in P. In this section we present the cases of the $\#(H, C, K)$-coloring and list $\#(H, C, K)$-coloring problems which are known to be solved by an algorithm with time bound $O(n^{k+c})$, and therefore can be solved in polynomial time.

The main result relates the problem of counting list (H, C, K)-colorings with the problem of counting list $(H - C)$-colorings. This result is the key for all the positive results in this subsection and it appears in [**16**].

THEOREM 3.2. The list $\#(H, C, K)$-coloring problem can be solved in n^{k+c} steps, whenever the list $\#(H - C)$-coloring problem can be solved in $O(n^c)$ steps.

Recalling, once more, that the number of H-colorings is the same as the number of list H-coloring for an adequate list selection, we get

COROLLARY 3.2. The $\#(H, C, K)$-coloring problem can be solved in n^{k+c} steps, if the list $\#(H - C)$-coloring problem can be solved in $O(n^c)$ steps.

As mentioned in the introduction, the dichotomy for counting list H-colorings is the same as for counting H-colorings. Therefore, if we can solve $\#H$-coloring problem in polynomial time, we can also solve the list $\#H$-coloring problem in polynomial time.

COROLLARY 3.3. The list $\#(H, C, K)$-coloring and the $\#(H, C, K)$-coloring problems can be solved in n^{k+c} steps, if the $\#(H - C)$-coloring problem can be solved in polynomial time. Where c is a constant independent of k.

3.3. Hardness results. Let us consider negative results giving conditions under which the list $\#(H, C, K)$-coloring remains $\#\mathsf{P}$-complete. The following result has not been published previously and we provide here a complete proof.

THEOREM 3.3. For any parameter assignment (H, C, K), the list $\#(H, C, K)$-coloring problem is $\#\mathsf{P}$-complete whenever the $\#(H - C)$-coloring problem is $\#\mathsf{P}$-complete.

PROOF. Let $H' = H - C$. We reduce the $\#H'$-coloring problem to the list $\#(H, C, K)$-coloring problem. Let G be an instance of the H'-coloring problem. We construct an instance G' of the (H, C, K)-coloring problem as follows: Take G' as the disjoint union of G and a graph F, that consists of k isolated vertices. We assign to every $v \in V(G)$, the list $L(v) = V(H')$ and to every $f \in V(F)$, we assign the list $L(f) = C$.

Notice that, from the way we have defined the lists, the additional vertices can be only mapped to the set of parameterized vertices, but any arrangement of images is possible. The size of F guarantees that every parameterized vertex receives the prescribed number of images. Furthermore, combining an arrangement of the vertices of F with an H'-coloring of G we obtain a valid list H-coloring of $(G', \{L(v)\})$.

Therefore, for any H'-colorings of G there are $k!$ list H-colorings of $(G', \{L(v)\})$ and the statement of the theorem follows. $\square$

Putting together Theorems 3.2, 3.3 and the results in [**16, 32, 23**] we get the following dichotomy.

THEOREM 3.4. The list $\#(H, C, K)$-coloring is $\#\mathsf{P}$-complete if $H - C$ has a connected component which is not a complete reflexive graph or a complete irreflexive bipartite graph. Otherwise the counting problem is in P.

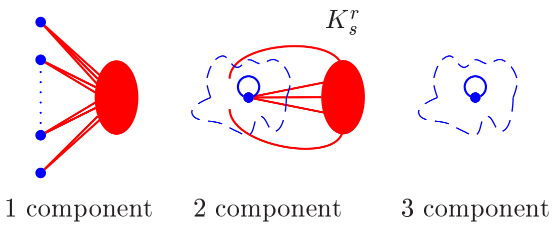


FIGURE 7. Components for a compact partial weighted assignment

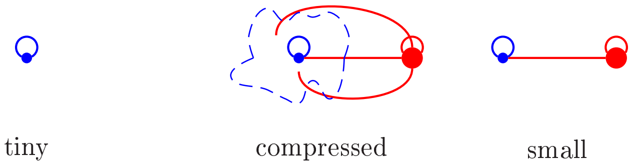


FIGURE 8. Special cases of components for a compact partial weighted assignment

3.4. The decision version. All the positive results for the counting versions can be translated to their decision versions, however, polynomial time and hardness results can be reinforced. In [**17**] another necesary condition for the (H, C, K)-coloring problem to be in the class FPT was presented. First we need a definition.

We say that a partial weighted assignment (H, C, K) is *compact* when each connected component H_i of H satisfies one of the following exclusive conditions:

(1) $E(H_i - C) = \emptyset$,
(2) $H_i[C]$ is a non-empty reflexive clique with all its vertices adjacent with one looped vertex of $H_i - C$, or
(3) $V(H_i) \cap C = \emptyset$ and H_i contains at least one looped vertex.

Figure 7 gives a representation of the types of components allowed in a compact weighted assignment. Let (H, C, K) be a compact partial weighted assignment. Notice that if $\chi : V(G) \to V(H)$ is a (H, C, K)-coloring of G, the total size of the connected components of G that are mapped to the vertices of some component H_i of type (2), is at least k_i, where a_i is H_i contribution to the total weight.

The next theorem was proved in [**17**]. The crunch of the proof is an $O(n(k + h)) + \gamma(G)f(k, h))$ time algorithm, where $\gamma(G)$ is the number of the connected components of the input graph G. The basic ideas of the algorithm are the following: first, reduce the problem to a compact partial weighted assignment for the particular components depicted in Figure 8. When G has "many" connected components an adaptation of the algorithm Count-List-Colorings is used to check whether a "small" number of them are mapped to the 1-components of H, if this is possible plenty of components are left unmapped, which will allow an extension of the mapping to the remainning of G. When G has few connected components, the number of components of G that can be mapped to the components of H does not depend on the size of G, and a exhaustive search algorithm checks the existence of an (H, C, K)-coloring.

THEOREM 3.5. If (H, C, K) is compact, then the (H, C, K)-coloring problem, parameterized by K, belongs to the class FPT.

It is not known any equivalent result for the list (H, C, K)-coloring problem. For the hardness results, an extension of the argument used in the proof of Theorem 3.3 yields the following result on the complexity of the list (H, C, K)-coloring problem.

COROLLARY 3.4. The list (H, C, K)-coloring problem is NP-complete, whenever the list $(H - C)$-coloring problem is NP-complete.

On the other hand, a much more involved argument is presented in [**17**], which provides a similar hardness result for the (H, C, K)-coloring problem.

THEOREM 3.6. The (H, C, K)-coloring problem is NP-complete if $(H - C)$ is not bipartite and it does not contain a loop.

On the positive side, Theorem 3.2 can be extended to get both problems in the class P.

COROLLARY 3.5. The (H, C, K)-coloring and the list (H, C, K)-coloring problems can be solved in n^{k+c} steps, whenever the list $(H - C)$-coloring can be solved in $O(n^c)$ steps.

The previous result include the particular cases where $H - C$ is a reflexive interval graph [**25**] and where $H - C$ is an irreflexive graph whose complement is a circular arc graph of clique covering number two [**26**]. Furthermore, Theorem 3.6 and Corollary 3.5 provide a dichotomy theorem for the list (H, C, K)-coloring problem.

THEOREM 3.7. The list (H, C, K)-coloring problem is NP-complete if $(H - C)$ is not a bi-arc graph, otherwise it is in P.

For the (H, C, K)-coloring problem, there is a gap, because the positive results relate to the dichotomy for the list H-coloring problem, while the negative results relate to the dichotomy for the H-coloring problem. For some graphs in the gap, there are negative and positive results depending on the weigthed extension selected.

The next result proved in [**17**] shows that if F is a graph, for which the list F-coloring problem is NP-complete, but the F-coloring problem is in P, then F has weighted extensions for which the problem falls in P.

THEOREM 3.8. Let F_1 be either a bipartite graph or a graph with at least one loop, and let F_2 be any graph. Then, the $(F_1 \oplus F_2, V(F_2), K)$-coloring can be solved in polynomial time for any partial weighted assignment K.

In the previous theorem, $F_1 \oplus F_2$ denotes the graph obtained from F_1 and F_2 adding all the edges between vertices in different graphs. It is also worth mentioning that for any weighted extension of a graph F for which the F-coloring problem is in P, the problem is also polynomially solvable in the trivial case $K = (0, \dots, 0)$.

The condition of Theorem 3.6 is not necessary. The next result presents weighted assignments (H, C, K) for which the $(H - C)$-coloring problem is in P, but the (H, C, K)-coloring problem is NP-compete.

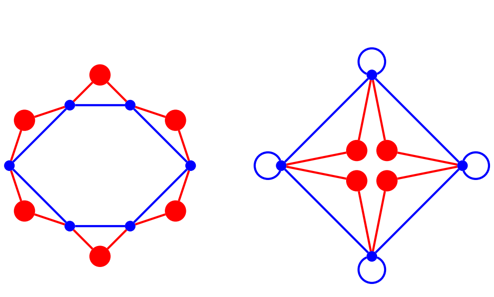


Figure 9. Some hard partial weighted assignments(the big vertices represent the labeled vertices of H)

Theorem 3.9. The (H, C, K)-coloring problem is NP-complete for the partial weighted assignments depicted in Figure 9, provided that (H, C, K) is positive.

Finally, notice that the subgraph induced by the non parameterized vertices in each of the 3 examples given in Figure 9 have, by Theorem 3.8, another weighted extension, so that the corresponding parameterized coloring problem belongs to the class P.

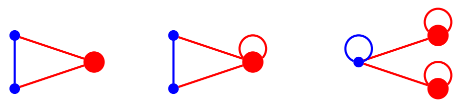
4. Remarks and open problems

The results for all the versions of the H-coloring problem are sumarized in Tables 1 and 2. Notice that, all the hardness results presented in this paper, can be extended to the case in which the input graph has degree bounded by some constant. Several problems remain unsolved, we present some of them together with some conjectures. We propose other interesting ways to parameterize the H-coloring problem, and finish by citing some open problems related to lines of attack #P-hard problems.

4.1. Dichotomy conjectures. The hardness result for the $\#(H, C, K)$-coloring problem seems more difficult to obtain. The results in [**16**] provide a partial answer, by showing #P-completeness when H has the property that no homomorphism from H to H sends vertices in C to vertices in $V(H) - C$.

Conjecture 4.1. The $\#(H, C, K)$-coloring is #P-complete if $H - C$ has a connected component which is not a complete reflexive graph or a complete irreflexive bipartite graph. Otherwise the counting problem is in P.

The results in Theorems 3.6 and 3.8 are sharp in the following sense: If F is a graph where the F-coloring is NP-complete then, for any weighted extension (H, C, K) of F, the (H, C, K)-coloring is also NP-complete. On the other hand, if the F-coloring problem is in P then *there exist* a weighted extension (H, C, K) of F so that the (H, C, K)-coloring problem is in P. Moreover, the results in Theorem 3.9 and Corollary 3.5 are also sharp, in the sense that if F is a graph where the list F-coloring problem is in P then, for any weighted extension (H, C, K) of F, the (H, C, K)-coloring problem is also in P. We provided some examples where the list F-coloring problem is NP-complete and F has a weighted extension (H, C, K) such that the (H, C, K)-coloring problem is also NP-complete.



(H_1, C_1, K_1) (H_2, C_2, K_2) (H_3, C_3, K_3)

FIGURE 10. Three weighted extensions conjectured to be W[1]-hard when K is parameterized (the big vertices represent the labeled vertices of H)

The above observations indicate that different weighted extensions of some graphs produce parameterized coloring problems with different complexities. It seems to be a hard problem to achieve a dichotomy discriminating those parameterized assignments (H, C, K) of a given graph F for which the (H, C, K)-coloring problem is NP-complete or in P. However we conjecture the following.

CONJECTURE 4.2. For any graph F such that the list F-coloring problem is NP-complete, there is a weighted extension (H, C, K) of F such that the (H, C, K)-coloring problem is also NP-complete.

The results on NP-completeness indicate, that the frontier depends not only on the structure of the graph $H - C$, but also on the structure imposed by the weighted vertices. We observe that in all the complexity results described in this survey, positive or negative, the dichotomy does not depend on the choice of the numbers in K (when K is positive).

We now fix our attention on conditions classifying the (H, C, K)-coloring problem in FPT. We conjecture that the condition of Theorem 3.5 is also necessary.

CONJECTURE 4.3. For any partial weighted assignment (H, C, K), the (H, C, K)-coloring problem is in FPT if (H, C, K) is compact, otherwise the problem is W [1]-hard.

In support of this conjecture we recall that the parameterized *independent set* problem, known to be W [1]-complete [**20**], satisfies the conjecture.

Conjecture 4.3 is sufficiently general to express several open problems in parameterized complexity such as

(1) (Parameter: k) Does G contain an independent set S, where $|S| = k$, and such that $G[V(G) - S]$ is bipartite? (This problem can be seen as a parameterization of 3-coloring where some color should be used exactly k times.)
(2) (Parameter: k) Is there a set $S \subseteq V(G)$, with $|S| = k$, such that $G[V(G) - S]$ is bipartite?
(3) (Parameters: k, l) Does G contain $K_{k,l}$ as subgraph?

These three problems correspond to the parameterized colorings given in Figure 10. Observe that the (H_3, C_2, K_3)-problem is equivalent to ask if the complement of G contains K_{k_1,k_2} as a subgraph.

FIGURE 11. The graphs (H, C, K) for the parameterized cut as parameterized colorings

4.2. Other parameterizations of H-coloring. Other interesting parameterizations of the H-coloring problem are obtained by modifying the equality condition on K. We can consider the $(H, C, \geq K)$-coloring and the $(H, C, \leq K)$-coloring problems, defined like the (H, C, K)-coloring problem with the difference that, for any $a_i \in C$, the number of preimages of a_i is at most k_i or at least k_i, respectively. If we consider the first partial weighted assignment in Figure 5, the $(H, C, \geq K)$-coloring problem is equivalent to the (H, C, K)-coloring problem and the $(H, C, \leq K)$-coloring problem is trivial. For the second partial weighted assignment in Figure 5, the $(H, C, \leq K)$-coloring problem is equivalent to the (H, C, K)-coloring problem and the $(H, C, \geq K)$-coloring problem is trivial.

In the parameterization described in Section 3, we fixed the number of preimages of a set of vertices in an H-coloring. As an H-coloring of G maps also the edges of G to edges of H, one could enhance the definition of a (H, C, K)-coloring by including also restrictions on the number of preimages of a subset of edges. This can be done by allowing C to be a subset of $V(H) \cup E(H)$. As usual, if θ is an H-coloring, we define the preimages of an edge $e = \{a, b\}$ as $\theta^{-1}(e) = \{\{v, u\} \in E(G) \mid \theta(v) = a \text{ and } \theta(u) = b\}$. Both the $(H, C, \leq K)$-coloring and the $(H, C, \geq K)$-coloring can be generalized to allow edge subsets in the obvious way.

An interesting case is the parameterization (H, C, K) (see Figure 11), where H is the graph with two looped vertices connected by an edge e, $C = \{e\}$ and $K = (k)$. This particular problem is equivalent to the problem of deciding whether G has an edge cut of size equal to k. Accordingly, the "$\leq$"-version and the "$\geq$"-version of the same problem asks for an edge cut of size $\geq k$ (the decision version of the *min-cut* problem, which is in P) and an edge cut of size $\leq k$ (the decision version of the *max-cut* problem, which is NP-complete) respectively. Moreover, the *max-cut* problem is known to be in the class FPT. Recall, that all the problems in the class MAXNP are also in FPT [**10**], and furthermore it is known that the *max-cut* problem is MAXSNP-complete [**45**].

On the other hand, if $\mathbf{1}$ denotes the vector with all components set to one, the $(H, E(H), \geq \mathbf{1})$-coloring is the exact H-coloring problem, which is NP-complete [**40**].

Another interesting parameterization for the list H-coloring, is to set as parameter the maximum allowed size for each list.

4.3. Other directions. Given the hardness results of this survey, a natural question is to approximate the counting versions of the H-coloring that are known to be in #P. As the problem is self reducible, a candidate technique would be

-coloring	FPT	P	NP-complete
list H	bounded trewidth [**15**]	dichotomy [**27**]	
H	bounded trewidth [**15**]	dichotomy [**31**]	
list (H, C, K)	$E(V-C)=\emptyset$ [**17**]	dichotomy [**17**]	
(H, C, K)	(H, C, K) compact [**17**]	list $(H-C)$-coloring in P [**17**]	$(H-C)$-coloring NP-hard [**17**]

TABLE 1. Complexity of H-coloring problems

-coloring	#PPT	P	#P-complete
list $\#H$	bounded trewidth [**15**]	dichotomy [**16, 32**]	
$\#H$	bounded trewidth [**15**]	dichotomy [**23**]	
list $\#(H, C, K)$	$E(V-C)=\emptyset$ [**16**]	dichotomy [**16**]	
$\#(H, C, K)$	$E(V-C)=\emptyset$ [**16**]	list $\#(H-C)$ -coloring in P [**16**]	(H, C, K) irreducible [**16**]

TABLE 2. Complexity of the counting H-coloring problems

the almost uniform generation, via the Markov chain method, to generate an almost uniform sampling. In [**11, 22**], there are given negative results for uniform sampling in the case of some particular selection of H and one positive result for weighted sampling when H is a tree. It remains a challenging problem, to get more general results to approximate the $\#H$-coloring and the list $\#H$-coloring and their parameterizations.

Other model of research consider that H has weights on all the vertices. This weighting of the vertices of H is interpreted as a product measure on the set of H-colorings [**8**]. It will be of interest to know if it is posible to have a polynomial time algorithm for sampling H-colorings accordingly to the probability distribution induced by the product measure.

Positive and negative complexity results concerning the special case of the coloring problem in which the degree of the input graph is large were studied in [**24**]. Edwards results on colorings were partially extended to the H-coloring problem in [**14**]. It will be of interest to extend the results on counting colorings in [**24**] to the H-coloring problem.

Acknowledgement. We thank an anonymous referee for improving the quality of the reading.

References

[1] A. Andrzejak. An algorithm for the Tutte polynomials of graphs of bounded treewidth. *Discrete Mathematics*, 190:39–54, 1998.

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

[3] S. Arnborg, D. G. Corneil, and A. Proskurowski. Complexity of finding embeddings in a k-tree. *SIAM Journal on Algebraic and Discrete Methods*, 8:277–284, 1987.

[4] R. Balasubramanian, M. Fellows, and V. Raman. An improved fixed-parameter algorithm for vertex cover. *Information Proccessing Letters*, 65:163–168, 1998.

[5] H. Bodlaender. Treewidth: algorithmic techniques and results. In *Mathematical Foundations of Computer Science*, Lectures Notes in Computer Science, pages 19–36. Springer-Verlag, 1997.

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

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

[8] G. Brightwell and P. Winkler. Graph homomorphism and phase transitions. *Journal of Combinatorial Theory (series B)*, 77:415–435, 1999.

[9] R. Bubley, M. Dyer, C. Greenhill, and M. Jerrum. On approximately counting colorings of small degree graphs. *SIAM Journal on Computing*, 29(2):387–400, 1999.

[10] L. Cai and J. Chen. On fixed-parameter tractability and approximability of NP optimization problems. *Journal of Computer and System Sciences*, 54(3):465–474, 1997.

[11] C. Cooper, M. Dyer, and A. Frieze. On Markov chains for randomly H-coloring a graph. *Journal of Algorithms*, 39:117–134, 2001.

[12] B. Courcelle, J. Makowski, and U.Rotics. On the fixed parameter complexity of graph enumeration problems definable in monadic second order logic. *Discrete Applied Mathematics*, 108(1-2):23–52, 2001.

[13] B. Courcelle and M. Mosbah. Monadic second-order evaluations on tree-decomposable graphs. *Theoretical Computer Science*, 109:49–82, 1993.

[14] J. Díaz, J. Nešetřil, M. Serna, and D. M. Thilikos. H-coloring of large degree graphs. In *Advances in Information and Communication Technology (EURASIA-ICT-2002)*, volume 2510 of *Lecture Notes in Computer Science*, pages 850–857. Springer-Verlag, 2002.

[15] J. Díaz, M. Serna, and D. M. Thilikos. Counting H-colorings of partial k-trees. In *Computing and Combinatorics, COCOON 2001*, volume 2108 of *Lectures Notes in Computer Science*, pages 298–307. Springer-Verlag, 2001. To appear in *Theoretical Computer Science*.

[16] J. Díaz, M. Serna, and D. M. Thilikos. Fixed parameter algorithms for counting and deciding bounded restrictive list H-colorings. Technical Report, Departament de Llenguatges i Sistemes Informàtics, Universitat Politècnica de Catalunya, Barcelona, 2004.

[17] J. Díaz, M. Serna, and D. M. Thilikos. (H, C, K)-colorings: Fast, easy and hard cases. In *Mathematical Foundations of Computer Science 2001, MFCS-2001*, volume 2136 of *Lecture Notes in Computer Science*, pages 304–315. Springer-Verlag, 2001.

[18] Q. Donner. On the number of list-colorings. *Journal of Graph Theory*, 16(3):239–245, 1992.

[19] R. G. Downey and M. R. Fellows. Fixed-parameter tractability and completeness. II. On completeness for $W[1]$. *Theoretical Computer Science*, 141(1-2):109–131, 1995.

[20] R. G. Downey and M. R. Fellows. *Parameterized complexity*. Springer-Verlag, New York, 1999.

[21] R. G. Downey and M. R. Fellows. Parameterized complexity after (almost) ten years: review and open questions. In *Combinatorics, computation & logic '99 (Auckland)*, pages 1–33. Springer, Singapore, 1999.

[22] M. Dyer, A. Frieze, and M. Jerrum. On counting independent sets in sparse graphs. In *Proceedings of the 40th Annual IEEE Symposium on Foundations of Computer Science*, pages 210–217, 1999.

[23] M. Dyer and C. Greenhill. The complexity of counting graph homomorphisms. *Random Structures Algorithms*, 17:260–289, 2000.

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

[25] T. Feder and P. Hell. List homomorphisms to reflexive graphs. *Journal of Combinatorial Theory (series B)*, 72(2):236–250, 1998.

[26] T. Feder, P. Hell, and J. Huang. List homomorphisms and circular arc graphs. *Combinatorica*, 19:487–505, 1999.

[27] T. Feder, P. Hell, and J. Huang. Bi-arc graphs and the complexity of list homomorphism. manuscript, 2001.

[28] A. Galluccio, P. Hell, and J. Nešetřil. The complexity of H-colouring of bounded degree graphs. *Discrete Mathematics*, 222(1-3):101–109, 2000.

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

[30] R. Halin. S-functions for graphs. *Journal of Geometry*, 8:171–186, 1976.

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

[32] P. Hell and J. Nešetřil. Counting list homomorphisms and graphs with bounded degrees. this procedings, 2001.

[33] P. Hell, J. Nešetřil, and X. Zhu. Duality and polynomial testing of tree homomorphisms. *Transactions of the American Mathematical Society*, 348(4):1281–1297, 1996.

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

[35] M. Jerrum and A. Sinclair. The Markov chain Monte Carlo method: An approach to approximate counting and integration. In D. S. Hochbaum, editor, *Approximation Algorithms for NP-hard problems*, pages 482–520. PWS, Boston, 1995.

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

[37] Ph. G. Kolaitis and M. Y. Vardi. Conjunctive-Query containment and Constraint Satisfaction. Journal of Computer and System Sciences, 61(2):302–332 2000.

[38] J. Kratochvíl and Z. Tuza. Algorithmic complexity of list colorings. *Discrete Applied Mathematics*, 50(3):297–302, 1994.

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

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

[41] J. Matoušek and R. Thomas. Algorithms finding tree-decompositions of graphs. *Journal of Algorithms*, 12:1–22, 1991.

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

[43] S. D. Noble. Evaluating the Tutte polynomial for graphs of bounded treewidth. *Combinatorics, Probability and Computing*, 7:307–321, 1998.

[44] B. Oporowski, J. Oxley, and R. Thomas. Typical subgraphs of 3- and 4-connected graphs. *Journal of Combinatorial Theory (series B)*, 57(2):239–257, 1993.

[45] C. Papadimitriou and M. Yannakakis. Optimization, approximation, and complexity classes. *Journal of Computer and System Sciences*, 43:425–440, 1991.

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

[47] N. Robertson and P. D. Seymour. Graph minors — a survey. In I. Anderson, editor, *Surveys in Combinatorics*, pages 153–171. Cambridge University Press, 1985.

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

[49] N. Robertson and P. D. Seymour. Graph minors. V. Excluding a planar graph. *Journal of Combinatorial Theory (series B)*, 41:92–114, 1986.

[50] J. A. Telle and A. Proskurowski. Algorithms for vertex partitioning problems on partial k-trees. *SIAM Journal on Discrete Mathematics*, 10(4):529–550, 1997.

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

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

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