Fixed Parameter Algorithms for Counting and Deciding Bounded Restrictive list H-Colorings*

Josep Díaz, Maria Serna, and Dimitrios M. Thilikos

Dept. de Llenguatges i Sistemes Informàtics, Universitat Politècnica de Catalunya, Campus Nord–C6. E-08034, Barcelona, Spain. {diaz,mjserna,sedthilk}@lsi.upc.es

Abstract. We are interested in studying the fixed parameter tractability of the counting and the decision version of a parameterization of the H-coloring problem. These problems are defined by fixing a partial weight assignment (H, C, K). We define two classes of partial weight assignment the *simple* and the *plain*. We show an FPT algorithm for counting list (H, C, K)-colorings for simple partial weight assignment. For the more general class of plain partial weight assignment we give an FPT algorithm for the (H, C, K)-coloring decision problem. We introduce a new algorithmic technique, *compactor enumeration*, that allow us to design the FPT algorithms for counting problems.

1 Introduction

Given two graphs G and H, a *homomorphism* from G to H is any function mapping the vertices in G to vertices in H, in such a way that the image of an edge is also an edge. In the case that H is fixed, such a homomorphism is called H-coloring of G. For a given graph H, the *H-coloring problem* asks whether there exists an H-coloring of the input graph G. The more general version in which a list of allowed colors (vertices of H) is given for each vertex is known as the list H-coloring. The case in which the number of pre-images of the vertices in H are part of the input, is known as the *restrictive H-coloring* or the *restrictive list H-coloring*. Besides the decisional versions of these problems, we also consider the problem of counting the number of solutions. In that way one can define the $\#H$-coloring, the list $\#H$-coloring, the restrictive $\#H$-coloring, and the restrictive list $\#H$-coloring. The known results concerning the complexity of all the problems defined so far are summarized in Table 1.

The input to a parameterized problem, is a pair (I, K) where I is the main part and K is the parameterized part. Typically, K is some fixed integer (or a sequence of fixed integers) and is called *parameter*. The parameterized version of the *restrictive H-coloring* problems in which the number of allowed pre-images of some vertices in H are fixed independently of the input is known as the *(H, C, K)-coloring* and was introduced in [2]. Where C is the set of vertices with restrictions and K is a weight assignment on C defining the number of preimages. (H, C, K) is the *partial weight assignment* corresponding to the parameterized part of the problem. We studied the complexity of the (H, C, K)-coloring in [2, 5] and the results are summarized in Table 1. See [4] and [1] for surveys on different problems based on H-colorings.

In this paper we are interested in finding the largest class of (H, C, K) for which the (H, C, K)-coloring problems can be solved in FPT. Recall that an FPT algorithms solves a

* The work of all the authors was partially supported by the EU within the 6th Framework Programme under contract 001907 (DELIS). The work of the first author was partially supported by the Distinció per a la recerca of the Generalitat de Catalunya. The work of the second and third authors was partially supported by the Spanish CICYT project TIC-2002-04498-C05-03.

Problem	P	NP-complete/#P-complete
list H-coloring	dichotomy(1) [9]	
H-coloring	dichotomy(2) [12]	
list $\#H$-coloring	dichotomy(3) [3, 13]	
$\#H$-coloring	dichotomy(3) [7]	
restrictive list (H, C, K)-coloring	dichotomy(3) [3]	
restrictive (H, C, K)-coloring	dichotomy(3) [3]	
restrictive list $\#(H, C, K)$-coloring	dichotomy(3) [3]	
restrictive $\#(H, C, K)$-coloring	dichotomy(3) [3]	
list (H, C, K)-coloring (K fixed)	dichotomy(3) [2, 5]	
(H, C, K)-coloring (K fixed)	list $(H - C)$-coloring in P (K fixed)	$(H - C)$-coloring NP-hard [2, 5]
list $\#(H, C, K)$-coloring (K fixed)	dichotomy(3) [2, 5]	
$\#(H, C, K)$-coloring (K fixed)	list $\#(H - C)$-coloring in P	(H, C, K) irreducible [2, 5]

(1) H is a bi-arc graphs
(2) H is bipartite or contains a loop
(3) All the connected components of H are either complete reflexive graphs or complete irreflexive bipartite graphs

Table 1. Complexity of the parameterized H-coloring problems

parameterized problem in time $O(f(k)n^{O(1)})$ where k is the parameter and n is the input size. We isolate a subclass of the problem, the *simple partial weight assignments* (see definition later), for which we were able to design linear time fixed parameter algorithms for both counting and decision version of the list (H, C, K)-problem. In the case of the (H, C, K)-coloring we can isolate a larger class, the *plain partial weight assignments*, for which the (H, C, K)-problem problem is in FPT.

One of the fundamental tools for designing FPT algorithms for decision problems is the so called *Reduction to problem kernel.* The method consists in the polynomial time (in n and k) transformation of any problem input (I, K) to a reduced instance of the same problem with inputs (I', K') (the kernel) such that the sizes of $|I'|$ and K' depend *only* on some function of k and $(I, K) \in \mathcal{L} \Leftrightarrow (I', K') \in \mathcal{L}$. (See [6] for discussions on fixed parameter tractability and the ways of constructing kernels.)

In recent years, there has been a big effort focused on developing a theory for the intractability of parameterized counting problems (see for example [10, 14]). However, so far, no progress has been noticed on the development of techniques parallel to the one of the reduction to problem kernel for counting problems. We introduce a new algorithmic tool, instead of isolating a kernel to which the counting problem can be reduced to, we define a general type of structure, called *compactor*, which contains a *certificate* for each set in a special partition of the solution space. Such a structure (as well as the sizes of its elements) can be constructed in polynomial time and its size depends only on the parameterized part of the problem. Finally, *enumerating* all certifying structures and computing, for each of them, the number of solutions they certify, we obtain a FPT algorithm for counting the solutions of the initial problem.

We stress that our method is totally different than the one of reduction to a problem kernel as, in our approach, we essentially reduce the counting problem to an enumeration problem. Moreover, the reduced problem does not consists of an instance of the initial problem as in the kernel reduction case (in particular, the compactor is a structure consisting of a suitable collection of functions). We stress that our new method is also applicable to decision parameterized problems where the technique of reduction to a kernel seems to fail (see Section 5).

The complexity of all our FPT algorithms for counting are *linear* in n, and it is of the form $O(f_1(k)n+f_2(k)g+f_3(k)\log n+f_4(k))$, where g is the number of connected components of G and functions f_i increase in complexity as i grows. Although the counting algorithms can be used to solve the decision version of the problem, we design specific algorithms for the decision version where the contribution of the parameter is similar, with the exception that the term depending on $\log n$ dissapears. To achieve this, apart from using compactors, we also combine different design tools like reduction to a kernel, *bounded search trees*, and *parameterized reduction* to achieve fast FPT algorithms for particular subcases of our problems (see for example [15] for a survey on tools for parameterizing algorithms). Furthermore we extend the concept of H-colorings to node weighted graphs.

Finally, let us mention that several parameterized problems can be modeled as particular cases of the (H,C,K)-coloring, in which (H,C,K) is simple. Therefore they can be solved by an FPT algorithm both the counting and th edecision version. In particular the following parameterization of PARTITION: "Given integers $x_1,\dots,x_n$ and a parameter k, is there a set $I\subseteq[n]$ such that $\sum_{i\in I}x_i\geq k$ and $\sum_{i\notin I}x_i\geq k$?" and the Vertex Cover problem (recall that an FPT algorithm for counting the number of vertex covers of size k was given in [10]).

2 Definitions

We use standard notation for graphs. Following the terminology in [8, 9] we say that a graph is *reflexive* if all its vertices have a loop, and that a graph is *irreflexive* when none of its vertices is looped. We denote by $g=g(G)$ the number of connected components of G. For $U\subseteq V(G)$, we use $G-U$ as a notation for the graph $G[V(G)-U]$. Given two graphs G and G', define $G\cup G'=(V(G)\cup V(G'),E(G)\cup E(G'))$, and $G\oplus G'=(V(G)\cup V(G'),E(G)\cup E(G')\cup\{\{u,v\}\mid u\in V(G),v\in V(G')\}$. Given a bipartite connected graph G, the vertex set $V(G)$ is splitted among two sets $X(G)$ and $Y(G)$ so that every edge has an end point in the X-part and another in the Y-part. Observe that the X and Y parts of a connected bipartite graph are defined without ambiguity. $K_{r,s}$ denotes the complete irreflexive bipartite graph on two sets with r and s vertices each, notice that $K_{r,0}$ denotes a complete bipartite graph, and K_n^r denotes the reflexive clique with n vertices.

Given two functions $\varphi:A\to\mathbb{N}$ and $\psi:A'\to\mathbb{N}$, with $A'\cap A=\emptyset$, define $(\varphi\cup\psi)(x)$ to be $\varphi(x)$, for $x\in A$, and $\psi(x)$, for $x\in A'$. Given two functions $\varphi,\psi:A\to\mathbb{N}$, $\varphi+\psi$ denotes it sum and we say that $\phi\leq\psi$ if, for any $x\in A$, $\phi(x)\leq\psi(x)$. We denote by $\varnothing$ the empty function. We often need contiguous sets of indices, for $n,m\in\mathbb{N}$, we define $[n]=\{1,\dots,n\}$, $[-m]=\{-m,\dots,-1\}$, and $[-m,n]=\{-m,\dots,-1,0,1,\dots,n\}$.

Given two graphs G and H we say that $\chi:V(G)\to V(H)$ is a *H-coloring* of G if for any $\{x,y\}\in E(G)$, $\{\chi(x),\chi(y)\}\in E(H)$.

A *partial weight assignment* on a graph H is a triple (H,C,K), where $C\subseteq V(H)$ and $K:C\to\mathbb{N}$. We refer to the vertices of C as the *weighted vertices*. For each partial weight assignment (H,C,K), we use the notations $S=V(H)-C$, $h=|V(H)|$, $c=|C|$, $s=|S|$ and $k=\sum_{c\in C}K(c)$. We say that a (H,C,K) is *positive* if $K(a)\neq 0$ for any $a\in C$.

Given a partial weight assignment (H,C,K), $\chi:V(G)\to V(H)$ is a *(H,C,K)-coloring* of G if χ is an H-coloring of G such that for all $a\in C$, we have $|\chi^{-1}(a)|=K(a)$ (see Figure 1). We denote by $\mathcal{H}_{(H,C,K)}(G)$ the set of all the (H,C,K)-colorings of G, often we drop the subscript when (H,C,K) is clear from the context. For a fixed partial weight assignment (H,C,K), the (H,C,K)*-coloring* problem asks whether there exists an (H,C,K)-coloring of an input graph G. In this definition the parameters are the integers in the images of K. The parameterized independent set and vertex cover problems are particular instances of the (H,C,K)-coloring problem (see Figure 2)

For a fixed graph H, given a graph G, a (H,G)*-list* is a function $L : V(G) \to 2^{V(H)}$ mapping any vertex of G to a subset of $V(H)$. For any $v \in V(G)$, the *list* of v is the set $L(v) \subseteq V(H)$. The *list* H*-coloring* problem asks whether, given a graph G and an (H,G)-list L, there is an H-coloring χ of G so that for every $u \in V(G)$, $\chi(u) \in L(u)$. This problem can be parameterized in the same way as the H-coloring to define the *list* (H,C,K)*-coloring* and the *list* $(H,C,\leq K)$*-coloring* problems. We denote by $\mathcal{H}_{(H,C,K)}(G,L)$ the set of all the (H,C,K)-colorings of (G,L). As before we will use $\mathcal{H}(G,L)$ when (H,C,K) is clear from the context.

As usual, the $\#(H,C,K)$-coloring, the list $\#(H,C,K)$-coloring, the $\#(H,C,\leq K)$-coloring and the the list $\#(H,C,\leq K)$-coloring problems denote the counting versions of the corresponding decisional problems.

Notice that in contraposition to the H-coloring, an (H,C,K)-coloring, in general, cannot be obtained as the disjoint union of (H,C,K)-colorings for the connected components of an input graph G. This is expressed in the next lemmas, whose proof is straightforward.

Lemma 1. *Let (H,C,K) and (H,C,K) be two partial weight assignments. Let G be a graph with connected components G_i, $i = 1,\ldots,g$, and let χ_i be a (H,C,K_i)-coloring of G_i, for any i, $1 \leq i \leq g$. Then, $\chi^* = \chi_1 \cup \cdots \cup \chi_t$ is a $(H,C,K_1 + \cdots + K_t)$-coloring of G.*

Lemma 2. *Let (H,C,K) and (H',C',K') be two partial weight assignments. Let G and G' be two graphs, let χ be an (H,C,K)-coloring of G and let χ' be an (H',C',K')-coloring of G'. Then $\chi \cup \chi'$ is a $(H \cup H', C \cup C', K \cup K')$-coloring of $G \cup G'$.*

Finally, we define some types of partial weight assignments on connected graphs, an illustration of these types is given in Figure 3.

Let H be a connected graph, a partial weight assignment is a *1-component* if $E(H-C) = \emptyset$; a *2-component* if H is a reflexive clique; a *3-component* if H is a complete bipartite graph and $C \subsetneq Y(G)$; a *4-component* if $H[C]$ is a reflexive clique with all its vertices adjacent with one looped vertex of $H-C$; and a *6-component* if H is bipartite, $C \subsetneq Y(H)$, there is a vertex $x \in X(H)$ adjacent to all the vertices in C, and at least one vertex in $Y(H) - C$.

A partial weight assignment (H,C,K) is said to be *simple* when each connected component H' is either a 1-component, a 2-component or a 3-component; *plain* when each connected component H' of H is either a 1-component, a 4-component, or a 6-component. Observe that the simple partial weight assignment are a generalization of the *compact* partial weight assignment (only 1 and 2-components) used in [5].

3 Counting (H,C,K)-colorings of connected graphs

We introduce some vertex-weighted graphs and extend the notion of list (H,C,K)-coloring to them. A *tribal graph* is a graph G together with a positive weight assignment $p : V(G) \to \mathbb{N}^+$. We call the vertices in $T(G) = \{v \in V(G) \mid p(v) > 1\}$ *tribes* of G and the vertices in $I(G) = V(G) - T(G)$ *individuals* of G. The *expansion* $\widehat{G}$ of a tribal graph is a graph in which each $v \in T(G)$ is replaced by $p(v)$ copies of v. We associate a tribal graph with a subset of vertices of a graph, typically with "few" vertices, and an equivalence relation on the remaining vertices, represented by tribes. Our kernels will be obtained by restricting the maximum weight of a tribe, in the following way: For a non negative integer r, the *r-restriction* of a tribal graph (G,p), is the tribal graph (G,p') where, for any $v \in V(G)$, $p'(v) = \min\{p(v), r\}$. For sake of readability we denote a tribal graph as $\widetilde{G}$, eliminating the explicit reference to the node weights.

A *list* (H,C,K)*-coloring* of a tribal graph $\widetilde{G}$ and an $(H,\widetilde{G})$-list L is a mapping $w: V(\widetilde{G}) \times V(H) \to [k]$ where: (1) for any $v \in V(\widetilde{G})$ and $a \in S$, $w(v,a) \leq 1$; (2) for any $v \in V(\widetilde{G})$ and $a \in C$, $w(v,a) \leq K(a)$; (3) for any $v \in V(\widetilde{G})$, $1 \leq \sum_{a\in H} w(v,a) \leq p(v)$; (4) if $\{v,u\} \in E(\widetilde{G})$ then for all $a,b \in H$ with $w(v,a) > 0$ and $w(u,b) > 0$, $\{a,b\} \in E(H)$; (5) for any $a \in C$, $\sum_{v\in V(\widetilde{G})} w(v,a) = K(a)$; and for any $v \in V(\widetilde{G})$ and $a \in V(H)$ with $w(v,a) > 0$, $a \in L(v)$.

An example of a list (H,C,K)-coloring of a tribal graph is given in Figure 4. Notice that when $C = \emptyset$ the above definition coincides with the concept of list H-coloring. Moreover, in case $p(v) = 1$ for any $v \in V(\widetilde{G})$, we obtain a list (H,C,K)-coloring of $(\widetilde{G},L)$ by assigning to each vertex in $\widetilde{G}$ the unique vertex in $V(H)$ for which $w(v,a) = 1$. Also, by removing condition 6 we get a definition of (H,C,K)-coloring of a tribal graph.

The following result gives some initial results on colorings for tribal graphs. We use $\widetilde{n}$ ($\widetilde{m}$)to denote the number of vertices (edges) of a tribal graph $\widetilde{G}$.

Lemma 3. *Let (H,C,K) be a partial weight assignment. Let $\widetilde{G}$ be a tribal graph and let L be a $(H,\widetilde{G})$-list. Given $w: V(\widetilde{G}) \times V(H) \to [k]$ we can check whether w is a list (H,C,K)-coloring of $(\widetilde{G},L)$ in time $O(h^2\widetilde{n} + \widetilde{m})$. Furthermore, $|\{w : V(\widetilde{G}) \times V(H) \to [k]| \leq ((k+1)^c 2^s)^{\widetilde{n}}$.*

Our next step is to introduce an equivalence relation on the vertices of G and construct a tribal graph from there. Let (H,C,K) be a partial weight assignment, given a graph G, together with a (H,G)-list L. Define $\mathcal{P}$ to be the partition of $V(G)$ induced by the equivalence relation,

$$v \sim u \text{ iff } [N_G(v) = N_G(u) \ \wedge \ L(v) = L(u)].$$

For $u \in V(G)$, P_v denotes the set $\{u \mid u \sim v\}$ and for any $Q \in \mathcal{P}$, we select a representative vertex $v_Q \in Q$. Figure 5 illustrates the equivalence.

In general, we keep all the vertices in some classes as individuals and one representative of each remaining classes as tribes. We say that $R \subseteq V(G)$ is a *closed set* if for any $v \in R$ we have $P_v \subseteq R$. Let R be a closed set, the *tribal graph associated to* (G,R) is $\widetilde{G} = (G[R \cup \{v_Q \mid Q \cap R = \emptyset\}], p)$, where $p(v) = 1$ when $v \in R$, and $p(v_Q) = |Q|$ for any $Q \cap R = \emptyset$.

Let $\widetilde{G}_{k+s}$ be the $k+s$-restriction of the tribal graph $\widetilde{G}$ associated to G and some closed set R. The following property establishes the relationship between $\mathcal{H}(G,L)$ and $\mathcal{H}(\widetilde{G}_{k+s},L)$ showing the correct use of $\widetilde{G}_{k+s}$ as compactor. Observe that as $V(\widetilde{G})$ can be seen as a subset of $V(G)$, we keep the name of the associated (H,G) list as L in order to simplify notation.

Lemma 4. *Let (H,C,K) be a partial weight assignment. Given a graph G together with a (H,G)-list L and a closed set R, let $\widetilde{G}$ be the* tribal *graph associated to (G,R). Then, there is a surjective function from $\mathcal{H}(G,L)$ into $\mathcal{H}(\widetilde{G}_{k+s},L)$.*

In the next result we show how to compute the number of extensions of a given list (H,C,K)-coloring w of $(\widetilde{G}_{k+s},L)$ to a list (H,C,K)-coloring of (G,L), according to Lemma

4 and using the function W which for any $v \in T(\widetilde{G})$ is defined as follows

$$W(w,v) = \binom{|P_v|}{\tau} \frac{\tau!}{\prod_{a\in C} w(v,a)!} \binom{|P_v|-\tau}{\sigma} \sigma! \sigma^{|P_v|-\tau-\sigma},$$

where $\sigma = \sum_{a\in S} w(v,a)$ and $\tau = \sum_{a\in C} w(v,a)$. As it is usual when dealing with a counting problem, we assume that all basic arithmetic operations take constant time.

Lemma 5. *Let (H,C,K) be a partial weight assignment. Given a graph G together with a (H,G)-list L and a closed set R, let $\widetilde{G}$ be the* tribal *graph associated to (G,R) and let w be a list (H,C,K)-coloring of $(\widetilde{G}_{k+s}, L)$. Then,*

$$|\{\chi \in \mathcal{H}(G,L) \mid w = w_\chi\}| = \prod_{v\in T(\widetilde{G})} W(w,v).$$

Furthermore, for any $v \in T(\widetilde{G})$ the value $W(w,v)$ can be computed in $O(\log h + c\log k + \log n)$ given, w, $\widetilde{G}$ and $|P_v|$.

The central idea of the algorithm is to solve the list $\#(H,C,K)$-coloring problem by enumerating all the list $\#(H,C,K)$-colorings of $\widetilde{G}_{k+s}$. Given $w \in \mathcal{H}(\widetilde{G}_{k+s}, L)$ we have to compute the number of possible extensions to a list (H,C,K)-coloring of (G,L) and the overall sum of these numbers will be the number of the list (H,C,K)-colorings of (G,L), according to Lemmata 4 and 5.

```
function Count-List-Colorings(H, C, K, G, L, R)
    input A partial weight assignment (H, C, K) on H such that E(H − C) = ∅
          A graph G and a (H, G)-list L,
    output The number of (H, C, K)-colorings of (G, L).
    begin
        GT = Tribal(H, C, K, G, L, R)
        GTR = Restrict(GT, k + s)
        ν = 0
        for all w : V(GTR) × V(H) → {0, ..., k} do
            if w ∈ H(GTR, L)
              then
                ν = ν + ∏_{v∈T(GT)} W(w, v)
            end if
        end for
        return ν
    end
```

Theorem 1. *Let (H,C,K) be a partial weight assignment. Given a graph G together with a (H,G)-list L and a closed set R with r vertices. Then, algorithm* Count-List-Colorings *outputs the number of list (H,C,K)-colorings of (G,L) within*
$O(n(n+h) + (n+h)2^{n+h} + (h^2\widetilde{n} + \widetilde{m})\left((k+1)^c 2^s\right)^{\widetilde{n}})$,
steps, where $\widetilde{n}$ and $(\widetilde{m})$ are the number of vertices (edges) of the tribal graph $\widetilde{G}$ associated to (G,R).

In the following subsections we especify the particular definition of the equivalence relation for the case of 1,2 and 3-components.

3.1 The compactor for a 1-component

In this case the compactor is obtained after splitting the vertices of G according to degree and connectivity properties. Given a graph G and an integer k, the *k-splitting* of G is a partition of $V(G)$ in three sets, (R_1, R_2, R_3) where R_1 is the set of vertices in G of degree $> k$, R_2 is formed by the non isolated vertices in $G' = G[V(G) - R_1]$ and R_3 contains the isolated vertices in G'.

Lemma 6. *Let (H, C, K) be a partial weight assignment, where $H - C$ is edge-less. Given a graph G, let (R_1, R_2, R_3) be the k-splitting of G. If $|R_1| > k$ or $|R_2| > k^2 + k$, then $\mathcal{H}(G, L) = \emptyset$. Furthermore, if $\mathcal{H}(G, L) \neq \emptyset$ the k-splitting of G can be computed in $O(kn)$ steps.*

The next result is obtained taking into account that all the vertices in R_3 have degree at most k and that for any $v \in R_3$ we have $N_G(v) \subseteq R_1$.

Lemma 7. *Let (H, C, K) be a partial weight assignment, where $H - C$ is edge-less. Given a graph G and a (H, G)-list, let (R_1, R_2, R_3) be the k-splitting of G. Then, $R = R_1 \cup R_2$ is a closed set. Furthermore, the tribal graph $\widetilde{G}$ associated to (G, R) has at most $k^2 + 2k + 2^{k+h}$ vertices and $(k^2 + 2k)^2 + k2^{k+h}$ edges, and $\widetilde{G}$ can be computed in $O((h + k)n + (h + k)2^{k+h})$ steps.*

Theorem 2. *Let (H, C, K) be a partial weight assignment, where $H - C$ is edge-less. Given a graph G and a (H, G)-list L, let (R_1, R_2, R_3) be the k-splitting of G and let $\widetilde{G}$ be the tribal graph associated to $(G, R_1 \cup R_2)$. Then, $|\mathcal{H}(G, L)| = \sum_{w \in \mathcal{H}(\widetilde{G}, L)} \mathsf{NbExt}(w)$. Furthermore, the adaptation of algorithm $\mathsf{Count\text{-}List\text{-}Colorings}$ outputs the number of list (H, C, K)-colorings of (G, L) within*
$O((h + k)n + (h^2k^2 + k^4 + (\log h + c \log k + \log n)2^{h+k})\,((k + 1)^c 2^s)^{k^2+2k+2^{h+k}})$ steps.

3.2 The compactor for a 2-component

We start with a trivial result that allow us to remove all the edges from G. So, let $G' = (V(G), \emptyset)$.

Lemma 8. *Let (H, C, K) be a weight assignment, where $H = K_h^r$. Given a graph G and an (H, G)-list L. Then, $\mathcal{H}(G, L) = \mathcal{H}(G', L)$.*

Given a graph G together with a (H, G)-list L, the *list splitting* of (G, L) is a partition of $V(G)$ in two sets, (N_1, N_2) where $N_1 = \{v \in V(G) \mid L(v) \cap S = \emptyset\}$. From the previous definition, we can find a condition for the existence of list colorings, taking into account that in any (H, C, K)-coloring χ of G, $|\chi^{-1}(C)| = k$, and that all the vertices in N_1 must be mapped to C.

Lemma 9. *Let (H, C, K) be a weight assignment, where $H = K_h^r$. Given a graph G and an (H, G)-list L, let (N_1, N_2) be the list splitting of (G, L). Then, if $|N_1| > k$, there are no (H, C, K)-colorings of (G, L). Furthermore, if $\mathcal{H}(G, L) \neq \emptyset$, the list splitting can be computed in $O(hn)$ steps.*

As G' has no edges, the number of classes induced by the $\sim$ equivalence is at most 2^h. So, from 1, taking into account the complexity of constructing the list splitting, we have

Theorem 3. *Let (H, C, K) be a partial weight assignment with $H = K_h^r$. Given an edge-less graph G and a (H, G)-list L, let (N_1, N_2) be the list-splitting of (G, L). Then N_1 is a closed set and the tribal graph $\widetilde{G}$ associated to (G', N_1) has at most $k + 2^h$ vertices and no edges. Furthermore, $|\mathcal{H}(G, L)| = \sum_{w \in \mathcal{H}(\widetilde{G}, L)} \mathsf{NbExt}(w)$ and the adaptation of algorithm* Count-List-Colorings *outputs the number of list (H, C, K)-colorings of (G, L) within*
$O(hn + 2^h + (h^2(k + 2^h) + (\log h + c \log k + \log n)2^h)((k + 1)^c 2^s)^{k+2^h})$ *steps.*

3.3 The compactor for a 3-component

We start as in the case of 2-components by a simplification. Given a connected bipartite graph G, let $G' = (X(G), Y(G), X(G) \times Y(G))$.

Lemma 10. *Let (H, C, K) be a partial weight assignment with $H = K_{xy}$ and $C \subseteq Y(H)$. Given a connected bipartite graph G, let $G' = (X(G), Y(G), X(G) \times Y(G))$. Then, $\mathcal{H}(G, L) = \mathcal{H}(G', L)$.*

Let (H, C, K) be a partial weight assignment with $H = K_{xy}$ and $C \subseteq Y(H)$. Given a complete bipartite graph G together with a (H, G)-list L. The *bipartite splitting* of (G, L) with respect to (H, C, K) is a partition of $V(G)$ in three sets, (M_1, M_2, M_3) where $M_3 = \{v \in V(G) \mid L(v) \cap S \neq \emptyset\}$, $M_1 = X(G) - M_3$ and $M_2 = Y(G) - M_3$.

Lemma 11. *Let (H, C, K) be a partial weight assignment with $H = K_{xy}$ and $C \subseteq Y(H)$. Given a complete bipartite graph G, let (M_1, M_2, M_3) be the bipartite splitting of (G', L). Then, if $|M_1| > 0$ and $|M_2| > 0$, or $|M_1| + |M_2| > k$, there are no list (H, C, K)-colorings of (G, L). Furthermore the bipartite splitting can be computed in $O(hn)$ steps.*

Observe that, as G' has exactly two neighborhoods, there are at most 2^{h+1} equivalence classes. From Lemma3 and Theorem 1, we have

Theorem 4. *Let (H, C, K) be a partial weight assignment with $H = K_{xy}$ and $C \subseteq Y(H)$. Given a complete bipartite graph G together with a (H, G)-list L, let (M_1, M_2, M_3) be the bipartite list partition of (G, L). Then, $M = M_1 \cup M_2$ is a closed set. The tribal graph $\widetilde{G}$ associated to (G', M) has at most $k + 4^h$ vertices and $(k + 2^h)2^h$ edges, and it can be computed in $O(hn + (h + 1)2^h)$ steps. Furthermore, the adaptation of algorithm* Count-List-Colorings *outputs the number of list (H, C, K)-colorings of (G, L) within a number of steps*
$O(hn + (h + 1)2^h + (h^2(k + 4^h) + (k + 2^h)2^h + (\log h + c \log k + \log n)4^h)\, ((k + 1)^c 2^s)^{k+4^h})$.

4 Counting list (H, C, K)-coloring for simple weight assignments

In this section we provide an FPT algorithm for counting the number of list (H, C, K)-coloring when all the connected components of H are 1, 2 or 3-components. The graph H is decomposed as the disjoint union of γ components, $H_{-\rho}, \dots, H_\eta$, where H_0 is is the disjoint union of the 1-components, for $\iota \in [\eta]$, H_ι is a 2-component, while for $\iota \in [-\rho]$, H_ι is a 3-component.

Furthermore, the graph G may have different connected components, at least one for each component of H that contains a vertex in C. We assume that G has g connected components, $\{G_1, \dots, G_g\}$. Given a (H, G)-list L, we often have to speak of a list coloring of a component G_i, we will refer to this as a (G_i, L) coloring, understanding that we keep the original list restricted to the adequate sets and that the mapping maps G_i to some connected component of H.

To define an equivalence relation between components of G, we introduce some further notation. For any $i \in [g]$ and $\iota \in [-\rho, \eta]$, let $\widetilde{G}_{i\iota}$ be the tribal graph associated to $(G_i, L_{i\iota})$, according to the type of $(H_\iota, C_\iota, K_\iota)$ through the corresponding splitting. Let $\widehat{G}_{i\iota}$ be the expansion of the $k+s$ restriction of $\widetilde{G}_{i\iota}$. Let $\widehat{G}_{i\iota}$ be relabeled so that if $n_{i\iota} = |V(\widehat{G}_{i\iota})|$, then $V(\widehat{G}_{i\iota}) = [n_{i\iota}]$.

For any $i \in [g]$, $\iota \in [-\rho, \eta]$, and $f : [n_{i\iota}] \to V(H_\iota)$ set $\beta(i, \iota, , f) = 1$ or 0 depending on whether f is a list H_ι-coloring of $\widehat{G}_{i\iota}$ or not.

For any $\iota \in [-\rho, \eta]$ we define the following equivalence relation on $[g]$:

$$i \sim_\iota j \text{ iff } \ n_{i\iota} = n_{j\iota} \text{ and } \forall f : [n_{i\iota}] \to V(H_\iota) \ \ \beta(i, \iota, f) = \beta(j, \iota, f).$$

We say that two components of G are equivalent when they are equivalents with respect to all the components of H. Formally, for $i, j \in [g]$, $i \sim j$ iff $\ \forall \iota \in [-\rho, \eta] \ i \sim_\iota j$. For $i \in [g]$, Q_i denotes the equivalence class of i.

Let $\mathcal{Q}$ be the partition induced in $[g]$ by $\sim$. Notice that we have at most

$$\left((k+s)(k^2+2k+2^{k+h})2^{h^{(k+s)(k^2+2k+2^{k+h})}}\right)^\gamma$$

equivalence classes. Each class can be represented by a sequence of γ pairs $(n_\iota^Q, \beta_\iota^Q)$, formed by an integer and a binary string of length at most $h^{(k+s)(k^2+2k+2^{k+h})}$, representing the size of the vertex set and the associated mask respectively.

We need further definitions to establish the form and properties of the compactor. For any $Q \in \mathcal{Q}$ define

$$\mathcal{F}_0(Q) = \cup_{\iota \in [-\rho,\eta]} \{f : [n_\iota^Q] \to V(H_\iota) \mid \beta_\iota^Q(f) = 1 \text{ and } f^{-1}(C) = \emptyset\}, \text{ and}$$
$$\mathcal{F}_1(Q) = \cup_{\iota \in [-\rho,\eta]} \{f : [n_\iota^Q] \to V(H_\iota) \mid \beta_\iota^Q(f) = 1 \text{ and } f^{-1}(C) \neq \emptyset.\}$$

Finally, the compactor is

$$\begin{aligned}\mathcal{F}(G, L) = \{((A_Q, o_Q, B_Q))_{Q \in \mathcal{Q}} \mid \forall Q \in \mathcal{Q}\ A_Q \subseteq \mathcal{F}_1(Q) \text{ and } o_Q : A_Q \to [k] \text{ and } B_Q \subseteq \mathcal{F}_0(Q)\\ \forall a \in C \sum_{Q \in \mathcal{Q}} \sum_{f \in A_Q} o(f)\, |f^{-1}(a)| = K(a)\}\end{aligned}$$

Lemma 12. *Let (H, C, K) be a simple partial weight assignment. Given a graph G together with a (H, G)-list L, let $\widehat{G}$ be the representative graph of (G, L) as defined previously. Then, there is a surjective mapping from $\mathcal{H}(G, L)$ into $\mathcal{F}(G, L)$.*

We want to compute, for any sequence $\sigma \in \mathcal{F}(G, L)$, the number of list (H, C, K)-coloring χ of (G, L), such that $\sigma_\chi = \sigma$. We can work for each class Q separately.

Once an equivalence class Q is fixed, we count the number of ways to assign the colorings to all the members of the class, taking into account that the colorings are defined over the component compactor, and therefore the same coloring f, used as coloring for one $\widehat{G}_i$ can be extended in a different number ways to G_i than when assigned to another component $j \in Q$. We give a recursive definition that allow us to compute the number of extensions of any given sequence in $\mathcal{F}(G, L)$.

By construction, for any equivalence class Q, we have to define an assignment of components in Q to functions in $A_Q \cup B_Q$, satisfying the properties: any coloring in $f \in A_Q$

must be used $o(f)$ times and any coloring in B_Q must be used at least once. For simplicity we replace A_Q by a multiset in which any function f appears $o_Q(f)$ times. We also make use of the function $W(w, G)$, as defined in Lemma 5. We work on triples (r, M, N) where $0 \leq r \leq n^Q$, $M \subseteq A_Q$ (a multiset), and $N \subseteq B_Q$. We define $T(r, M, N)$ recursively as follows $T(0, M, N) = 1$ when $M = N = \emptyset$, 0 otherwise, for $r > 0$,

$$T(r, M, N) = \sum_{\varphi \in M} T(r-1, M - \{\varphi\}, N) W(w_\varphi, G_r)$$
$$+ \sum_{\varphi \in N} T(r-1, M, N - \{\varphi\}) W(w_\varphi, G_r) + \sum_{\varphi \in N} T(r-1, M, N) W(w_\varphi, G_r).$$

Finally, set $T(Q, \sigma) = T(r_q, F_C, F_S)$.

The above defined function computes the number of ways of extending a a sequence $\sigma \in \mathcal{F}(G, L)$ to a list coloring. From Lemma 12, using induction and dynamic programming we have

Theorem 5. *Let (H, C, K) be a simple partial weight assignment. Given a graph G together with a (H, G)-list L,*

$$|\mathcal{H}(G, L)| = \sum_{\sigma \in \mathcal{F}(G,L)} \prod_{Q \in \mathcal{Q}} T(Q, \sigma).$$

Furthermore $|\mathcal{H}(G, L)|$ can be computed in time $O(\gamma(k+h)n + f_1(k, h)g + f_2(k, h)\log n + f_3(k, h))$, for suitable functions.

5 The list (H, C, K)-coloring problem for simple partial weight assignments

As in the case of counting we start by giving an algorithm for connected graphs and after extend the results to the case of multiple components.

5.1 Connected graphs

In section 3 we showed properties of the associated tribal graph that were adequate for solving the counting problem. In the next lemma we state the simpler property which will suffice for finding an adequate kernel for the decision problem. Observe that we only have to consider the $k+1$-restriction instead of the $k+s$ restriction of the corresponding tribal graphs.

Lemma 13. *Let (H, C, K) be a partial weight assignment. Given a graph G together with a (H, G)-list L and a closed set R, let $\widetilde{G}$ be the tribal graph associated to (G, R). Then, $\mathcal{H}(G, L) \neq \emptyset$ iff $\mathcal{H}(\widetilde{G}_{k+1}, L) \neq \emptyset$. Furthermore, $\mathcal{H}(G, L) \neq \emptyset$ iff $\mathcal{H}(\widehat{G}_{k+1}, L) \neq \emptyset$*

The algorithm for deciding the existence of a list (H, C, K)-coloring problem computes the $k+1$-restriction of the corresponding tribal graph $\widetilde{G}$ (depending on the type of component) according to the definitions given in Subsections 3.1–3.3 and, after computing its expansion $\widehat{G}_{k+1}$, checks whether there is a list (H, C, K)-coloring of $(\widehat{G}_{k+1}, L)$.

Lemma 14. *Let (H, C, K) be a partial weight assignment, let G be a graph, and let L be a (H, G)-list. When G has no edges or $H = K_h^r$, the existence of a list H-coloring of G can be decided in $O(hn)$ steps, while the existence of a list (H, C, K)-coloring of (G, L) can be decided in $O((k+h)n + nk \log k\sqrt{n+k})$ steps. When $H - C$ is edge-less, the existence of a list (H, C, K)-coloring of (G, L) can be decided in $O(2^k c^k[(k+h)n + nk \log k\sqrt{n+k}])$ steps.*

The final piece is to show how to decide the existence of a list (H, C, K)-coloring using the previous tests. Then, taking into account the size of the kernel, we have FPT algorithms for the list (H, C, K)-coloring, when (H, C, K) is a 1, 2 or 3-component. Further details are given in Appendix F.

5.2 Simple partial weight assignments

In this case, the key idea to reduce complexity is that when checking for the existence of a list (H, C, K)-coloring it is enough to keep information about whether a particular partial weight assignment can be covered by a legal list coloring of a component G_i.

To define the compactor we use partial weight assignments for any component of H, so for any $\iota \in [-\rho, \eta]$, define $\mathcal{W}_\iota = \{W : C \to \mathbb{N} \mid \mathbf{0} \leq W \leq K_\iota\}$. Let $k_\iota^* = \prod_{a \in C_\iota} K_\iota(a)$. Notice that $|\mathcal{W}_\iota| = k_\iota^*$ and that $k^* = \prod_{a \in C} K(a) = \prod_{\iota \in [-\rho,\eta]} k_\iota^*$. Finally, let $\mathcal{W}_\iota^+ = \mathcal{W}_\iota - \{\mathbf{0}\}$.

Our next definition will be the basis for classifying the components of G according to the combinations of components of H and partial weight assignments with correct list colorings. Given $i \in [g]$, $\iota \in [-\rho, \eta]$, and $W \in \mathcal{W}_\iota$ define $\alpha(i, \iota, W) = 1$ or 0 depending on whether there is a list-(H_ι, C_ι, W)-coloring of $(\widetilde{G}_{i\iota}, L_{\iota i})$ or not.

We isolate the information that has to capture a set of components of G to certify the existence of a list coloring. Given $A \subseteq [g]$, and $\iota \in [-\rho, \eta]$ define

$$\begin{aligned}
\mathcal{F}(A, \iota) = \{(Z, f) \mid\ & Z \subseteq A, |Z| \leq k_\iota, f : Z \to \mathcal{W}_\iota^+, \\
& \sum_{i \in Z} f(i) = K_\iota \wedge \forall i \in Z\ \alpha(i, \iota, f(i)) = 1\}, \text{ and the compactor is} \\
\mathcal{F}(A) = \{(Z_{-\rho}, f_{-\rho}), \dots, (Z_\eta, f_\eta) \mid\ & Z_1 \dots Z_\eta \text{ form a non trivial partition of } A, \\
& \forall \iota \in [-\rho, \eta]\ f_\iota : Z_\iota \to \mathcal{W}_\iota \\
& \forall \iota \in [-\rho, \eta]\ (Z_\iota, f_\iota) \in \mathcal{F}(A, \iota)\}.
\end{aligned}$$

Thus we can establish the main property of the compactor $\mathcal{F}(A)$. We say that $A \subseteq [g]$ is *proper* if $\mathcal{F}(A) \neq \emptyset$ and for any $i \in [g] - A$, there is $\iota \in [-\rho, \eta]$ such that $\alpha(i, \iota, \mathbf{0}) = 1$.

We establish an equivalence relation to reduce the number of candidate sets as follows

$$i \sim j \text{ iff } \ \forall \iota \in [-\rho, \eta]\ \forall W \in \mathcal{W}_\iota \ \ \alpha(i, \iota, W) = \alpha(j, \iota, W).$$

Let $\mathcal{P}$ be the partition induced by the equivalence classes. Notice that we have at most 2^{k^*} equivalence classes and that each class can be represented by a binary string of length k^*. Let $M \subseteq [g]$ be obtained by keeping $k + 1$ representatives from each class with more than k members, all the class otherwise.

Lemma 15. *Let (H, C, K) be a simple partial weight assignment. Given a graph G together with a (H, G)-list L. Then, $\mathcal{H}_{(H,C,K)}(G, L) \neq \emptyset$ if and only there is a $A \subseteq M$ that is proper.*

Finally, by generating all the sets $A \subseteq M$ and all the elements in $\mathcal{F}(A)$ and performing the adequate test, we have

Theorem 6. *For simple partial weight assignment (H, C, K), the list (H, C, K)-coloring problem can be solved in time $O(\gamma(k + h)n + f_4(k, h)g + f_5(k, h)$.*

6 The (H, C, K)-coloring problem for plain partial weight assignments

In this subsection we provide a fast algorithm for the (H, C, K)-coloring problem, for plain partial weight assignments. In order to prove the fixed parameter tractability we reduce the problem to an equivalent simple partial weight assignment. This goal is achieved through a series of parameterized reductions, from the general form of components to the *small* or *tiny* forms, see Figure 7. We only need to extend to the case of 3-components the reductions presented in [2].

Theorem 7. *For any plain partial weight assignment (H, C, K), the (H, C, K)-coloring problem can be solved in $O(kn + f_6(k, h))$ steps.*

We complement this positive result with a family of non-plain parameterized assignments for which the (H, C, K)-coloring problem is $\mathsf{W}[1]$-hard which, unless $\mathsf{W}[1]{=}\mathsf{FPT}$. This implies that no fixed parameter algorithm exists for such assignments.

Theorem 8. *The (H, C, K)-coloring problem is $\mathsf{W}[1]$-hard when $H = K_1^r \oplus C$ for some C that has at least one unlooped vertex.*

References

1. J. Díaz. H-colorings of graphs. *Bulletin of the European Association for Theoretical Computer Science*, (75):82–92, 2001.
2. J. Díaz, M. Serna, and D. Thilikos. (H, C, K)-coloring: Fast, easy and hard cases. In J. Sgall, A. Pultr, and P. Kolman, editors, *Mathematical Foundations of Computer Science*, volume 2136, pages 304–315. Lecture Notes in Computer Science, Springer-Verlag, 2001.
3. J. Díaz, M. Serna, and D. Thilikos. The restrictive H-coloring problem. *Discrete Applied Mathematics*, to appear.
4. J. Díaz, M. Serna, and D. M. Thilikos. Recent results on parameterized H-colorings In J. Nešetřil and P. Winkler, editors, *Graphs, Morphisms and Statistical Physics*, DIMACS series in Discrete Mathematics and Theoretical Computer Science. American Mathematical Society, 2004. To appear.
5. J. Díaz, M. Serna, and D. M. Thilikos. Complexity issues on Bounded Restrictive H-colorings. Submitted, 2004.
6. R. G. Downey and M. R. Fellows. *Parameterized complexity.* Springer-Verlag, New York, 1999.
7. M. Dyer and C. Greenhill. The complexity of counting graph homomorphisms. *Random Structures Algorithms*, 17:260–289, 2000.
8. T. Feder and P. Hell. List homomorphisms to reflexive graphs. *Journal of Combinatorial Theory (series B)*, 72(2):236–250, 1998.
9. T. Feder, P. Hell, and J. Huang. List homomorphisms and circular arc graphs. *Combinatorica*, 19:487–505, 1999.
10. J. Flum and M. Grohe. The parameterized complexity of counting problems. In *43rd Symposium on Foundations of Computer Science (FOCS 2002)*, page 538. IEEE Computer Society, 2002.
11. H. Gabow and R. Tarjan. Faster scaling algorithms for network problems. *SIAM Journal on Computing*, 18:1013–1036, 1989.
12. P. Hell and J. Nešetřil. On the complexity of H-coloring. *Journal of Combinatorial Theory (series B)*, 48:92–110, 1990.
13. P. Hell and J. Nešetřil. Counting list homomorphisms and graphs with bounded degrees. In J. Nesetril and P. Winkler, editors, *DIMACS Series in Discrete Mathematics and Theoretical Computer Science*, DIMACS/DIMATIA Workshop on Graphs, Morphisms and Statistical Physics. American Mathematical Society, 2003. To appear.
14. C. McCartin. Parameterized counting problems. In K. Diks and W. Rytter, editors, *Mathematical Foundations of Computer Science 2002, 27th International Symposium (MFCS 2002)*, volume 2420 of *Lecture Notes in Computer Science*, pages 556–567, 2002.
15. C. McCartin. *Contributions to Parameterized Complexity.* PhD thesis, Victoria University of Wellington, 2003.

Appendix

A Figures

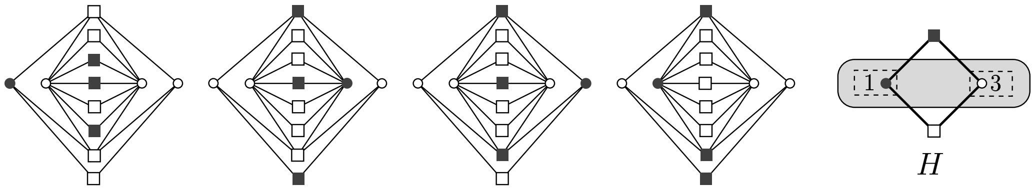

Fig. 1. A graph G a partial weight assignment (H, C, K), and some of its (H, C, K)-colorings. Notice that there are $4 \times 2^8 = 1024$ (H, C, K)-colorings of G.

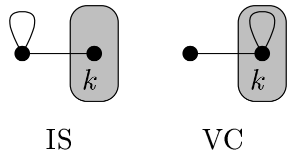

Fig. 2. The partial weight assignments (H, C, K) for the parameterized independent set and vertex cover.

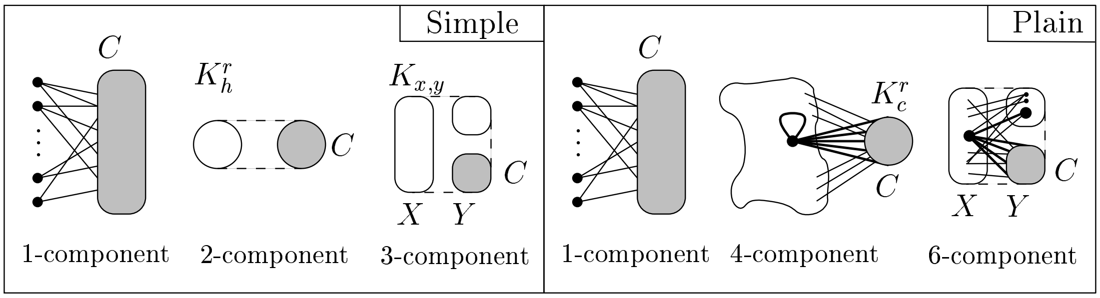

Fig. 3. Components for a plain and simple partial weighted assignments.

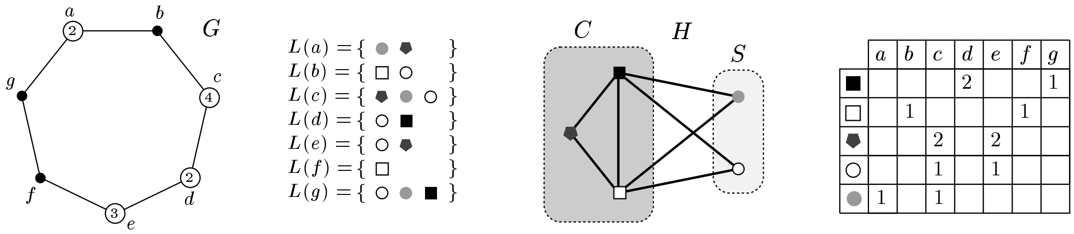

Fig. 4. An example of a (H, C, K)-coloring of the tribal graph G. The individuals of G are denoted by black vertices. In the table representing the (H, C, K)-coloring, empty positions correspond to 0's.

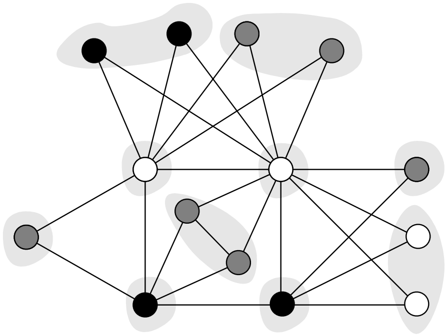

Fig. 5. An example of the partition $\mathcal{R}$ on $V(G)$. Vertices with the same list have the same color.

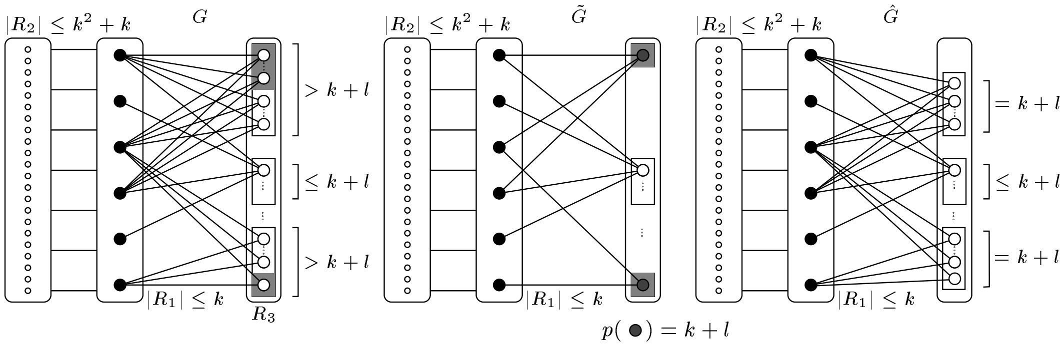

Fig. 6. An example the graphs G, $\widetilde{G}$, and $\widehat{G}$ obtained from the k-splitting of G with respect to a (H, G) list.

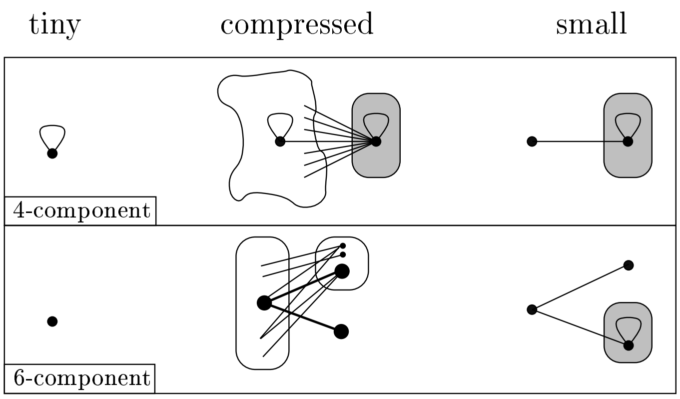

Fig. 7. Some particular cases of components in a plain weight assignment

B Proof of Lemma 4

Proof. We first associate to each list (H, C, K)-coloring χ of (G, L) a list (H, C, K)-coloring w_χ of $(\widetilde{G}_{k+s}, L)$.

Given a list (H, C, K)-coloring χ of (G, L) and $v \in V(\widetilde{G})$ we define the function w_χ, from $V(\widetilde{G}) \times V(H)$ into $\mathbb{N}$, as follows: For $v \in T(\widetilde{G})$,

$$w_\chi(v, a) = \begin{cases} 1 & \text{if } a \in S \text{ and } a \in \chi(P_v), \\ 0 & \text{if } a \in S \text{ and } a \notin \chi(P_v), \\ |\chi^{-1}(a) \cap P_v| & \text{if } a \in C. \end{cases}$$

For $v \in I(\widetilde{G})$,

$$w_\chi(v, a) = \begin{cases} 1 & \text{if } \chi(v) = a, \\ 0 & \text{otherwise.} \end{cases}$$

As χ is a list (H, C, K)-coloring of (G, L), w_χ satisfies conditions 1, 2, 4, 5, and 6 in the definition of tribal list (H, C, K)-coloring (see Page 3). Furthermore, for $v \in T(\widetilde{G})$, $1 \leq \sum_{a \in V(H)} w(v, a)$. Furthermore, from the definition of w_χ we have that $\sum_{a \in V(H)} w(v, a) \leq |P_v|$ and that $|\sum_{a \in V(H)} w(v, a) \leq k+s$, so $\sum_{a \in V(H)} w(v, a) \leq \min\{|P_v|, k+s\} = p(v)$, so condition 3 is also satisfied. Therefore, w_χ is a list (H, C, K)-coloring of $(\widetilde{G}, L)$.

To finish the proof, we show that for any list (H, C, K)-coloring w of $(\widetilde{G}_{k+s}, L)$ there is a list (H, C, K)-coloring χ of (G, L) such that $w_\chi = w$.

Let w be a list (H, C, K)-coloring of $(\widetilde{G}_{k+s}, L)$. We define a mapping χ from $V(G)$ into $V(H)$ as follows. For every vertex $v \in I(\widetilde{G}_{k+s})$, define $\chi(v) = a$ where a is the unique vertex in $V(H)$ such that $w(v, a) = 1$.

For every vertex $v \in T(\widetilde{G})$, with $\sum_{a \in C} w(v, a) = |P_v|$ partition P_v into h disjoint sets (some of them may be empty), $\{A(v, a) \mid a \in V(H)\}$, so that $|A(v, a)| = w(v, a)$ and $\cup_{a \in V(H)} A(v, a) = P_v$. Observe that as there are at least $\sum_{a \in V(H)} w(v, a)$ vertices in P_v we have enough vertices to guarantee that the sets with non zero size are disjoint. For any $u \in A(v, a)$ define $\chi(u) = a$.

For every vertex $v \in T(\widetilde{G})$, with $\sum_{a \in C} w(v, a) < |P_v|$, we know that $\sum_{a \in C} w(v, a) = k + s > k$. Therefore, there is $b \in S$ such that $w(v, b) = 1$. We partition P_v into $h+1$ disjoint sets $\{A(v, a) \mid a \in V(H)\}$ and B, so that $|A(v, a)| = w(v, a)$ and $\cup_{a \in V(H)} A(v, a) \cup B = P_v$. For any $u \in A(v, a)$ define $\chi(u) = a$, and for any $u \in B$ define $\chi(u) = b$.

Taking into account that any vertex in a class gets as image a vertex that was occupy by some representative of its tribe, and that both have the same neighborhood and the same list, it follows that the χ just defined is a list (H, C, K)-coloring of (G, L). Furthermore, it is easy to check that $\chi_w = w$.

C Proof of Lemma 12

Proof. First, to each list (H, C, K)-coloring χ of (G, L) we associate γ triples. Given a list (H, C, K)-coloring χ of (G, L), set χ_i to be the restriction of χ_i to G_i. Set $\chi(i)$ be such that $\chi(G_i) \subseteq H_{\chi(i)}$. Let f_i be the coloring of $\widehat{G}_i$ in $H_{\chi(i)}$ constructed according to Lemma 4 using the adequate relabeling of the vertices.

For each equivalence class $Q \in \mathcal{Q}$ define

$$\chi(Q)^+ = \{f_i \in \chi(Q) \mid f_i^{-1}(C) \neq \emptyset\}, \quad \chi(Q)^- = \{f_i \in \chi(Q) \mid f_i^{-1}(C) \neq \emptyset\}.$$

For any $f \in \chi(Q)^+$ set $o_Q(f) = |\{f_i = f \mid i \in Q\}|\}$. It is straightforward to show that the sequence $\sigma_\chi = ((\chi(Q)^+, o_Q, \chi(Q)^-))_{Q\in\mathcal{Q}}$ belongs to $\mathcal{F}(G, L)$.

Our next step is to show that for any sequence $\sigma \in \mathcal{F}(G, L)$ there is a list (H, C, K)-coloring φ of (G, L) such that $\sigma_\varphi = \sigma$. Observe that by construction, for any equivalence class Q, we have to define an assignment of components in Q to functions in $A_Q \cup B_Q$ that verifies the following properties:

(1) Each one of the mappings $f \in A_Q$ is assigned to exactly $o_Q(f)$ components in Q.
(2) Any coloring in B_Q is assigned to at least one component in Q.

In this way we end up with an assignment that associates to any component of G a mapping. For any $i \in [g]$, let φ_i be the coloring obtained after extending to G_i the mapping assigned to i according to Lemma 4, and let φ be its disjoint union. From the definition of $\mathcal{F}(G, L)$, and Lemmata 4, 1, and 2 guarantees that, for all $a \in C$, $|\varphi^{-1}(a)| = k(a)$. Furthermore, φ is a list H-coloring of $(\widehat{G}, L)$ as a consequence of the definition of equivalence between components and its definition.

D Sketch of proof for Theorem 5

The computation of the number of list (H, C, K)-colorings for simple partial weight assignment follows three main phases:

1. For any $i \in [g]$ and any $\iota \in [-\rho, \eta]$ we have to compute the corresponding $\widehat{G}_{i\iota}$. All those computations together require,

$$\sum_{i=1}^{g} \gamma n_i(h + k) + \gamma(h + k)2^{k+h} = \gamma(h + k)n + \gamma(h + k)2^{k+h}$$

   steps. In the above formula we have take the most costly computation of the kernel for the 1-component.
2. The second step is to compute the values of the β function. We do this together with other tasks. At the end of the phase we will end up with information for each class $Q \in \mathcal{Q}$: the value n_Q and its mask. Also the sets $\mathcal{F}_0(Q)$ and $\mathcal{F}_0(Q)$. All this computation can be done using only the graphs $\widehat{G}_{i\iota}$. Therefore, the overall time depends on g but not on n. This gives the contribution $gf_1(k, h)$ with an additive component that depends only on the parameters.
3. The last phase is implemented by an enumeration of $\mathcal{F}(G, L)$. For each element and class the dynamic programming algorithm is implemented. Notice that the size of the table is a function of the parameters. Therefore the unique dependence in n is given by the computation of the function W. Thus, the overall contribution is $f_2(k, h) \log n$ and additive factors that only depend on the parameters.

E Proof of Lemma 13

Proof. If $w \in \mathcal{H}(\widetilde{G}_{k+1}, L)$ using the construction described in the proof of Lemma 4, we get a list (H, C, K)-coloring of (G, L).

If $\chi \in \mathcal{H}(G, L)$ as, for any equivalence class Q with size bigger than $k+1$, we have that $|\chi(Q) \cap S| > 1$, we can select one vertex $a_Q \in \chi(Q) \cap S$. Now we define χ' from $V(G)$ to $V(H)$ as follows

$$\chi'(u) = \begin{cases} \chi(u) & \text{if } u \in R_1 \cup R_2, \\ \chi(u) & \text{if } u \in R_3 \wedge |P_u| \leq k+1, \\ \chi(u) & \text{if } u \in R_3 \wedge |P_u| > k+1 \wedge \chi(u) \in C, \\ a_{P_u} & \text{otherwise .} \end{cases}$$

As $N_{\widetilde{G}(v)} \subseteq I(\widetilde{G})$ for any $v \in T(\widetilde{G})$, χ' is a list (H, C, K)-coloring of (G, L). Furthermore, for any equivalence class Q with $|Q| > k+1$, it holds that $|\chi'(Q) \cap S| = 1$ and the construction described in Lemma 4 produces a list (H, C, K)-coloring of $(\widetilde{G}_{k+1}, L)$.

Finally, the result concerning the extension $\widehat{G}_{k+1}$ of $\widetilde{G}_{k+1}$ follows from the fact that the tribal graph associated to $(R \cap V(\widehat{G}_{k+1}), \widehat{G}_{k+1})$ is $\widetilde{G}_{k+1}$.

F Proof of Lemma 14

We split the proof in two parts, one for each of the two cases reflected in Lemma 14.

F.1 Two extreme cases: $E(G) = \emptyset$ or $H = K_h^r$

We start by giving algorithms for computing a list (H, C, K)-coloring in two extreme cases: the case when $E(G) = \emptyset$ and H is any graph; and the case when H is a reflexive clique and G is unrestricted.

Notice that if G is edge-less or H is a reflexive clique any mapping from $V(G)$ to H is a valid H-coloring of G, thus we have

Lemma 16. *Let H and G be two graphs such that G has no edges or $H = K_h^r$. Given a (H, G)-list L, there is a list H-coloring of (G, L) iff for all $v \in V(G)$, $L(v) \neq \emptyset$.*

As a corollary from this result we obtain an algorithm for computing a list H-coloring for the two cases.

```
function Basic(H, G, L)
    begin
        for all v ∈ V(G) do
            if L(v) = ∅ then return false  end if
        end for
        return true
    end
```

Trough the paper, we assume G (and H) is given by its adjacency-list representation and a (H, G) list by a boolean array of size nh, we get the following time bounds.

Corollary 1. *Let H and G be two graphs such that either G has no edges or $H = K_h^r$. Given a (H,G)-list L, function* Basic *decides in $O(hn)$ steps whether a list H-coloring of (G,L) exists.*

Our next results shows how to relate list (H,C,K)-colorings of (G,L) with maximum weight matchings in an associated bipartite graph.

Let (H,C,K) be a partial weight assignment, and let L be a (H,G)-list. We define the *associated bipartite* weighted graph $B = B(H,C,K,G,L)$ where

- $X(B) = V(G)$,
- for any $a \in C$ let $A(a)$ be a set with $K(a)$ new vertices.
- $Y(B) = \cup_{a\in C} A(a)$,
- for $u \in X(B)$ and $b \in Y(B)$, there is an edge $\{u,b\}$ if, for some $a \in L(u) \cap C$, $b \in A(a)$.

Finally, if $v \in X(B)$ and $L(v) \cap S \neq \emptyset$, the weight of all edges having v as endpoint is 1. All the remaining edges get weight k. We will usually refer to $X(B)$ as the G-part and to $Y(B)$ as the C-part of B. In Figure 8 we give an example of the construction.

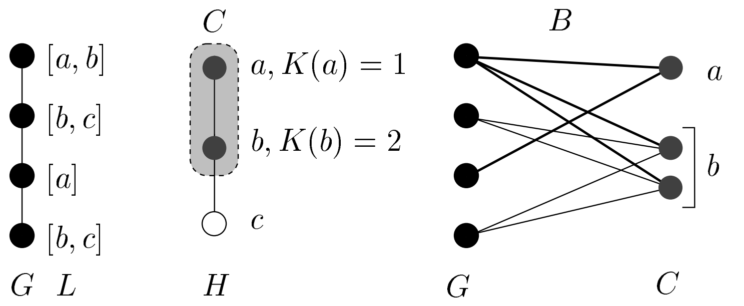

Fig. 8. A partial weight assignment (H,C,K), a graph G, a (G,H)-list, and the associated weighted bipartite graph $B = B(H,C,K,G,L)$. In B, the "fat" edges have weight 3 and the rest have weight 1.

Lemma 17. *Let (H,C,K) be a partial weight assignment, G a graph, and L a (H,G)-list. The weighted bipartite graph $B = B(H,C,K,G,L)$ associated to (G,L) can be computed in $O((k+h)n)$ steps.*

Proof. For any vertex in $V(G)$ its list of neighbors can be computed in $h+k$ steps by examining the bits in $L(u)$ and adding the corresponding elements. The list for the other half can be obtained y reversing the previously computed information in the same time bounds.

Next we relate the existence of list colorings to the weight of a maximum matching in B.

Lemma 18. *Let (H,C,K) be a partial weight assignment, G a graph, and L a (H,G)-list and let $B = B(H,C,K,G,L)$ be the bipartite graph associated to (G,L). Assume that G is edge-less or $H = K_h^r$. Let $p = |\{v \in V(G) \mid L(v) \cap S = \emptyset\}|$. If $p > k$, there is no list (H,C,K)-coloring of (G,L). Otherwise, there is a list (H,C,K)-coloring of (G,L) iff the weight of a maximum matching in B is $kp + k - p$.*

Proof. If $p > k$, then (G, L) has no list (H, C, K)-colorings as the maximum number of vertices in G that can be mapped to C is k, and all the vertices $v \in V(G)$ with $L(v) \cap S = \emptyset$ must be mapped to C. Assume now that $p \leq k$.

If χ is a list (H, C, K)-coloring of (G, L), for every $a \in C$, assigning the vertices in $\chi^{-1}(a)$ to different copies of a in B gives a matching with weight $pk + k - p$. Observe that, by construction this is the maximum weight of any selection of k vertices in the G-part of B, therefore, it is a maximum weight matching.

By definition of B, in any maximum weight matching M of B with $w(M) = kp + k - p$, all the p vertices that must be mapped to C appear in the matching. This gives the kp contribution. The $k - p$ contribution comes from the fact that all the vertices in the C part are covered by M. Therefore, we can extract from M a partial (H, C, K)-coloring of (G, L) that covers properly C. As all the remaining vertices can be mapped legally to some vertices in S, the mapping can be extended to a list (H, C, K)-coloring of (G, L).

As a corollary we obtain an algorithm for computing a list (H, C, K)-coloring.

```
function List-basic(H, C, K, G, L)
    begin
        p = 0
        for all v ∈ V(G) with L(v) ∩ S = ∅ do
            p = p + 1
            if p > k then return ∅ end if
        end for
        Construct the weighted bipartite graph B(H, C, K, G, L)
        Compute a maximum matching M of B
        if w(M) < kp + k − p then return false  end if
        V′ = {v ∈ V(G) | v is not matched in M}
        G′ = (V′, ∅)
        forall v ∈ V′ set L′(v) = L(v) ∩ S end for
        return Basic(H[S], G′, L′)
    end
```

Corollary 2. *Let (H, C, K) be a partial weight assignment, let G be a graph, and let L be a (H, G)-list. When G has no edges or $H = K_h^r$, List-basic correctly decides in $O((k + h)n + nk \log k\sqrt{n + k})$ steps whether a list (H, C, K)-coloring of (G, L) exists.*

Proof. The correctness follows from Lemma 18. From Lemma 17 we have that computing the associated bipartite graph takes $O((k + h)n)$ time, this bound dominates all the remaining steps except for computing the maximum weighted matching in B. We use the Gabow-Tarjan algorithm for computing a maximum weighted matching of a bipartite graph in time $O(m\sqrt{n} \log N)$, where m is the number of edges, n the number of vertices and N is the maximum weight of an edge [11]. As B has at most nk edges and $n + k$ vertices the claimed bound follows.

F.2 The case when $H - C$ is edgeless

As we have shown we can reduce the list (H, C, K)-coloring problem to the problem of the existence of a list (H, C, K)-colorings of the graph $\widehat{G}$ (the expansion of the $k+1$-restriction of the tribal graph associated to the k-splitting. We use the self-reducibility of the list H-coloring problem to design an algorithm to compute a list (H, C, K)-coloring, if one exists. Now we recall a well known fact.

Lemma 19. *Consider a graph G, a partial weight assignment (H, C, K), a (H, G)-list L, a vertex $v \in V(G)$, and a vertex $a \in C$ with $K(a) > 0$. Define $G' = G[V(G) - \{v\}]$, $K'(b) = K(b)$ for $b \neq a$ and $K'(a) = K(a) - 1$. For any $u \in V(G')$ let*

$$L'(u) = \begin{cases} L(u) & \text{if } u \notin N_G(v), \\ L(u) \cap N_H(a) & \text{otherwise.} \end{cases}$$

Then (G, L) has a list (H, C, K)-coloring χ where $\chi(v) = a$ iff (G', L') has a list (H, C, K')-coloring.

We use the above self-reduction to design a recursive algorithm to compute a list (H, C, K)-coloring when one exists or the empty function otherwise.

```
function Edgeless(H, C, K, G, L)
    begin
        if L(v) = ∅ for some v ∈ V(G) then return false end if
        if K = 0 ∧ E(G) ≠ ∅ then return false end if
        if K = 0 ∧ E(G) = ∅ then return Basic(H[S], G, L) end if
        if E(G) = ∅ then return List-basic(H, C, K, G, L) end if
        Select an edge e ∈ E(G)
        for all endpoints v of e do
            G' = G[V(G) − {v}]
            for all a ∈ C with K(a) > 0 do
                for all b ∈ C − {a} set K'(b) = K(b) end for
                K'(a) = K(a) − 1
                for all u ∈ V(G') set L'(u) = L(u) end for
                for all u ∈ N_G(v) set L'(u) = L'(u) ∩ N_H(a) end for
                if Edgeless(H, C, K', G', L') then return true end if
            end for
        end for
        return false
    end
```

Lemma 20. *Let (H, C, K) be a weight assignment, where $H - C$ is edge-less. Given an input graph G, function Edgeless decides whether there is a list (H, C, K)-coloring of (G, L) in $O(2^k c^k[(k + h)n + nk \log k\sqrt{n + k}]))$ steps.*

Proof. The correctness of Edgeless follows from Lemma 19 together with the observation that, for any edge in G, a (H, C, K)-coloring of G should map at least one of its endpoints to C. The algorithm applies the self-reduction on any possible mapping of any endpoint of a selection of $\min\{k, |E(G)|\}$ edges to some of the vertices in C. If for at least one of these choices, (G', L') has a list (H, C, K')-coloring that can be extended to a list (H, C, K)-coloring of (G, L) we compute one. Otherwise G does not have any. There are two trivial base cases, when $L(v) = \emptyset$ or when $K = \mathbf{0}$ and $E(G) = \emptyset$, in which there are no colorings of (G', L'). In the other cases $K = \mathbf{0}$ or $E(G) = \emptyset$, so we are in one of the extreme cases that can be solved according to Corollaries 1 and 2 using functions Find-basic or Find-list-basic.

The time required for the application of the self-reduction involved in the main loop is $O(n + k)$. Furthermore, we consider at most k edges and the 2^k possibilities of end-point selections, together with the c^k possible assignments for any selection. Thus the overall number of calls to the extreme cases is $O(2^k c^k)$. Taking into account the time bound for the List-basic function given in Corollary 2 the claimed bound follows.

F.3 1, 2 and 3-components

As pointed out before the algorithms for these cases starts computing the corresponding kernel $\widetilde{G}_k + 1$, as defined in Sections 3.1, 3.2, and 3.3. The next step is to construct the expansion $\widehat{G}_k + 1$ of $\widetilde{G}_k + 1$. An apply the corresponding algorithm. For the case of a 1-component, we use Algorithm Edgeless and for the case of a 2-component Algorithms List-basic. To complete the picture we give know the steps needed to solve the list (H, C, K)-coloring for the case of 3-components, algorithm Bipartite. We refer to such combination as Algorithm List-Coloring-C

Given a connected bipartite graph G together with a (H, G)-list L, we define two (H, G)-lists, L_1 and L_2, as follows: for any $u \in V(G)$

$$L_1(u) = \begin{cases} L(u) \cap X(H) & \text{if } u \in X(G), \\ L(u) \cap Y(H) & \text{if } u \in Y(G), \end{cases} \qquad L_2(u) = \begin{cases} L(u) \cap Y(H) & \text{if } u \in X(G), \\ L(u) \cap X(H) & \text{if } u \in Y(G). \end{cases}$$

Taking into account that in any coloring the vertices in one part (X or Y) of G must be mapped to only one part in H and that H is complete, we have

Lemma 21. *Let (H, C, K) be a partial weight assignment with $H = K_{xy}$ and $C \subseteq Y(H)$. There is a list (H, C, K)-colorings of (G, L) iff: there is a list $H[X(H)]$-coloring of $(G[X(G)], L_1)$ and a list $(H[Y(H)], C, K)$-coloring of $(G[Y(G)], L_1)$, or there is a list $H[X(H)]$-coloring of $(G[Y(G)], L_2)$ and a list $(H[Y(H)], C, K)$-coloring of $(G[X(G)], L_2)$.*

Thus, we arrive to the following algorithm.

```
function Bipartite(H, C, K, G, L)
    begin
        Set L = L|V(GH)
        if
            Basic(H[X(H)], G[X(Ĝ)], L1) ∧ List-basic(H[Y(H)], C, K, Ĝ[Y(Ĝ)], L1)
            then return true
            else
                return Basic(H[Y(H)], Ĝ[Y(Ĝ)], L2) ∧ Find-list-basic(H[X(H)], C, K, Ĝ[X(Ĝ)], L2)
        end if
    end
```

Finally, taking into account the number of vertices in $\widehat{G}$ is at most $k^2 + 2k + (k+1)2^{k+h}$ (taking the maximum of the sizes obtained for each type of component) and the time complexity of the described algorithms, we get

Theorem 9. *Let (H, C, K) be a partial weight assignment, where (H, C, K) is a 1, 2 or 3-component. Given as input a connected graph G and a (H, G)-list L, algorithm List-Coloring-C checks the existence of a list (H, C, K)-coloring of (G, L) in $O(\mathcal{F}(n, k, h, c, s)$ steps, where*

$$\mathcal{F}(n, k, h, c, s) = (h + k)n + 2^{k+h}k(h + k) + 2^k c^k[(k + h)k^2 + (k^2 + (k + 1)2^{k+h})^{\frac{3}{2}} k \log k].$$

G Proof of Lemma 15

Proof. Let χ be a list (H,C,K)-coloring of (G,L). For any $\iota \in [-\rho,\eta]$, define $B_\iota = \{i \mid \chi(V(G_i)) \subseteq V(H_\iota)\}$, the set of indices of connected components of G that are mapped to $V(H_\iota)$. For any $i \in B_\iota$, let W_i be the partial weight assignment defined as $W_i(a) = |\chi^{-1}(a) \cap V(G_i)|$. Let $Z_\iota = \{i \in B_\iota \mid W_i \neq \mathbf{0}\}$. Clearly, $Z = \cup_{\iota\in[-\rho,\eta]} Z_\iota$ is proper.

For the reverse implication, if there is some $A \subseteq [g]$ that is proper, the partition in $\mathcal{F}(A)$ guarantees the existence of a list coloring that fills the weight, as the remaining components can be mapped to some component of H using zero weight, while respecting the list restrictions. So, we have that (G,L) has a list (H,C,K)-coloring.

H Proof of Theorem 8

Proof. We provide a parameter preserving reduction from the k-independent set problem to the (H,C,K)-coloring, when $K = (k,\dots,k)$, so from now on we assume that $K = (k,\dots,k)$. Given a graph G we construct a new graph G', in the following way. For vertex v of C, add to G' a graph G_v, that is a marked copy of G when v is unlooped or a marked copy of K_k^r otherwise. Furthermore, for every edge $(u,v) \in E[C]$ with $u \neq v$ add all the additional edges in $G_u \oplus G_v$.

When G has an independent set of size k, G' has trivially an (H,C,K)-coloring. Now we prove the other implication. Assume that there is an (H,C,K)-coloring φ of G'.

Let $C_1 \subseteq C$ be the set of unlooped vertices. Observe that $\varphi^{-1}(C_1)$ contains only vertices from copies of G. Let S a maximum independent set in $H[C_1]$, let $s = |S|$ and note that $s > 1$. We know that $\varphi^{-1}(S) = ks$ and that $\varphi^{-1}(S)$ is an independent set of G'.

Let $I = \{v \in V(G) | \text{there is a copy of } v \text{ in } \varphi^{-1}(S)\}$, and for $v \in I$ let s_v be the number of copies of v in $\varphi^{-1}(S)$. By construction, if we take a set of vertices from more that s different copies of G the set is not independent in G', because s is the size of the maximum independent set of C_1, and thus at least two of the copies are connected. Therefore, $s_v \leq s$.

We know that $\sum_{v\in I} s_v = ks$ and that $s_v \leq s$, therefore $|I| \geq k$. Therefore G has an independent set of size at least k.