Efficient algorithms for parameterized H-colorings*†

Josep Díaz Maria Serna Dimitrios M. Thilikos

ALBCOM Research Group.
Dept. de Llenguatges i Sistemes Informàtics
Universitat Politècnica de Catalunya,
Campus Nord–Edifici Ω.
E-08034, Barcelona, Spain.
{diaz,mjserna,sedthilk}@lsi.upc.edu

Abstract

We study the fixed parameter tractability of the restrictive H-coloring and the restrictive list H-coloring problems, introduced in [2]. The parameterizations are defined by fixing the number of pre-images of a subset C of the vertices in H through a partial weight assignment (C, K). We define two families of partially weighted graphs: the *simple* and the *plain*. For the class of simple partially weighted graphs, we show the fixed parameter tractability of the list (H, C, K)-coloring problem. For the more general class of plain partially weighted graphs, we prove the fixed parameter tractability of the (H, C, K)-coloring problem.

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 an *H-coloring* of G. The more general version in which a list of allowed colors (vertices of H) is given for each vertex is known as a *list H-coloring*. See [15] for further extensive background on morphisms on graphs and in particular on H-coloring.

For a fixed graph H, the *H-coloring problem* asks whether there exists an H-coloring of the input graph G and the *list H-coloring problem* asks whether there exists a list H-coloring of the input graph G. The complexity of the H-coloring problem was studied in [14], where the following complexity dichotomy was proved: the H-coloring problem belongs to P if H is bipartite or contains a loop, otherwise the problem is NP-Complete. For the list H-coloring problem, there is also a dichotomy: the problem is in P if H is a bi-arc graph, otherwise the problem is NP-Complete [12].

---

*An extended abstract containing some of the results in this paper was presented at the *12th Annual European Symposium on Algorithms* (ESA 2004).

†The work of the first author was partially supported by the Distinció per a la recerca of the Generalitat de Catalunya 2002. The work of the first and third authors was partially supported by the Spanish CICYT project TIN-2004-0795 (GRAMMARS). The work of the second author was partially supported by the Spanish CICYT project TIN-2005-09918-C02-02 (ASCE).

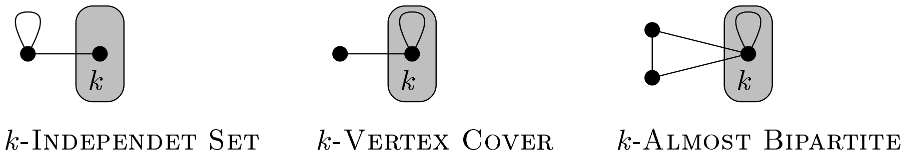

Figure 1: The partially weighted graphs (H, C, K) for the problems k-Independent Set k-Vertex Cover and k-Almost Bipartite.

The case of an H-coloring in which, in addition, the number of pre-images of the vertices in H are restricted, is known as a *restrictive H-coloring* or a *restrictive list H-coloring*. The corresponding decision problems are known as the *restrictive H-coloring* and the *restrictive list H-coloring* problems. The complexity of those problems was studied in [4], where it was proved that the problems are in P if all the connected components of H are either complete reflexive graphs or complete irreflexive bipartite graphs, otherwise the problems are NP-Complete.

To study the structural characteristics that make a problem difficult to solve, a fruitful approach has been to divide the input of a problem into two parts one of them, the *non-parameterized*, remains as part of the input while the other one, the *parameterized*, is fixed independently of the input. This line of research is known as *parameterized complexity*, see for ex. [7]. More specifically, the input to a parameterized problem, is a pair (S, K) where S is the main part and K is the parameterized part. Typically, K is an integer, a sequence of integers, or some other structure, that is fixed independently of the input, and thus it is called the *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]. In the triple (H, C, K), C is the set of vertices in H with restrictions and K is a weight assignment on C defining the number of pre-images of these vertices in G. We will say that (H, C, K) is the *partial weight assignment* corresponding to the parameterized part of the restrictive H-coloring problem.

The considered parameterization allows us to capture some well known parameterized problems as particular cases. In Figure 1 we give three examples where the (H, C, K)-coloring corresponds to the problems, k-Independent Set: does G have an independent set of size k?, k-Vertex Cover: does G have a vertex cover of size k?, and k-Almost Bipartite Graph does G contain a set of k vertices whose removal leaves a bipartite graph?. The shadowed regions represent the set C.

The complexity of the (H, C, K)-coloring problems was studied in [2, 6]. We proved a dichotomy for the list (H, C, K)-coloring problem: the problem is in P if $H - C$ is a bi-arc graph, otherwise it is NP-Complete. For the (H, C, K)-coloring, we proved that the problem is in P if $H - C$ is bipartite or contains a loop, and it is NP-Complete if $H - C$ is not a bi-arc graph, leaving open the dichotomy. In [3], the interested reader can find a survey on results for different parameterization of H-coloring.

In this paper we are interested in finding a large class of (H, C, K) for which the (H, C, K)-coloring problems are *fixed parameter tractable*. Recall that the class FPT of fixed parameter tractable problems is formed by those parameterized problems that can be solved by an algorithm in time $O(f(k)n^c)$ where k is the parameter, n is the input size, and c is a constant independent of k. One of the fundamental tools for designing efficient fixed-parameter algorithms for decision problems is the so called *reduction to problem kernel*. Using this method we give a way to transform G, to a graph $\widehat{G}$, the *kernel*, such that:

- the transformation of G to $\widehat{G}$ can be done in time that is polynomial on the size of G,
- G has an list (H, C, K)-coloring if and only if $\widehat{G}$-has a list (H, C, K)-coloring,
- the size of $\widehat{G}$ depends only on the parameter k.

We isolate two special families, the *simple* and the *plain* partially weighted graphs (see definitions in section 2). For simple (H, C, K) we design linear time fixed parameter algorithms for the list (H, C, K)-problem. For the case of plain (H, C, K) we show that the (H, C, K)-coloring problem is in FPT. Notice that the definition of those families is an extension of the ones considered in [5]. Let us mention that in this paper we also provided several results for the counting version of the (H, C, K)-coloring and the list (H, C, K)-coloring.

The paper is organized as follows: in Section 2 we provide the basic definitions used in the paper.

In Section 3, we present fast exact algorithms for the list (H, C, K)-coloring problem for three basic graph families: (a) when either G is edge-less or H is a complete reflexive; (b) $H - C$ is edge-less; and (c) H is a complete bipartite graph.

In Section 4, we describe how to construct a kernel $\widehat{G}$, for a connected graph G. This construction allows us to solve the list (H, C, K)-coloring problem, in the case that (H, C, K) is plain, by running the exact algorithms obtained in Section 3 on the kernel.

In the case that G is not a connected graph, its connected components may be mapped in different connected components of H and more than one connected component of G can be mapped to the same component of H. To deal with such a case, in Section 5 we define a special structure called *signature*. A signature keeps information on all the ways in which a subset of the connected components of G can be mapped into components of (H, C, K) respecting the list L. Then we construct a special subset D of representatives of the connected components such that the size of D depends only on the parameter. We show that the existence of a non-empty signature for some subset of D is equivalent to the existence of a list (H, C, K)-coloring of G. Thus, checking for this property in linear time allow us to produce an efficient algorithm for the list (H, C, K)-coloring problem.

In Section 6, we present a parameterized reduction from the (H, C, K)-coloring problem for plain (H, C, K) to the list (H, C, K)-coloring for simple (H, C, K). The reduction is done through a series of sub-reductions that gradually transforms a plain (H, C, K) into a simpler (H, C, K).

In Section 7, a series of negative results on the parameterized complexity of the (list) (H, C, K)-coloring problem are presented. We identify some (H, C, K) where the corresponding problems are W[1]-hard. Recall that W[1] is a class of parameterized problems defined in [8, 9] and the W[1]-hardness of a problem makes it quite unexpected that the problem belongs in FPT. In Section 8 we examine several extensions of our results. We modify our algorithms in order to construct a list (H, C, K)-coloring (if exists) and we explain how the existing algorithms should be modified for different versions of the problem. Finally we conclude in Section 9 with a discussion of our results and open problems.

2 Basic definitions and notation

Given a graph G, let $V(G)$ denote its vertex set and $E(G)$ denote its edge set. For $u, v \in E(G)$, $\{u, v\}$ denotes an edge between u and v and $\{u, u\}$ a loop at vertex u. If $U \subseteq V(G)$, $G[U]$

denotes the *subgraph of* G *induced by* U. Following the terminology in [10, 11], 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 $G[V(G)-U]$. Define the neighborhood of a vertex v in G by $N_G(v) = \{u \in V(G) | \{u,v\} \in E(G)\}$. Given two graphs G and G', with no vertices in common, their *disjoint union* is the graph $G \cup G' = (V(G) \cup V(G'), E(G) \cup E(G'))$, and their *join* is the graph $G \oplus G' = (V(G) \cup V(G'), E(G) \cup E(G') \cup \{\{u,v\} \mid u \in V(G), v \in V(G')\}$.

Given a bipartite connected graph G, the vertex set $V(G)$ is splited 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. We assume that the X and Y parts of a connected bipartite graph are defined without ambiguity, for example setting the X part to be the part that contains the first vertex. Let $K_{r,s}$ denote the complete irreflexive bipartite graph on two parts with r and s vertices each. Observe that $K_{r,0}$ denotes a graph G with r vertices a no edges. Finally, K_n^r denotes the reflexive clique with n vertices.

For any function $\varphi : A \to B$ and any subset $A' \subseteq A$, we define the *restriction of* φ *to* A' by $\varphi|_{A'} = \{(a,b) \in \varphi \ \mid a \in D\}$. Given two functions $\varphi : A \to \mathbb{N}$ and $\psi : A' \to \mathbb{N}$, with $A' \cap A = \emptyset$, the *disjoint union* of φ and ψ, is a function from $(\varphi \cup \psi) : A \cup A' \to \mathbb{N}$ such that $(\varphi \cup \psi)(x) = \varphi(x)$ if $x \in A$, and $(\varphi \cup \psi)(x) = \psi(x)$ if $x \in A'$.

Given two functions $\varphi, \psi : A \to \mathbb{N}$, for any $x \in A$, we define $(\varphi + \psi)(x) = \varphi(x) + \psi(x)$. We say that $\phi \leq \psi$ if, for any $x \in A$, $\phi(x) \leq \psi(x)$. We say that $\varphi : A \to \mathbb{N}$ is positive if, for any $a \in A$, we have $\varphi(A) > 0$. We denote by $\varnothing$ the empty function.

We often need sets of consecutive indices. For $n, m \in \mathbb{N}$, we use the notation $[n] = \{1, \ldots, n\}$ and $[-m, n] = \{-m, \ldots, -1, 0, 1, \ldots, n\}$.

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

For a fixed graph H, given a graph G, an *(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 $L(v) \subseteq V(H)$. Given an (H,G)-list L, a *list H-coloring* of (G,L) (or a list H-coloring for short) is an H-coloring χ of G such that, for every $u \in V(G)$, $\chi(u) \in L(u)$.

A *partial weight assignment* on H is a pair (C,K), where $C \subseteq V(H)$ and $K : C \to \mathbb{N}$. We refer to the vertices of C as the *weighted vertices*. Given a partial weight assignment (C,K) on H, a mapping $\chi : V(G) \to V(H)$ is a *restrictive H-coloring* of G if χ is an H-coloring of G such that for all $a \in C$, we have $|\chi^{-1}(a)| = K(a)$.

For a fixed graph H, given an input graph G, the *H-coloring* problem asks whether there exists an H-coloring of G. In the same way, for a given graph G and an (H,G)-list L, the list *H-coloring* asks whether there exists a list H-coloring of (G,L). If in addition a partial weight assignment (C,K) is given and we ask for a restrictive or a list restrictive H-coloring, we get the *restrictive H-coloring* and the *restrictive list H-coloring* problems.

We consider the parameterization of the restrictive H-coloring problems in which the partial weight assignment (C,K) on H is selected as parameter. In order to simplify notation, we represent by a triple (H,C,K) the target graph and the partial weight assignment. We also extend the term restrictive to the set of colorings: An (H,C,K)-coloring of G is a restrictive H-coloring of G and (C,K), while a list (H,C,K)-coloring of G is a restrictive list H-coloring of G, an (H,G)-list L, and (C,K).

For a fixed (H,C,K), given an input graph G, the *(H,C,K)-coloring* problem asks whether there exists an (H,C,K)-coloring of G. The *list (H,C,K)-coloring* problem asks whether there exists an (H,C,K)-coloring of (G,L), when G and an (H,G)-list L are given as input.

Notice H-coloring and (H,C,K)-colorings behave differently for non connected graphs. An H-coloring of G is obtained as the union of the H-colorings of the connected components of G. However, the union of (H,C,K)-colorings for the connected components of G might not be an (H,C,K)-coloring of G. This is due to the fact that the total number of pre-images of the vertices in H might exceed the corresponding bound.

A partially weighted graph (H,C,K) is *positive* if K is positive. We say that (H,C,K) and (H',C',K') are *equivalent*, and we denote it as $(H,C,K) \sim (H',C',K')$, if, for any graph G, G has an (H,C,K)-coloring if and only if G has an (H',C',K')-coloring.

Next, we introduce some types of partially weighted graphs. An illustration of these types is given in Figure 2.

A partially weighted graph (H,C,K) with $C \neq \emptyset$, is a

- *1-component* if $E(H-C) = \emptyset$.
- *2-component* if H is a reflexive clique.
- *3-component* if H is a complete bipartite graph.
- *4-component* if $H[C]$ is a reflexive clique with all its vertices adjacent with one looped vertex x of $V(H) - C$.
- *6-component* if H is bipartite and
  - (1) if $C \cap X(H) \neq \emptyset$ and $C \cap Y(H) \neq \emptyset$, there are vertices $x \in X(H) - C$, $y \in Y(H) - C$, $a \in C \cap X(H)$, and $b \in C \cap X(H)$, such that $(x,y) \in E(H)$, x is adjacent to all the vertices in $C \cap Y(H)$, y is adjacent to all the vertices in $C \cap x(H)$, and $\{a,b\} \in E(H)$, or
  - (2) if $C \cap X(H) \neq \emptyset$ but $C \cap Y(H) = \emptyset$, there are vertices $x \in X(H) - C$, $y \in Y(H)$, and $a \in C \cap X(H)$ such that $(x,y) \in E(H)$, and y is adjacent to all the vertices in $C \cap X(H)$, or
  - (3) if $C \cap X(H) \emptyset$ and $C \cap Y(H) \neq \emptyset$, there are vertices $x \in X(H) - C$, $y \in Y(H) - C$, and $b \in C \cap X(H)$, such that $(x,y) \in E(H)$ and x is adjacent to all the vertices in $C \cap Y(H)$.

Notice that the cases of 2 and 3-components are the only ones in which the graph H is required to be connected. Furthermore the connected components of any other type of component must be of the same type.

A partially weighted graph (H,C,K) with $C = \emptyset$, is a

- *1-component* when H is edge-less,
- *2-component* when H is a reflexive clique,
- *3-component* when H is a complete bipartite graph,
- *4-component* when H has a looped vertex,
- *6-component* when H is bipartite.

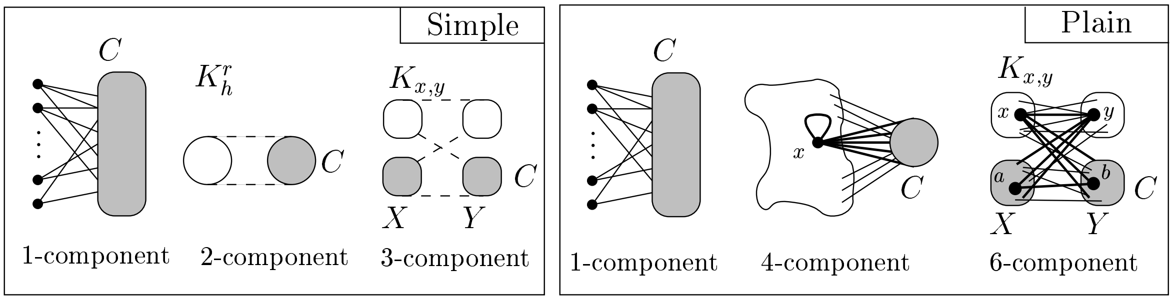

Figure 2: Components for simple and plain (H, C, K).

(H, C, K) is said to be *simple* when, for each connected component H' of H, $(H', C \cap H', K|_{V(H')})$ is either a 1-component, a 2-component, or a 3-component.

(H, C, K) is said to be *plain* when, for each connected component H' of H, $(H', C \cap H', K|_{V(H')})$ is either a 1-component, a 4-component, or a 6-component.

We assume that (H, C, K) is pre-processed once in our algorithms. This preprocessing consists on the computation of the connected components of H, and their classificartion according to the above types. The overall additional computation can be performed in $O(h^2)$ steps. In Sections 3, 4, and 5, we give algorithms for the list (H, C, K)-problem assuming that the input G is a graph without loops. This does not harm the generality of our results as looped vertices of G should be mapped to looped vertices of H. Therefore, an equivalent instance can be constructed if, for any looped vertex v in G, we replace $L(v)$ by $L(v) \cap \{a \mid a \text{ is a looped vertex in } H\}$, thus removing all loops in G.

All through the paper, for a given (H, C, K), we use the notation $S = V(H) - C$, $h = |V(H)|$, $c = |C|$, $s = |S|$ and $k = \sum_{c \in C} K(c)$.

3 Exact algorithms for list (H, C, K)-coloring in particular cases

The fixed parameter algorithms for deciding the existence of a list (H, C, K)-coloring problem, when (H, C, K) is a 1, 2 or 3-component, are based on the kernelization technique. We first construct a kernel, and then we solve the (H, C, K)-coloring problem on it. In this section, we devise fast algorithms for particular cases of H, G and $H - C$.

3.1 G is edge-less or H is a reflexive clique

For the first two cases ($E(G) = \emptyset$ or $H = K_h^r$) we consider the following algorithm:

```
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
```

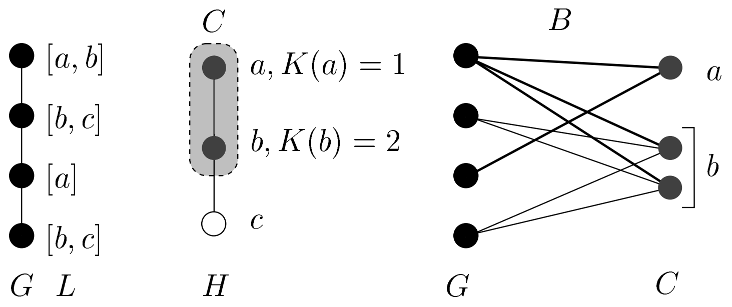

Figure 3: A partially weighted graph (H, C, K), a graph G, a (G, H)-list, and the associated weighted bipartite graph $B = B(H, C, K, G, L)$. In B, the "fat" edges have weight 3 and the rest have weight 1.

Theorem 3.1. 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 correctly decides in $O(hn)$ steps whether a list H-coloring of (G, L) exists.

Proof. 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, given a (H, G)-list L, there is a list H-coloring of (G, L) if and only if for all $v \in V(G)$, $L(v) \neq \emptyset$. □

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 partially weighted graph, and let L be an (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,
- $Y(B) = \bigcup_{a \in C} A(a)$,
- 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 usually refer to $X(B)$ as the G-part and to $Y(B)$ as the C-part of B. In Figure 3 we give an example of the construction, notice that the graph B depends on the structure of L but not on the edges present in either G or H.

Lemma 3.2. Let (H, C, K) be a partially weighted graph, G a graph, and L an (H, G)-list. The weighted bipartite graph $B = B(H, C, K, G, L)$ 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 by 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 3.3. Let (H, C, K) be a partially weighted graph, G a graph, and L an (H, G)-list and let $B = B(H, C, K, G, L)$ be the associated bipartite graph. Assume that $E(G) = \emptyset$ or that $H = K_h^r$. Let $p = |\{v \in V(G) \mid L(v) \subseteq C\}|$. 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) if and only if the weight of a maximum matching in B is $kp + k - p$.

Proof. Note that the maximum number of vertices in G that can be mapped to C is k. Furthermore, all the vertices $v \in V(G)$ with $L(v) \subseteq C$ must be mapped to C. Therefore, if $p > k$, (G, L) does not have a list (H, C, K)-coloring.

Assume now that $p \leq k$ and let us prove the second part of the lemma.

Assume that χ is a list (H, C, K)-coloring of (G, L). Let us construct a maximum weight matching for B. For every $a \in C$, assign the vertices in $\chi^{-1}(a)$ to the different copies of a in B. This assignment provides a matching with weight $pk + k - p$. Observe that, by construction, $pk + k - p$ is the maximum weight of any selection of k vertices in the G-part of B, as the number of edges with weight k is maximized. Therefore, the constructed matching has maximum weight.

For the reverse implication, note that, by definition of B, in any maximum weight matching M of B with $w(M) = kp + k - p$, all the vertices in the C-part are matched. Furthermore, all the number of edges with weight k in the matching is maximum and that all the vertices in G that must be mapped to C appear matched. Therefore, the matching M provides an assignment of vertices of G to vertices of $Y(B)$. Composing this mapping with the function φ_a, the function that assigns value a to any $y \in A(a)$, we get a valid (H, C, K)-coloring of (G, L). □

Consider the following algorithm:

function List-basic(H, C, K, G, L)
  begin
    $p = 0$
    for all $v \in V(G)$ with $L(v) \cap S = \emptyset$ **do**
      $p = p + 1$
      if $p > k$ **then return** $\varnothing$ **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**
    $V' = \{v \in V(G) \mid v \text{ is not matched in } M\}$
    $G' = (V', \emptyset)$
    forall $v \in V'$ **set** $L'(v) = L(v) \cap S$ **end for**
    return Basic$(H[S], G', L')$
  end

Theorem 3.4. Let (H, C, K) be a partially weighted graph, let G be a graph, and let L be a (H, G)-list. When G has no edges or $H = K_h^r$, function 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 3.3. From Lemma 3.2 we have that computing the associated bipartite graph takes $O((k + h)n)$ time. This bound dominates all the remaining steps except from computing the maximum weighted matching in B. We use the Gabow-Tarjan algorithm in [13] 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. As B has at most nk edges and $n+k$ vertices, the claimed bound follows. □

3.2 $H-C$ is edge-less

We use the self-reducibility of the list H-coloring problem to design an algorithm for the list (H,C,K)-coloring when $E(H-C)=\emptyset$. Recall that the self-reducibility can be used in the following well known way: Consider a graph G, a partially weighted graph (H,C,K), an (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 $\chi(v)=a$ if and only if (G',L') has a list (H,C,K')-coloring.

The idea behind the algorithm is to reduce the time computation by using a truncated search tree. Each edge in G must have at least one of its endpoints mapped to C, the algorithm tries both possibilities recursively. Each recursive call removes one vertex v that has already being mapped to some $c\in C$, modifies the lists of v's neighbors to guarantee that the recursive call is correct (according to self-reducibility), and decreases $K(c)$ by one. The recursion finishes either when the remaining weight is zero or when G looses all its edges.

function Self-red(H,C,K,G,L)
    begin
        if $L(v)=\emptyset$ for some $v\in V(G)$ **then return false end if**
        if $K=\mathbf{0}\wedge E(G)\neq\emptyset$ **then return false end if**
        if $K=\mathbf{0}\wedge E(G)=\emptyset$ **then return** Basic$(H[S],G,L)$ **end if**
        if $E(G)=\emptyset$ **then return** List-basic(H,C,K,G,L) **end if**
        Select an edge $e\in E(G)$
        for all endpoints v of e **do**
            $G'=G-\{v\}$
            for all $a\in C$ **with** $K(a)>0$ **do**
                for all $b\in C-\{a\}$ **set** $K'(b)=K(b)$ **end for**
                $K'(a)=K(a)-1$
                for all $u\in V(G')$ **set** $L'(u)=L(u)$ **end for**
                for all $u\in N_G(v)$ **set** $L'(u)=L'(u)\cap N_H(a)$ **end for**
                if Self-red(H,C,K',G',L') **then return true end if**
            end for
        end for
        return false
    end

Theorem 3.5. Let (H,C,K) be a partially weighted graph, where $H-C$ is edge-less. Given an input graph G, function Self-red decides whether there is a list (H,C,K)-coloring of (G,L) in $O(2^kc^k((k+h)n+nk\log k\sqrt{n+k}))$ steps.

Proof. The correctness of Self-red follows from the self-reducibility defined at the begining of this subsections, and from the fact 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, the case $L(v)=\emptyset$ or the case when $K=\mathbf{0}$ and $E(G)\neq\emptyset$. In both cases there are no colorings of (G',L'). In the other two base cases $K=\mathbf{0}$ or $E(G)=\emptyset$, we are in one of the basic cases that can be solved according to Corollaries 3.1 and 3.4 using functions Basic or List-basic respectively.

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 3.4, the claimed bound follows. □

3.3 H is a complete bipartite graph and G is connected

To complete the picture, we give the steps needed to solve the list (H,C,K)-coloring for the case when H is a complete bipartite graph and G is connected.

Consider the following algorithm.

function Bipartite(H,C,K,G,L)
  begin
    for all $u\in X(G)$ **do**
      Set $L_1(u)=L(u)\cap X(H)$
      Set $L_2(u)=L(u)\cap Y(H)$
    end for
    for all $u\in Y(G)$ **do**
      Set $L_1(u)=L(u)\cap Y(H)$
      Set $L_2(u)=L(u)\cap X(H)$
    end for
    if (List-basic$(H[X(H)],C,K,G[X(G)],L_1)$ $\wedge$ List-basic$(H[Y(H)],C,K,G[Y(G)],L_1)$
      then return true
      else
        return (List-basic$(H[Y(H)],G[Y(G)],L_2)$ $\wedge$ List-basic$(H[X(H)],C,K,G[X(G)],L_2)$)

      end

Theorem 3.6. Let (H,C,K) be a partially weighted graph where $H=K_{xy}$. Given G a connected bipartite graph and an (H,G)-list L, function Bipartite 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. 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 $H = K_{xy}$, we have that there is a list (H, C, K)-colorings of (G, L) if and only if there is a list $(H[X(H)], C, K)$-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 $(H[X(H)], C, K)$-coloring of $(G[Y(G)], L_2)$ and a list $(H[Y(H)], C, K)$-coloring of $(G[X(G)], L_2)$. Thus the algorithm is correct.

The time bound follows from Corollaries 3.1 and 3.4. $\square$

4 The list (H, C, K)-coloring problem for connected graphs

We start presenting a generic algorithm, called List-Coloring, for the case where G is a connected graph. In the following subsections we show the adjustments needed for the different types of components.

Our first step is to introduce an equivalence relation on the vertices of a connected graph G. This equivalence will be used as the main tool to construct the kernels. Assume that (H, C, K), a graph G, and an (H, G)-list L are given. We 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 $v \in V(G)$, P_v denotes the set $\{u \mid u \sim v\}$. We say that $R \subseteq V(G)$ is a *closed set* of the partition $\mathcal{P}$, if for any $v \in R$ we have $P_v \subseteq R$.

For $v \in V(G)$ let P_v^k be P_v if $|P_v| \leq k$, otherwise P_v^k is a subset of P_v with $k+1$ elements. Let R be a closed set, the *graph* $\widehat{G}$ *associated to* (G, R) is the graph induced by all the vertices in R, and for the classes P_v with $v \notin R$, all the vertices in P_v^k.

Note, that to ensure that $\widehat{G}$ has bounded size it is required that $\mathcal{P}$ has a small number of classes. Furthermore, $V(\widehat{G})$ keeps all the vertices of some classes with few vertices (the set R) and a restricted number of representatives for the remaining classes.

Next lemma shows the property which suffices for showing that the graph associated to a closed set is an adequate kernel, provided that its size is small.

Lemma 4.1. Let (H, C, K) be a partially weighted graph. Given a connected graph G together with an (H, G)-list L and a closed set R, let $\widehat{G}$ be the graph associated to (G, R). Then, (G, L) has a list (H, C, K)-coloring if and only if $(\widehat{G}, L)$ has a list (H, C, K)-coloring.

Proof. Let w be an (H, C, K)-coloring of $(\widehat{G}, L)$. We extend the coloring w to a coloring χ of G as follows: For those vertices $u \in V(G)$, we set $\chi(u) = w(u)$. For those vertices $u \in V(G) \setminus V(\widehat{G})$, we know that $|P_u| > k+1$, therefore there are exactly $k+1$ vertices in $V(\widehat{G})$ from P_u, which implies that there exists $v \in P_u$ for which $w(v) \in S$. Set $\chi(u) = w(v)$. The definition of $\mathcal{P}$ ensures that the obtained coloring is an (H, C, K)-coloring of (G, L).

To prove the reverse implication, let χ be a (H, C, K)-coloring of (G, L), we construct a coloring w of $\widehat{G}$. Notice that for any equivalence class Q with $|Q| > k+1$, we have that $|\chi(Q) \cap S| > 1$, therefore one vertex $a_Q \in \chi(Q) \cap S$ can be selected. For any $d \in C$ let $d_Q = |\chi^{-1}(c) \cap Q|$. Define $w : V(\widehat{G}) \to V(H)$ as follows: For $u \in R$, set $w(u) = \chi(u)$; otherwise, if $u \notin R$ but $|P_u| \leq k+1$, set $w(u) = \chi(u)$. To each representative of a class Q with $|Q| > k+1$, assign in a round robin way, vertex $d \in C$ to d_Q vertices, and complete the coloring assigning a_Q to the remaining vertices in $Q \cap V(\widehat{G})$. Again the definition of $\mathcal{P}$ guarantees that w is a (H, C, K)-coloring of $(\widehat{G}, L)$. $\square$

The generic algorithm for deciding the existence of a list (H, C, K)-coloring problem uses $\widehat{G}$ as kernel:

function List-Coloring(H, C, K, G, L)
  input A partially weighted graph (H, C, K).
    A connected graph G and a (H, G)-list L.
  begin
    $k = \sum_{a\in C} K(a)$; $s = |V(H) - C|$;
    $R =$ Closed-set(G, k)
    if $R[0] = 0$ **then return** `NO` **end if**
    $\widehat{G} =$ Kernel$(G, R, k+1)$
    return Exact$(H, C, K, \widehat{G}, L)$
  end

List-Coloring(H, C, K, G, L) computes first the adequate closed set, defined in a different way for each type of component. The adequate definition of R and the definition of $\mathcal{P}$ will guarantee that $\widehat{G}$ has bounded size. In a second step, the algorithm checks whether there is a list (H, C, K)-coloring of $(\widehat{G}, L)$, using the exact algorithms presented in the previous section. The central **if** detects whether the non-existence of list (H, C, K)-colorings of G has been observed while computing the corresponding closed set.

In the following subsections we give the definition of the closed-set that guarantees that $\widehat{G}$ is a kernel for each type of component and fix the remaining components for the analysis of the adaptation of algorithm List-Coloring.

4.1 The case of 1-components

To obtain the adequate kernelization, we start considering a classical kernelization for vertex cover due to Buss and Goldsmith [1].

Given a graph G and an integer k, the *k-splitting* of G is the 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 at least 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 4.2. Let (H, C, K) be a partially weighted graph, where $H - C$ is edge-less. Given a connected graph G, let (R_1, R_2, R_3) be the k-splitting of G. Then $R_1 \cup R_2$ is a closed set. Furthermore, if $|R_1| > k$ or $|R_2| > k^2 + k$, then there are no (H, C, K)-colorings of (G, L).

Proof. Notice that C is a vertex cover in H. Therefore, for any (H, C, K)-coloring χ of G, $\chi^{-1}(C)$ must be a vertex cover in G of size $k = \sum_{i\in C} k_i$. Therefore, for every $v \in \chi^{-1}(V(H) - C)$ $d_G(v) \leq k$. In any (H, C, K)-coloring of G, if there exists a $v \in G$ such that $d_G(v) > k$, v must be mapped to C, and thus $|R_1| \leq k$.

For the reversed implication, we have that in any (H, C, K)-coloring χ of G, $R_1 \subseteq \chi^{-1}(C)$. Furthermore, for all $v \notin R_1$, $d_{G(v)} \leq k$, so at most k vertices outside of R_1 can be mapped to C. The remaining vertices must be connected to vertices in $\chi^{-1}(C)$, which implies $\chi^{-1}(C) \cup N_G(\chi^{-1}(C)) = R_2$. Therefore, for all $v \in V(G')$, $d_{G'}(v) \leq k$ and thus $|R_2| \leq k^2 + k$. □

Lemma 4.3. Let (H, C, K) be a partially weighted graph, where $H - C$ is edge-less. Given a connected graph G and an (H, G)-list, let (R_1, R_2, R_3) be the k-splitting of G and let $\widehat{G}$ be the graph associated to $(G, R_1 \cup R_2)$. If $|R_1| \leq k$ and $|R_2| \leq k^2 + k$, then $\widehat{G}$ has $k^2 + 2k + (k+1)2^k$ vertices. Furthermore, $\widehat{G}$ can be obtained in time $O((k+h)n + 2^{k+h})$.

Proof. The following algorithm constructs the closed set associated to the k-splitting of G.

function Closed-set(G, k)
  begin
    $\nu = 0$
    for all $v \in V(G)$ **do**
      if $d_G(v) < k$ **then** $R[v] = 1; \nu = \nu + 1$ **else** $R[v] = 2$ **end if**
    end for
    if $\nu > k$ **then** $R[0] = 0;$ **return** R **else** $R[0] = \nu; \nu = n - \nu$ **end if**
    for all v with $R[v] = 2 \wedge N_G(v) \subseteq R_1$ **do**
      $R[v] = 3; \nu = \nu - 1$
    end for
    if $\nu > k^2 + k$ **then** $R[0] = 0$ **end if**
    return R
  end

From the above description, taking into account that G is represented by an array of linked list of sorted neighbors, we have that function Closed-set finishes in $O(kn)$ steps. Furthermore, if $|R_1| \leq k$ and $|R_2| \leq k^2 + k$, it computes the closed set associated to the k-splitting.

Notice that, by construction, the vertices in R_3 are connected only to vertices in R_1. To obtain $\widehat{G}$ we select a set of vertices U from R_3. The set U is initially empty. We use an auxiliary rooted tree T whose nodes hold a counter. Each node in the tree identifies a pair formed by a subset of R_1, defined by a sorted list of vertices, and a subset of $V(H)$, identified also by a sorted list. Notice that such a tree has size bounded by 2^{k+h}.

We process the vertices in R one after the other. Let u be a vertex in R_3. We check whether the counter associated to the pair $(N_G(u), L(u))$ has not reached $k + 1$. If so we increase the counter by one and keep u in U, otherwise we discard u.

Once the set U is computed, we set $\widehat{G} = G[U \cup R]$. Notice that it takes $k + h$ steps to identify the associated pair. Furthermore, each vertex in $\widehat{G}$ has degree at most k thus $O(kn)$ additional steps are needed to construct the kernel. By the results obtained in the previous lemma, the total time is $O((k + h)n + 2^{k+h})$. □

Finally, we put together the two pieces to obtain an algorithm for the case of 1-components.

Theorem 4.4. Let (H, C, K) be a partially weighted graph, where $E(H - C) = \emptyset$. Given an input graph G and a (H, G)-list L, there is an algorithm that decides whether there is a list (H, C, K)-coloring of (G, L) in time

$$O((h + k)n + 2^{k+h} + 2^k c^k((k + h)(k^2 + k2^k) + (k^2 + k2^k)^{\frac{3}{2}} k^2)).$$

Proof. The correctness of the implementation of algorithm List-coloring follows from Lemmata 4.1, 4.2, and 4.3 together with Theorem 3.5.

Recall that, according to Lemma 4.3, $|V(\widehat{G})| \leq k^2 + 2k + (k + 1)2^k = O(k^2 + k2^k)$ and constructing $\widehat{G}$ takes $O((h + k)n + 2^{k+h}$ steps. The correct exact algorithm to solve the kernel problem is the function Self-red, described in page 9. This algorithm requires $O(2^k c^k((k + h)|V(\widehat{G})| + |V(\widehat{G})| \cdot k \log k \sqrt{|V(\widehat{G})| + k}))$ steps. □

In Theorem 4.4, we may assume that both G and H are not necessarily connected. We can do this as we never required H to be connected in this subsection. This observation will be useful in Section 5 where we consider all connected 1-components of H as a unique unified 1-component.

4.2 The kernel for 2-components

We start with an easy and trivial observation: for any partially weighted graph (H, C, K) with $H = K_h^r$, there exists a list (H, C, K)-coloring of (G, L) if an only if there is a list (H, C, K)-coloring of (G', L), where $G' = (V(G), \emptyset)$. Without loss of generality, in the remaining of this section, we assume that $E(G) = \emptyset$. This fact simplifies the structure of the equivalence classes in relation $\mathcal{P}$. Notice that the only existing neighborhood is the empty set. Therefore, we have to take into account only the list associated to each vertex.

Given a graph G together with an (H, G)-list L, define a *list splitting* of (G, L) as a partition of $V(G)$ in two sets, (N_1, N_2), where $N_1 = \{v \in V(G) \mid L(v) \subseteq C\}$.

The following condition follows from the previous definition.

Lemma 4.5. 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 N_1 is a closed set. Furthermore, if $|N_1| > k$, there are no list (H, C, K)-colorings of (G, L).

Our next result shows that we can use the graph associated to N_1 as kernel.

Lemma 4.6. Let (H, C, K) be a partially weighted graph with $H = K_h^r$. Given a graph G and a (H, G)-list L, let (N_1, N_2) be the list-splitting of (G, L), and let $\widehat{G}$ be the graph associated to (G, N_1). If $|N_1| \le k$, then $\widehat{G}$ has at most $k + (k + 1)2^h$ vertices. Furthermore, $\widehat{G}$ can be obtained in time $O(hn + kh2^h)$.

Proof. In a way similar to the one used in the case of 1-components, the closed set associated to the list-splitting can be computed, using the following $O(hn)$-algorithm:

```
function Closed-set(G, k)
    begin
        ν = 0
        for all v ∈ V(G) do
            if L(v) ⊆ C then R[v] = 1; ν = ν + 1 else R[v] = 0 end if
        end for
        if ν > k then R[0] = 0; return R else R[0] = 1 end if
    end
```

Note that the case $|N_1| > k$ is recorded by setting $R[0] = 0$.

As G has no edges, the number of equivalence classes is at most 2^h. Thereforem, each class can be identified by a string of length h representing a subset of H. Keeping an array we can process all the vertices in N_2 in a way similar to that used in the proof of Lemma 4.3. For every vertex $v \in N_2$, we check whether the entry associated to $L(v)$ has reached the value $k+1$. If so we discard the vertex, otherwise we keep the vertex in U and increment the counter. Finally, $\widehat{G}$ is the graph induced by $N_1 \cup U$.

The number of steps needed per vertex is at most h, therefore the overall computation of the kernel takes time $O(hn + h(k + (k + 1)2^h))$ steps. ☐

Putting together the previous results, following the schema given by algorithm List-coloring, and taking into account that the algorithm List-basic, given in Page 8 solves the kernel problem, from Lemmata 4.1, 4.5, and 4.6, and Theorem 3.4 we have

Theorem 4.7. Let (H, C, K) be a partially weighted graph with $H = K_h^r$ and let G be a connected graph, there is an algorithm that decides whether there is a list (H, C, K)-coloring of (G, L) in time

$$O(hn + k^{5/2} \log k 2^{3h/2})$$

4.3 The kernel for 3-components

Observe that for any partially weighted graph (H, C, K) with $H = K_{xy}$, and given a (H, G)-list L, (G, L) has a list (H, C, K)-coloring if and only if (G', L) has a list (H, C, K)-coloring, where $G' = (X(G), Y(G), X(G) \times Y(G))$. Therefore we may assume that G is a complete bipartite graph.

Let (H, C, K) be a partially weighted graph with $H = K_{xy}$. 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 $M_3 = \{v \in V(G) \mid L(v) \not\subseteq C\}$, $M_1 = X(G) - M_3$ and $M_2 = Y(G) - M_3$.

Note that all the vertices in M_1 and M_2 must be mapped into C. Therefore, we obtain the following result,

Lemma 4.8. Let (H, C, K) be a partially weighted graph with $H = K_{xy}$. Given a complete bipartite graph G, let (M_1, M_2, M_3) be the bipartite splitting of (G, L). Then, $M_1 \cup M_2$ is a closed set. Furthermore, if $|M_1| + |M_2| > k$, there are no list (H, C, K)-colorings of (G, L).

The next result gives the kernel conditions for 3-components.

Lemma 4.9. Let (H, C, K) be a partially weighted graph with $H = K_{xy}$. Given a connected bipartite graph G and an (H, G)-list L, let $M = (M_1, M_2, M_3)$ be the bipartite splitting of (G, L) and let $\widehat{G}$ be the graph associated to $(G, M_1 \cup M_2)$. If $|M_1| + |M_2| \leq k$, $\widehat{G}$ has at most $k + (k + 1)2^{h+1}$ vertices. Furthermore, $\widehat{G}$ can be obtained in time $O(hn + kh2^h)$.

Proof. Working in a way similar to that of the case of 1 and 2-components, we devise first an algorithm for computing the associated closed set. The following function constructs the bipartite splitting of (G, L). At the end of its execution $R[0]$ will hold a 0 when $|M_1| + |M_2| > k$; and a 1 otherwise.

function Closed-set(G, k)
  begin
    $\nu_1 = 0$
    for all $v \in X(G)$ **do**
      if $L(v) \subseteq C$ **then** $R[v] = 1; \nu_1 = \nu_1 + 1$ **else** $R[v] = 3$ **end if**
    end for
    $\nu_2 = 0;$
    for all $v \in Y(G)$ **do**
      if $L(v) \subseteq C$ **then** $R[v] = 2; \nu_1 = \nu_1 + 1$ **else** $R[v] = 3$ **end if**
    end for
    if $\nu_1 + \nu_2 > k$
      then $R[0] = 0$

else $R[0] = 1$ **end if**
end if
return R
end

As G is a complete bipartite graph, the number of equivalence classes is at most 2^{h+1}. Furthermore, each class can be identified by a string of length h representing a subset of H and a letter x or y representing whether the neighborhood of the vertex is $X(G)$ or $Y(G)$. Using an array as data structure, we can process all the vertices that do not belong to the closed set, in a way similar to the one used in the proof of Lemma 4.3. For every vertex v, we check whether the entry associated to $L(v)$ and their neighborhood have reached the value $k+1$. In the positive case, we discard the vertex, otherwise we keep the vertex in a set U and increment the counter. Finally, $\widehat{G}$ is the graph induced by $M_1 \cup M_2 \cup U$.

The number of steps needed per vertex is at most $h+1$, therefore the overall computation of the kernel takes $O(hn + hk2^h)$ steps. □

Putting together the previous results, following the schema given by algorithm List-coloring, and taking into account that algorithm Bipartite solves the kernel problem, from Lemmata 4.1, 4.8, and 4.9, together with Theorem 3.6 we have

Theorem 4.10. Let (H, C, K) be a partially weighted graph with $H = K_{xy}$ and let G be a connected bipartite graph. There is an algorithm that decides whether exists a list (H, C, K)-coloring of (G, L), which works in time

$$O(hn + hk2^h + k^{5/2} \log k 2^{3h/2})$$

5 The list (H, C, K)-coloring problem for simple partially weighted graphs

When dealing with simple partially weighted graphs, we have to take care of the connected components of G. The key idea 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 filled by a legal list coloring of a component G_i of G. Furthermore, as H has bounded size, we can classify the components of G and maintain a small set of representatives.

Our purpose is to define a *signature* associated to subsets of connected components of G, formally subsets of the set $[g]$. Assume that the number of 3-components in H is ρ, and the number of 2-components is η. A 3-component H_ι of H has a negative index ($\iota \in [-\rho, -1]$) and a 2-component H_ι has a positive index ($\iota \in [1, \eta]$). The union of the remaining 1-components of H is considered as a unique 1-component H_0 (not necessarily connected).

Before defining formally signatures, let us introduce some additional notation. For any $\iota \in [-\rho, \eta]$, define $\mathcal{W}_\iota = \{W : C \to \mathbb{N} \mid \mathbf{0} \leq W \leq K_\iota\}$ where $K_\iota = K|_{V(H_\iota)}$. Let $k^*_\iota = \prod_{a \in C_\iota} K_\iota(a)$ and $k^* = \prod_{a \in C} K(a)$. Observe that $k^* = \prod_{\iota \in [-\rho, \eta]} k^*_\iota$ and that $|\mathcal{W}_\iota| = k^*_\iota$. Finally, define $\mathcal{W}^+_\iota = \mathcal{W}_\iota - \{\mathbf{0}\}$.

Our next definition is 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 or not a list-(H_ι, C_ι, W)-coloring of $(\widehat{G}_{i\iota}, L_{\iota i})$.

We isolate the information that must be captured from a set of components to certify the existence of a list coloring. Given $A \subseteq [g]$, and $\iota \in [-\rho, \eta]$ define:

$$\mathcal{F}(A, \iota) = \{(Z, f) \mid Z \subseteq A, |Z| \leq k_\iota, f : Z \to \mathcal{W}_\iota^+, \\ \sum_{i \in Z} f(i) = K_\iota \wedge \forall i \in Z\ \alpha(i, \iota, f(i)) = 1\}$$

For a set $A \subseteq [g]$ a *signature* is a tuple $(Z_{-\rho}, f_{-\rho}), \ldots, (Z_\eta, f_\eta)$, where $Z_{-\rho} \ldots Z_\eta$ form a non-trivial partition of A and for all $\iota \in [-\rho, \eta]$ and $f_\iota : Z_\iota \to \mathcal{W}_\iota$, we have $(Z_\iota, f_\iota) \in \mathcal{F}(A, \iota)$. Denote by $\mathcal{S}(A)$ the set of signatures of A. Notice A may be empty. We say that $A \subseteq [g]$ is *proper with respect to* G if $\mathcal{S}(A) \neq \emptyset$ and, for any $i \in [g] - A$, there is $\iota \in [-\rho, \eta]$ such that $\alpha(i, \iota, \mathbf{0}) = 1$.

The next step is to establish an equivalence relation to reduce the number of candidates in $[g]$ to form a proper set:

$$i \sim j \text{ iff } \ \forall\, \iota \in [-\rho, \eta]\ \forall\, W \in \mathcal{W}_\iota\ \ \alpha(i, \iota, W) = \alpha(j, \iota, W). \tag{1}$$

Let $\mathcal{P}$ be the partition of $[g]$ defined by the above equivalence relation. Notice that we have at most 2^{k^*} equivalence classes and that each class can be represented by a binary string of length k^*. Consider the set $D \subseteq [g]$ obtained by keeping $k+1$ representatives from each class with more than k members, otherwise take all the elements in the class. Notice that F corresponds to a set of components of G whose cardinality *does not depend* on n. Set $\widehat{G} = \bigcup_{i \in F} G_i$.

Lemma 5.1. Let (H, C, K) be a simple partially weighted graph. Given a graph G together with a (H, G)-list L. Then, G has a list (H, C, K)-coloring if and only if there is some $A \subseteq D$ that is proper with respect to $\widehat{G}$.

Proof. Let χ be a list (H, C, K)-coloring of (G, L). For any $\iota \in [-\rho, \eta]$, let $B_\iota = \{i \mid \chi(V(G_i)) \subseteq V(H_\iota)\}$ be 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 partially weighted graph 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 of the connected components of G considered in one of the signatures in $\mathcal{S}(A)$ guarantees the existence of a list coloring that fills the weight. Note that 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. $\square$

It follows from Lemma 5.1, that in order to check whether G has a list (H, C, K)-coloring, it suffices to compute the set D and check whether it contains a subset A that is proper with respect to $\widehat{G}$. Notice that this last check depends only on k and h and therefore it requires $O(f_2(k, h))$ steps. In order to compute D, we have to compute first the function α. If α is given, we can build the partition $\mathcal{P}$ of $[g]$ by applying g times the check in (1), which depends only on k and h. Therefore, the construction of D and $\widehat{G}$ can be done in time $O(g \cdot f_1^0(k, h))$, provided that the function α is known. To compute α, we first calculate the closed set $R_{i,\iota}$ and the kernel $\widehat{G}_{i,\iota}$ for the problem of asking whether there exists an (H_ι, C_ι, W)-coloring of G_i. This construction depends on the type of (H_ι, C_ι, W) and L (we use Lemmata 4.3, 4.6, and 4.9 respectively). Recall that only the type of (H_ι, C_ι, W) and L differentiates the way $R_{i,\iota}$ is being defined. Also, from the proof of Lemma 4.1, the choice of

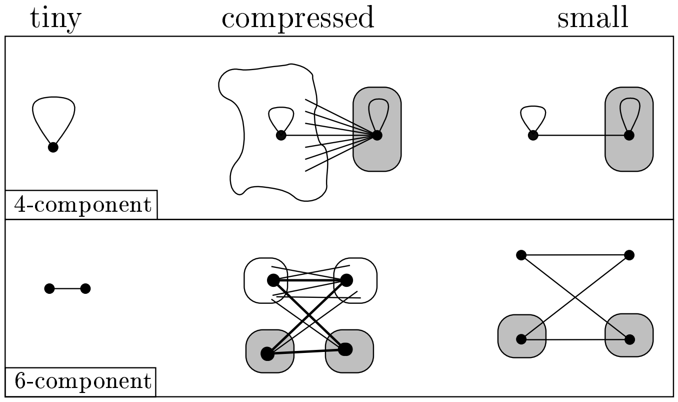

Figure 4: Some particular cases of components in a plain (H, C, K)

W is not important for the construction of these kernels as long as in the construction of the graph associated to $(G_i, R_{i,\iota})$ we keep $k+1$ vertices for each "big" equivalence class P_v and $k+1 \geq 1 + \sum_{u \in C_\iota} W(u)$ for any $W \in \mathcal{W}_\iota$. By Lemmata 4.3, 4.6, and 4.9, the construction of each $\widehat{G}_{i,\iota}$ requires $O((k+h)n_i + kh2^{h+k})$ steps where $n_i = |V(G_i)|$. Therefore, the computation of all kernels take at most $O((k+h)n + g \cdot kh2^{k+h}) = O((k+h)n + g \cdot f_1^1(k,h))$ steps. Going back to the computation of α, as long as we have the kernels, to compute $\alpha(i, \iota, W)$ for any $i \in [g], \iota \in [\gamma]$, and $W \in \mathcal{W}_\iota$ depends now only on k and h. However, as $1 \leq i \leq g$, we need $O(g \cdot f_1^2(k,h))$ steps to compute α. Choosing $f_1 \leq f_1^0, f_1^1, f_1^2$ we obtain the following result.

Theorem 5.2. For simple partially weighted graph (H, C, K), the list (H, C, K)-coloring problem can be solved in time $O((k+h)n + f_1(k,h)g + f_2(k,h))$.

6 The (H, C, K)-coloring problem for plain partially weighted graphs

In this section we show the fixed parameter tractability of the (H, C, K)-coloring problem, when (H, C, K) is plain. In order to prove the result, we reduce the problem to an equivalent (H', C', K')-coloring problem in which (H', C', K') is simple.

We call a 4-component H' of H *compressed* if $|V(H') \cap C| = 1$. A 6-component is *compressed* if $|X(H') \cap C|, |Y(H') \cap C| \leq 1$. We say that a 4-component is *small* if it is compressed and $|V(H') - C| = 1$. A 6-component is *small* if it is compressed and $|V(H') - C| = 2$. Finally, we call a 4-component or a 6-component of H *tiny* if it is small and $C = \emptyset$. Figure 4 presents the different component subtypes.

A plain partially weighted graph (H, C, K) is said to be

- *compressed* if all the 4, and 6-components of H are compressed.
- *small* if all the 4 and 6-components of H are small.
- *positive* if all its components are positive.

Observe that if (H, C, K) is plain and small then (H, C, K) is also simple.

The goal of this section is achieved through a series of parameterized reductions, from the general form of components to the *small* or *tiny* forms (see Figure 4). We assume that all the components of (H, C, K) have the form described in Section 2, even if the parameterized vertices have zero weight. Recall that $c = |C|$ and $h = |H|$.

Our first result allows us to consider only compressed plain partially weighted graphs.

Lemma 6.1. For any plain partially weighted graph (H, C, K), there exists an equivalent (H', C', K'), which is compressed and plain. Moreover, such an assignment can be computed in $O(ch)$ steps.

Proof. To prove the claim, we first show that for any plain partially weighted graph (H, C, K), where H contains $m \geq 1$ non-compressed 4-components, there exists an equivalent plain (H', C', K') such that H contains $m - 1$ non-compressed 4-components.

Let F be a non-compressed 4-component of H and set $D = C \cap V(F)$. Let $a_{\text{new}} \notin V(F)$ be a new vertex, and let, $k_D = \sum_{d \in D} K(d)$. We define (F', C', K') in the following way:

$$\begin{aligned}
&V(F') = V(F) - D \cup \{a_{\text{new}}\},\\
&E(F') = E(F[V(F) - D]) \cup \{\{a, a_{\text{new}}\} \mid a \in N_H(D) - D\} \cup \{\{a_{\text{new}}, a_{\text{new}}\}\},\\
&C' = (C \cap V(F)) - D \cup \{a_{\text{new}}\}, \text{ and}\\
&K' \text{ is such that } K'|_{C-D} = K \text{ and } K'(a_{\text{new}}) = k_D.
\end{aligned}$$

Notice that replacing F by F' gives a plain partially weighted graph (H', C', K') that can be constructed in $O(ch|V(H')|)$ steps. Let us show that $(H', C', K') \sim (H, C, K)$. Given a graph G, if χ is a (H, C, K)-coloring of G, we define $\chi' : V(G) \to V(H')$ as

$$\chi'(v) = \begin{cases} \chi(v) & \text{if } \chi(v) \notin D,\\ a_{\text{new}} & \text{otherwise.} \end{cases}$$

then χ' is a (H', C', K')-coloring of G.

On the contrary, given an (H', C', K')-coloring χ' of G, then $|{\chi'}^{-1}(a_{\text{new}})| = k_D$. Therefore, we can define a function $\sigma : {\chi'}^{-1}(a_{\text{new}}) \to D$ such that, for all $d \in D$, we have $|\sigma^{-1}(d)| = K(d)$. Define $\chi : V(G) \to V(H)$ as:

$$\chi(v) = \begin{cases} \chi'(v) & \text{if } \chi'(v) \neq a_{\text{new}},\\ \sigma(v) & \text{otherwise.} \end{cases}$$

Notice that χ is an (H, C, K)-coloring of G.

We now show that for any plain (H, C, K), where H contains $m \geq 1$ non-compressed 6-components, there exists an equivalent plain (H', C', K') such that H' contains $m - 1$ non-compressed 6-components.

Let F be a non-compressed 6-component of H and set $D_x = C \cap X(F)$ and $D_y = C \cap Y(F)$. Let $x_{\text{new}}, y_{\text{new}} \notin V(H)$ be two new vertices, and let, $k_x = \sum_{d \in D_x} K(d)$ and $k_y = \sum_{d \in D_y} K(d)$. We define (F', C', K') in the following way:

$$\begin{aligned}
&X(F') = X(F), Y(F') = Y(F) - D \cup \{x_{\text{new}}, y_{\text{new}}\},\\
&E(F') = E(F[V(F) - D_x - D_y]) \cup \{\{a, y_{\text{new}}\} \mid a \in N_H(D_x)\}\\
&\qquad \cup \{\{b, x_{\text{new}}\} \mid b \in N_H(D_y)\} \cup \{\{x_{\text{new}}, y_{\text{new}}\}\}\\
&C' = \{x_{\text{new}}, y_{\text{new}}\}, \text{ and}\\
&K' \text{ is such that } K'|_{C-D} = K, K'(x_{\text{new}}) = k_x. \text{ and } K'(y_{\text{new}}) = k_y.
\end{aligned}$$

Again replacing F by F' provides a plain partially weighted graph (H', C', K'), that can be constructed in $O(r|V(H')|)$ steps. Furthermore, it is easy to check that $(H', C', K') \sim (H, C, K)$.

The proof is complete, as we can iterate the construction until the point in which all the 4 and 6-components are compressed. □

The next result allows us to consider only small components.

Lemma 6.2. For any compressed plain partially weighted graph (H, C, K) there exists an equivalent one, (H', C, K), which is small and plain. Moreover, the assignment can be computed in $O(h)$ steps.

Proof. Again we show that for any compressed plain partially weighted graph (H, C, K), with $m \geq 1$ non-small 4-components, there exists an equivalent compressed plain partially weighted graph (H', C, K) with $m - 1$ non-small 4-components. Later on, we show a similar result for the case of 6-components.

Suppose that F is a compressed 4-component of H such that $C \cap V(F) = \{d\}$ and d is adjacent in F to a looped non-weighted vertex x. Let H' be the graph obtained from H by replacing F with $F' = F[\{x, d\}]$. Then (H', C, K) is compressed and plain, has one non-small component less, and can be constructed in $O(V(H'))$ steps.

Let us show that the two partially weighted graphs are equivalent. Notice that any (H', C, K)-coloring of G is also a (H, C, K)-coloring of G, because F' is a subgraph of F containing all the parameterized vertices in F.

Let χ be an (H, C, K)-coloring of G, and consider the function $\rho : V(H) \to V(H')$ defined as follows:

$$\rho(a) = \begin{cases} a & \text{if } a \notin V(F), \\ d & \text{if } a = d, \\ x & \text{if } a \in V(F) - \{d\}. \end{cases}$$

Notice that ρ is an H'-coloring of H, and the function $\rho \circ \chi : V(G) \to V(H')$ is an (H', C, K)-coloring of G.

Suppose that F is a compressed 6-component of H such that for some $x, y \in V(F) - C$ we have that $C \cap V(F) = \{a, b\}$, a is adjacent in F to a vertex x, and b is adjacent in F to a vertex y. Furthermore, we also have that $\{x, y\} \in E(F)$. Let H' be the graph obtained from H by replacing F with $F' = (\{x, y, a, b\}, \{\{x, y\}, \{y, a\}, \{x, b\}, \{a, b\}\})$. Then (H', C, K) is compressed and plain, has one non-small 6-component less and can be constructed in $O(1)$ steps.

In a similar way to the case of 4-components, it can be shown (H', C, K) and (H, C, K) are equivalent.

As (H', C, K) contains less non-small 4 or 6-components, the lemma follows. □

Our next results establishes the equivalence with positive partial weight assignments.

Lemma 6.3. For any plain small partially weighted graph (H, C, K) there exist an equivalent one (H', C', K') that is positive and small. Furthermore, it can be computed in $O(ch)$ steps.

Proof. Consider the set $Z = \{c \in C \mid K(c) \neq 0\}$. Notice that $(H, C, K) \sim (H[Z], Z, K|_{C \cap Z})$, as it is not possible to map vertices in G to vertices outside Z. Furthermore, as we only remove parameterized vertices, $(H, C, K)[Z]$ is plain and small. □

Observe that a tiny component is $K_{1,0}$, $K_{1,1}$, or K_1^r. Note that $K_{1,0}$ can be mapped to $K_{1,1}$ and $K_{1,1}$ can be mapped to K_1^r. Therefore, it is enough to keep only one tiny component and preserve equivalence. We conclude with the following lemma.

Lemma 6.4. For any plain, positive, and small partially weighted graph (H, C, K) there exists a plain, positive, and small partially weighted graph (H', C', K') equivalent to (H, C, K) that has at most one tiny component. Moreover, such an assignment can be computed in $O(h)$ steps.

From Theorem 5.2 and by observing that a plain, positive and small, partially weighted graph is also simple, and that after applying the sequence of reductions we get a partially weighted graph in which each component has less than four vertices, we obtain the following result.

Theorem 6.5. For any plain partially weighted graph (H, C, K), the (H, C, K)-coloring problem can be solved in $O(ch + (k + h)n + f_1(k, \gamma)g + f_2(k, \gamma))$ steps.

7 Some hardness results

It is easy to show that there are partially weighted graphs in which the list (H, C, K)-coloring problem is NP-hard, taking H to be a non bi-arc graph and selecting $C = \emptyset$. In [2] it is also proved that the list (H, C, K)-coloring problem is NP-hard for any (H, C, K) in which $H - C$ is not a bi-arc graph, independently of the selection of C. In this section we produce a family of non-plain (H, C, K) for which the (H, C, K)-coloring problem is W[1]-hard. As we already mentioned in Section 1, this is considered a strong evidence that no fixed parameter algorithm exists for the list (H, C, K)-coloring problem for this family.

Theorem 7.1. The (H, C, K)-coloring problem is W[1]-hard, in the case that $H = K_1^r \oplus H'$ and $C = V(H')$, for some graph H' which contains at least one un-looped vertex.

Proof. We provide a parameter preserving reduction from the k-independent set problem to the (H, C, K)-coloring, in the special case that $K = (k, \ldots, k)$. Given a graph G we construct a new graph G' as follows: for every non-looped vertex v of F, set G_v to be a marked copy of G. For every looped vertex v of F, set G_v to be a marked copy of K_k^r. Furthermore, for every edge $\{u, v\} \in E(F)$ with $u \neq v$, add all the edges connecting vertices of G_u with vertices of G_v.

If G has an independent set of size k, then G' has trivially an (H, C, K)-coloring.

To prove the other direction, assume that there is an (H, C, K)-coloring φ of G'. Let $C_1 \subseteq C$ be the set of non-looped vertices in F. Observe that $\varphi^{-1}(C_1)$ contains only vertices from copies of G. Let D be a maximum independent set in $H[C_1]$, let $d = |D|$ and note that $d > 1$ as, by hypothesis, C_1 contains at least one vertex. We know that $|\varphi^{-1}(D)| = kd$ and that $\varphi^{-1}(D)$ is an independent set of G'. Let $I = \{v \in V(G) |$ there is a copy of v in $\varphi^{-1}(S)\}$, and for $v \in I$ let d_v be the number of copies of v in $\varphi^{-1}(D)$. By construction, if we take a set of vertices from more that d different copies of G, the set is not independent in G', because d is the size of the maximum independent set of C_1, and thus at least two of the copies are connected. Therefore, $d_v \leq s$. We know that $\sum_{v \in I} d_v = kd$ and that $d_v \leq d$, which implies that $|I| \geq k$. Therefore, G has an independent set of size at least k. $\square$

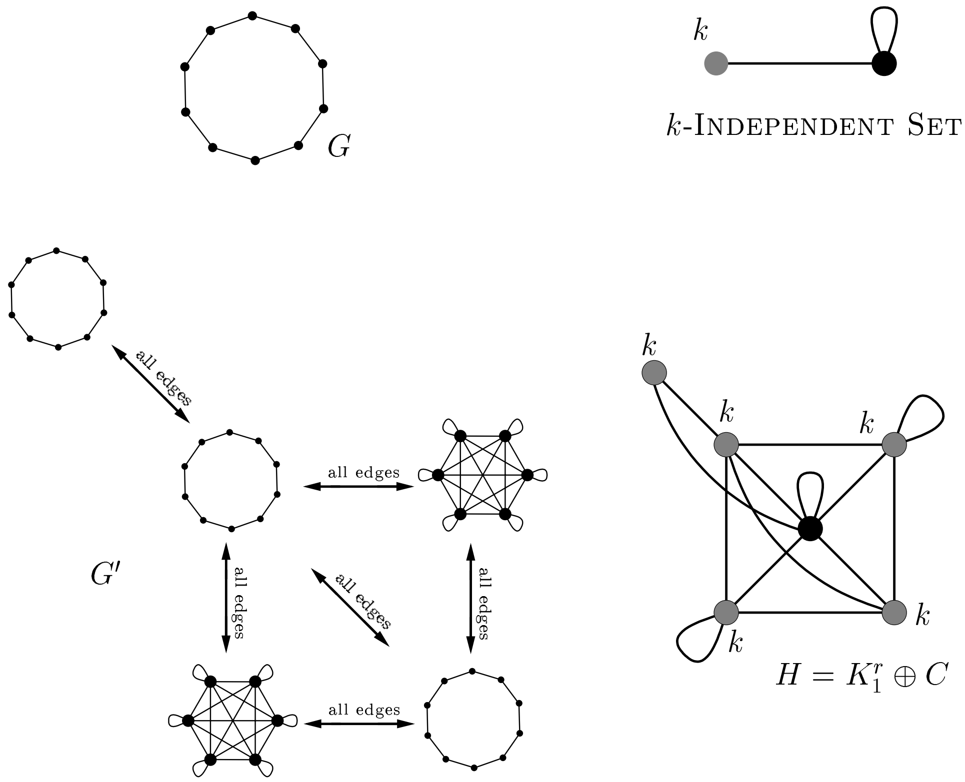

Figure 5: The construction of the graph G' in the proof of Theorem 7.1.

Assume that a partially weighted graph (H, C, K) has an un-looped vertex a in C and a looped vertex b in S, and that furthermore $\{a, b\} \in E(H)$. Given a graph G, by fixing the list of all the vertices in G to $\{a, b\}$, the corresponding list (H, C, K)-colorings correspond to the independent sets of G with k vertices, so we obtain the following result.

Theorem 7.2. The list (H, C, K)-coloring problem is W[1]-hard if there is a looped vertex in $H - C$ connected to an un-looped vertex in C.

8 Extension of the results

8.1 Computing a (list) (H, C, K)-coloring

We describe now the changes to be performed in the decision algorithms described in Sections 3, 4, and 5, in order to compute a solution to the list (H, C, K)-coloring problem, if one exists. We use the emptyfunction, $\varnothing$, to return the non-existence of a list (H, C, K)-coloring.

Algorithm Find-basic is obtained from algorithm Basic (Page 6) making it return $\varnothing$ when there is no list (H, C, K)-coloring, otherwise the coloring is obtained by selecting, for each vertex $v \in V(G)$, exactly one vertex in $L(v)$.

Algorithm Find-list-basic is obtained from algorithm List-basic (Page 8) making it return $\varnothing$ when appropriate. In the case that there is a list (H, C, K)-coloring, one of the list (H, C, K) colorings is defined by the pairs associated by the computed matching, together with a coloring on the unmatched vertices computed with an adequate call to Find-basic.

Algorithm Find-self-red is obtained from algorithm Self-red (Page 9) replacing the base calls to calls to the constructive algorithm. After a recursive call finishes, the algorithm checks whether it has provided a valid coloring. In such a case it adds to the computed coloring the last selected pair and returns it. In the case that all the recursive calls are unsuccessful, it returns $\varnothing$.

Algorithm Find-bipartite is obtained from algorithm Bipartite (Page 10) replacing the base calls to corresponding calls to the constructive algorithm for the other basic cases.

Using the results in Section 3 and the previous algorithms we have the following result:

Lemma 8.1. Let (H, C, K) be a partially weighted graph. Given a graph G and an (H, G)-list L such that there is a a list (H, C, K)-coloring of (G, L), then

- When $E(G) = \emptyset$ or $H = K_h^r$, Algorithm Find-list-basic correctly computes in $O((k + h)n + nk \log k\sqrt{n+k})$ steps, a list (H, C, K)-coloring of (G, L).
- When $E(H - C) = \emptyset$, Algorithm Find-self-red correctly computes in $O(2^k c^k[(k + h)n + nk \log k\sqrt{n+k}]))$ steps, a list (H, C, K)-coloring of (G, L).
- When $H = K_{x,y}$, Algorithm Find-bipartite correctly computes in $O((k+h)n+nk \log k\sqrt{n+k})$ steps, a list (H, C, K)-coloring of (G, L).

For the case in which G is connected and (H, C, K) is a 1,2 or 3-component, we run the previous algorithms on the corresponding kernel $\widehat{G}$. As a corollary of Lemma 8.1, we can derive an algorithm Extend that given a list (H, C, K)-coloring of $(\widehat{G}, L)$ it obtains a list (H, C, K)-coloring of (G, L), in $O(n)$ steps. Therefore, by modifying the algorithms List-coloring, for each of the three types of components, changing the call to the basic algorithm for the call to the corresponding find algorithm and computing the extension of the obtained coloring, we have algorithms for computing a list (H, C, K) coloring if there is one. The time bounds are the same as for the decision problem.

Theorem 8.2. Let (H, C, K) be a simple partially weighted graph. Given a connected graph G and an (H, G)-list L such that there is a a list (H, C, K)-coloring of (G, L). There are efficient fixed parameter algorithms for computing a list (H, C, K)-coloring of (G, L).

The following procedure is a description of an algorithm that constructs a list (H, C, K)-coloring for the general case. We first enhance the function α in such a way that $\alpha(i, \iota, W)$ returns either a list (H_ι, C_ι, W)-coloring of $(\widehat{G}_i, L_i)$ or $\varnothing$ if such a coloring does not exist. We also enhance accordingly the definitions of $\mathcal{F}(A, \iota)$ and signature.

$$\begin{aligned}\mathcal{F}(A, \iota) = \{(Z, f, \chi) \mid Z \subseteq A, |Z| \leq k_\iota, f : Z \to \mathcal{W}_\iota^+,\\ \sum_{i \in Z} f(i) = K_\iota \wedge \forall i \in Z\ \alpha(i, \iota, f(i)) = \chi\}.\end{aligned}$$

The signature of a set $A \subseteq [g]$ is the tuple $(Z_{-\rho}, f_{-\rho}, \chi_{-\rho}), \ldots, (Z_\eta, f_\eta, \chi_\eta)$ where $Z_{-\rho} \ldots Z_\eta$ form a non trivial partition of A, and for all $\iota \in [-\rho, \eta]$ and $f_\iota : Z_\iota \to \mathcal{W}_\iota$ we have $(Z_\iota, f_\iota, \chi_\iota) \in \mathcal{F}(A, \iota)$. We compute the partition $\mathcal{P}$ and the set D as we did in Section 5 for the decision version, and check whether there exists some subset $A \subseteq D$ that is proper with respect to the graph $\widehat{G}$ formed by the union of the components with indices in D. If no such a subset exists, we return $\varnothing$. Otherwise, we compute one of the signatures of A, let us say $(Z_{-\rho}, f_{-\rho}, \chi_{-\rho}), \ldots, (Z_\eta, f_\eta, \chi_\eta)$. Notice that, given the enhanced function α, this computation takes time that depends only on k and h. Then $\chi_A = \bigcup_{\iota\in[-\rho,\eta]} \chi_\iota$ is a (H,C,K)-coloring of the graph formed by the union of the components in A. As A is proper, then for any $i \in D - A$ there is a $\iota_i \in [\rho,\eta]$ such that $\alpha(i,\iota_i,\mathbf{0}) = \chi_i$. If $\chi_{A^*} = \bigcup_{i\in D-A} \chi_i$, then $\chi_A \cup \chi_{A^*}$ is an (H,C,K)-coloring of $\widehat{G}$. We have to extend this coloring to a coloring of G. By construction of M, for each $j \in [g]-D$ there are at least $k+1$ indices $i_1,\ldots,i_{k+1} \in D-A$, where $i_h \sim j$ for $h = 1,\ldots,k+1$. As we have chosen in D, $k+1$ representatives of the same class, one of them must be mapped to some component of H with zero weight. Therefore, for any $i \in [g] - A$ we can find a $\iota_i \in [\rho,\eta]$ such that $\alpha(i,\iota_i,\mathbf{0}) = \chi_i'$, which takes time that is linear on g and some function depending only on k and h. We set $\chi_{\bar{A}} = \cup_{i\in[g]-A}\chi_i'$ and return $\chi = \chi_A \cup \chi_{A^*} \cup \chi_{\bar{A}}$ as an (H,C,K)-coloring of G.

Theorem 8.3. Let (H,C,K) be a partially weighted graph, given a graph G and an (H,G)-list L. The algorithm described in the previous paragraph computes a list (H,C,K)-coloring of (G,L), if there is one, in time $O((k+h)n + f_1(k,h)g + f_2(k,h)$, where f_1 and f_2 are the functions in Theorem 5.2.

For the (H,C,K)-coloring problem, our approach performs a parameter preserving reduction that starts with a plain partially weighted graph and ends up with a simple partially weighted graph. Observe, that if we look at the coloring produced by the algorithm for simple partially weighted graphs, the pre-images of the non-weighted vertices should be maintained, while the pre-images of the weighted vertices must be redistributed among the original vertices. However, as the connection with the non-weighted vertices consists of all edges, we can separate them in whatever order to fill the weighted vertices. This leads us to the following result.

Theorem 8.4. Let (H,C,K) be a plain partially weighted graph. Given a graph G that is (H,C,K)-colorable, an (H,C,K)-coloring of G can be computed in $O(ch + (k+h)n + f_1(k,\gamma)g + f_2(k,\gamma))$ steps, where f_1 and f_2 are the functions described in Theorem 6.5.

8.2 The list $(H,C,\leq K)$-coloring

If in the definition of (H,C,K)-coloring we replace "$=$" with "$\leq$", we get what we call a $(H,C,\leq K)$-*coloring* of G (see for ex. [2]). We can define again all the parameterized versions in the same way, replacing the notion of (H,C,K)-coloring for that of $(H,C,\leq K)$-coloring. Thus, the $(H,C,\leq K)$-*coloring* problem checks whether G has an $(H,C,\leq K)$-coloring. The list $(H,C,\leq K)$-coloring problem checks whether G has a list $(H,C,\leq K)$-coloring.

In a similar way, we can define the $\leq$-*equivalent* denoted by $(H,C,K) \sim_\leq (H',C',K')$. The following result is easy to prove,

Lemma 8.5. Let (H,C,K) be a partially weighted graph, G a graph, L an (H,G)-list, and let $B = B(H,C,K,G,L)$ be the bipartite graph associated to (G,L). Assume that either 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 $(H,C,\leq K)$-coloring of (G,L). Otherwise, there is a list $(H,C,\leq K)$-coloring of (G,L) if and only if the weight of a maximum matching in B is at most kp.

Thus, Algorithm List-basic in Section 3, can be easily adapted to decide the existence of an $(H,C,\leq K)$-coloring, by changing the condition $w(M) < kp + k - p$ in the second **if** by the condition $w(M) < kp$. This change insures that the matching covers the set of vertices that must be mapped to C, although this assignment might not fill completely the vertices in

C. We refer to such a modification as the Find-list-basic-leq function. Observe that the Basic algorithm, in Section 3, also works for the considered cases of the $(H, C, \leq K)$ coloring. The remaining basic algorithms perform calls either to the Basic or to the List-basic functions. We need to add the suffix leq for denoting the variation of the algorithms given in Section 5 in which the calls to List-basic are replaced by call to List-basic-leq. We have:

Lemma 8.6. Let (H, C, K) be a simple partially weighted graph and let G be a connected graph and L an (H, G)-list.

- If G has no edges or $H = K_h^r$, List-basic-leq correctly checks whether there is a list $(H, C, \leq K)$-coloring of (G, L), in $O((k + h)n + nk \log k\sqrt{n + k})$ steps,
- If $H - C$ is edge-less, Self-red-leq(H, C, K, G, L) correctly computes a list $(H, C, \leq K)$-coloring of (G, L), if there is one, and it runs in $O(2^k c^k[(k + h)n + nk \log k\sqrt{n + k}])$ steps,
- If $H = K_{x,y}$, Bipartite-leq(H, C, K, G, L) correctly computes a list $(H, C, \leq K)$-coloring of (G, L), if there is one, and it runs in $O((k + h)n + nk \log k\sqrt{n + k})$ steps.

For the general case we have to adapt the definition of the set M. After doing so, the rest of the algorithm is similar. In fact, we only have to modify the definition of $\mathcal{F}$ given in Section 5. Given $A \subseteq [g]$ and $\iota \in [-\rho, \eta]$, define

$$\begin{aligned}\mathcal{F}(A, \iota) = \{(Z, f) \mid Z \subseteq A, |Z| \leq k_\iota, f : Z \to \mathcal{W}_\iota^+,\\ \sum_{i \in Z} f(i) \leq K_\iota \wedge \forall i \in Z\ \alpha(i, \iota, f(i)) = 1\}\end{aligned}$$

The signature of a subset of $[g]$ and the partition $\mathcal{P}^{\leq}$ of $[g]$ are defined exactly as in Section 5. The same holds also for the construction of the set $D^{\leq}$ and the graph $\widehat{G}$. The following version of Theorem 5.1 holds.

Lemma 8.7. Let (H, C, K) be a simple partially weighted graph. Given a graph G together with an (H, G)-list L, G has a list $(H, C, \leq K)$-coloring if and only if there is an $A \subseteq D^{\leq}$ that is proper with respect to $\widehat{G}$.

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

Theorem 8.8. Let (H, C, K) be a simple partially weighted graph. Then, the list $(H, C, \leq K)$-coloring problem can be solved in $O((k + h)n + f_1'(k, h)g + f_2'(k, h))$ steps.

We have just proved that when (H, C, K) is simple, the list $(H, C, \leq K)$-coloring problem is in FPT. Observe that the reductions presented in Section 6 are also valid when considering the $(H, C, \leq K)$-coloring. Therefore, we conclude that the (H, C, K)-coloring problem is in FPT when (H, C, K) is plain.

Theorem 8.9. Let (H, C, K) be a plain partially weighted graph. Then the $(H, C, \leq K)$-coloring problem can be solved in $O(ch + (k + h)n + f_1'(k, \gamma)g + f_2'(k, \gamma))$ steps.

9 Conclusions and open problems

We have shown that for the family of simple partially weighted graphs, the list (H, C, K)-coloring problem has an efficient linear fixed parameter algorithm. We also have shown that for the family of plain partially weighted graphs, the (H, C, K)-coloring problem has an efficient linear fixed parameter algorithm.

As a future line of research it is worth to mention that the class of simple partially weighted graphs has been obtained by starting from a complete reflexive graph or a complete bipartite irreflexive graph and fixing the parameterized vertices. Next step is to look for other edges in the parameterized part that could be removed. It is clear from the hardness results that loops play a especial role, as by removing them from some of the weighted vertices of a reflexive graph or by adding them to some unweighted vertices of an irreflexive graph we get partially weighted graphs for which the list (H, C, K)-coloring is W[1]-hard. It remains to explore other classes of variations in which we add or remove edges.

Finally let us mention that recently the k-Almost Bipartite Graph (mentioned in the introduction) has been shown to belong to FPT [16]. This problem can be reformulated as a (H, C, K)-coloring problem where $H = K_{1,1} \oplus K_1^r$ and $C = V(K_1^r)$ (see Figure 1). A natural extension is to consider $H = K_{x,y} \oplus K_c^r$ and put weight only in the reflexive clique. We conjecture that for such a partially weighted graph the (H, C, K)-coloring problems is also fixed parameter tractable.

References

[1] J. F. Buss and J. Goldsmith. Nondeterminism within P. *SIAM Journal on Computing*, 22(3):560–572, 1993.

[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. 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, vol 63, pp. 65–85. American Mathematical Society, 2004.

[4] J. Díaz, M. Serna, and D. Thilikos. The restrictive H-coloring problem. *Discrete Applied Mathematics*, 145:297–305, 2005.

[5] J. Díaz, M. Serna, and D. Thilikos. Fixed parametar algorithms for counting and deciding bounded restrictive H-coloring. In S. Albers and T. Radzic, editors, *Algorithms ESA 2004*, volume 3221, pages 275–286. Lecture Notes in Computer Science, Springer-Verlag, 2004.

[6] J. Díaz, M. Serna, and D. M. Thilikos. Complexity issues on Bounded Restrictive H-colorings. *Discrete Applied Mathematics*, to appear, 2005.

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

[8] R. G. Downey and M. R. Fellows. Fixed-parameter tractability and completeness. I. Basic results. *SIAM Journal on Computing*, 24(4):873–921, 1995.

[9] 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.

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

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

[12] T. Feder, P. Hell, and J. Huang. Bi-arc graphs and the complexity of list homomorphisms. *Journal of Graph Theory*, 42:61–80, 2003.

[13] H. Gabow and R. Tarjan. Faster scaling algorithms for network problems. *SIAM Journal on Computing*, 18:1013–1036, 1989.

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

[15] P. Hell and J. Nešetřil. *Graphs and Homomorphisms* Oxford University Press, 2004.

[16] B. Reed, K. Smith, and A. Vetta. Finding odd cycle transversals. *Operation Research Letters*, 32:299-301, 2004.