Efficient algorithms for counting parameterized list H-Colorings ⋆

Josep Díaz Maria Serna

ALBCOM Research group, Dept. Llenguatges i Sistemes Informàtics
Universitat Politècnica de Catalunya,
Campus Nord, Edifici Ω, Jordi Girona Salgado 1-3
08034, Barcelona, Spain.

Dimitrios M. Thilikos

Department of Mathematics
National and Kapodistrian University of Athens
Panepistimioupolis, GR157 84, Athens, Greece

Abstract

We study the fixed parameter tractability of the counting version of a parameterization of the restrictive list H-coloring problem. The parameterization is defined by fixing the number of preimages of a subset C of the vertices in H through a weight assignment K on C. We show the fixed parameter tractability of counting the number of list (H, C, K)-colorings, for the case in which (H, C, K) is simple. We introduce the concept of compactor and a new algorithmic technique, *compactor enumeration*, that allow us to design fixed parameter algorithms for parameterized counting problems.

Key words: restrictive H-coloring, restrictive list H-coloring, parameterization, counting problems.

⋆ Work partially supported by the FET pro-actives Integrated Project 15964 (AEOLUS) and by the Spanish MEC TIN2005-25859-E. The work of the first author was partially supported by the *Distinció per a la recerca of the Generalitat de Catalunya* and MEC TIN2007-66523 (MEFOALDISI). The work of the second author was partially supported by MEC TIN-2005-09198-C02-02 (ASCE). The work of the third author was partially supported by project KAPODISTRIAS (ΑΠ 736/24.3.2006) of the National and Kapodistrian University of Athens (project code: 70/4/8757)

Email addresses: diaz@lsi.upc.edu (Josep Díaz), mjserna@lsi.upc.edu (Maria Serna), sedthilk@math.uoa.gr (Dimitrios M. Thilikos).

Problem	P	#P-complete	Ref
list $\#H$-coloring	(1)	not (1)	[22]
$\#H$-coloring	(1)	not (1)	[16,15]
restrictive list $\#H$-coloring	(1)	not (1)	[7]
restrictive $\#H$-coloring	(1)	not (1)	[7]
list $\#(H,C,K)$-coloring	(1)	not (1)	[9]
$\#(H,C,K)$-coloring	(2)	not (2) and (H,C,K) irreducible	[9]

(1) All the connected components of H are either complete reflexive graphs or complete irreflexive bipartite graphs.

(2) All the connected components of $H - C$ are either complete reflexive graphs or complete irreflexive bipartite graphs.

Table 1
Complexity of the parameterized counting H-coloring problems

1 Introduction

Given two graphs G and H, an *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. 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 in G is known as the *list H-coloring*. The case in which the number of preimages of the vertices in H are part of the input is known as the *restrictive H-coloring* or the *restrictive list H-coloring*. See [23] for more variations on the problem and complexity results.

We consider the counting problems associated to the previous problems. The $\#H$-coloring, the list $\#H$-coloring, the restrictive $\#H$-coloring, and the restrictive list $\#H$-coloring. The known results concerning their complexity are summarized in Table 1. We use the term *dichotomy* for a property P and a problem Π when we know that Π is polynomial time solvable on instances that satisfy P and #P-hard on instances that do not satisfy P. Observe that most of the coloring problems have such a dichotomy.

In this paper, we continue the study of the various versions of the restrictive

$\#H$-coloring problem from the viewpoint of Parameterized Complexity. The idea of parameterization appeared as an alternative approach to cope with the hardness of combinatorial problems coming from the classic Complexity Theory. When parameterizing, we split the input of the problem to a main part and a parameterized part, hoping that there exists an algorithm in which the super-polynomial part of its running time depends exclusively of the parameter. When this holds and the parameter is small, which might be the case for most real applications of the problem, we can consider such an algorithm efficient. More formally, a parameterized problem is a pair (S, K) where S is the main part and K is the parameteried part. The Parameterized Complexity settles the question of whether the problem is solvable by an FPT-*algorithm*) with time complexity

$$f(K)\, n^{O(1)}$$

where $f(K)$ is a (super-polynomial) function that depends only on the parameterized part and n is the size of the input. If there exists such an algorithm, we say that the parameterized problem belongs to the class FPT. In a series of fundamental papers (see [12,13,1,10,11]), Downey and Fellows defined a hierarchy of complexity classes, namely the classes $\mathsf{W}[1] \subseteq \mathsf{W}[2] \subseteq \cdots \subseteq \mathsf{W}[SAT] \subseteq \mathsf{W}[P]$ and proposed suitable types of reductions such that hardness for some of the above classes makes it rather impossible that a problem belongs in FPT (we recall that $\mathsf{FPT} \subseteq \mathsf{W}[1]$). The previous theoretical framework is used for the classification of several parameterized problems according to their parameterized complexity, and since then, it has been extended to problems of counting in [20] and [25,26]. See [20,24] for further references on Parameterized Complexity, .

A parameterization for the *restrictive H-coloring* was defined in [4] where the parameterized part is H together with the number of required preimages of some set C of vertices in H. The corresponding parameterized problem is known as the (H, C, K)-*coloring* problem. In other words, in the triple (H, C, K), H is a fixed graph, C is the set of vertices with restrictions, and K is a weight assignment on the vertices of C (encoded by a function $K : C \to \mathbb{N}$) defining the number of the required preimages of these vertices. Thus, the *partially weighted graph* (H, C, K) constitutes the parameterized part of the problem while the input graph G constitutes the main part.

In [4,9], we studied the complexity of the (H, C, K)-coloring problems and in [8] we provided efficient fixed parameter algorithms for some particular classes of partially weighted graphs. See [5] for a survey on different parameterized problems based on H-colorings.

In this paper we are interested in finding classes of (H, C, K) for which the list $\#(H, C, K)$-coloring problem is *fixed parameter tractable*. In a previous paper [8] , we have designed linear time fixed parameter algorithms for the list (H, C, K)-coloring problem and for the (H, C, K)-coloring problem (see also

[6]). In the present paper we consider the class of partially weighted graphs, the *simple partially weighted graphs* introduced in [8], and we provide a fixed parameter algorithm for the list $\#(H, C, K)$ problem.

One of the fundamental tools for designing efficient fixed-parameter algorithms for decision problems is the so called *reduction to problem kernel*. The method consists in constructing a polynomial time self-reduction that transforms a problem input (S, K) to another instance (S', K'), in such a way that the size of the new instance depends *only* on some function of K. The instance (S', K') is refered to as *the kernel*. The interested reader can look to [14,19,24] for discussions on fixed parameter tractability and the ways to construct 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 [20,25,26]). However, so far, no progress has been done in the development of algorithmic techniques for parameterized counting problems.

In this direction, we introduce a new algorithmic tool, that we call the *compactor enumeration*, which instead of isolating a kernel to which the counting problem can be reduced to, develops a technique correspondent to the "reduction to problem kernel" to deal with counting problems. We define a general type of set, called the *compactor*, which contains a *certificate* for each class in a suitable partition of the solution space of (S, K). The size of the compactor, as well as the sizes of its elements, depends only on the parameterized part of the problem. Formally, given a parameterized problem Π, let $\mathsf{Sol}(S, K)$ be the set of solutions for input (S, K). We say that a set $\mathsf{Cmp}(S, K)$ is a *compactor* for $\mathsf{Sol}(S, K)$ if

(1) $|\mathsf{Cmp}(S, K)|$ is a function that depends only on K.
(2) $\mathsf{Cmp}(S, K)$ can be enumerated with an algorithm whose complexity depends only on $|\mathsf{Cmp}(S, K)|$.
(3) There is a surjective function $\mathcal{M} : \mathsf{Sol}(S, K) \to \mathsf{Cmp}(S, K)$.
(4) For any $s \in \mathsf{Cmp}(S, K)$, the value $|\mathcal{M}^{-1}(s)|$ can be computed within time $f(K)\ n^{O(1)}$.

Observe that when the above conditions hold, we obtain a fixed parameter algorithm for counting the solutions of the initial problem by *enumerating* all the certifying structures and computing, for each one of them, the number of solutions they certify.

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 consist of an instance of the initial problem as it is the case in a kernelization. In particular, some of our compactors are sets whose elements are defined by a structure formed with a suitable collection of functions.

The complexity of all our algorithms is *linear* in n, and it is of the form $O(f_1(\kappa)n + f_2(\kappa)g + f_3(\kappa)\log n + f_4(\kappa))$, where κ depends on (H, C, K), g is the number of connected components of G and $f_1 \ll f_2 \ll f_3 \ll f_4$, thus functions f_i increase in complexity. We assume that the input graph is given as a collection of its connected components, each connected component is given by an array of adjacency lists. In case that the connected components are not given we incurred in an additional $O(n + m)$ time to compute the connected components of G [3].

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 a simple partially weighted graph. Therefore, the techniques developed in this paper can be used to produce efficient fixed-parameter algorithms for other counting problems. For instance counting the number of bipartite subgraphs of size k whose removal leaves a complete graph or counting the number of vertex covers of size k. The later problem is known to be in FPT [20].

2 Definitions and preliminary results

2.1 Graphs, functions, and sets

Given a graph G, let $V(G)$ denote its vertex set and let $E(G)$ 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. For $U \subseteq V(G)$, $G[U]$ denotes the *subgraph of G induced by U*. Following the terminology in [17,18] 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 a vertex v let $N_G(v) = \{u \in V(G) | \{u, v\} \in E(G)\}$ denote the neighborhood of v in G.

Given a bipartite connected graph G, the vertex set $V(G)$ is partitioned in two sets $X(G)$ and $Y(G)$ so that every edge has an end point in the X-part and another one in the Y-part. We assume that the X and Y parts of a connected bipartite graph are defined without ambiguity.

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 $G \oplus G'$ with vertex set $V(G) \cup V(G')$ and edge set

$$E(G) \cup E(G') \cup \{\{u, v\} \mid u \in V(G), v \in V(G')\}.$$

We use standard graph terminology: $K_{r,s}$ denotes the complete irreflexive bipartite graph on two parts with r and s vertices each, notice that $K_{r,0}$ denotes an *independent set* in the notation of a complete bipartite graph, and K_n^r denotes the reflexive clique with n vertices.

For any function $\varphi : A \to B$ and any subset $D \subseteq A$, we define the *restriction of φ to D* by $\varphi|_D = \{(a, b) \in \varphi \mid a \in D\}$. 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 $A \cup A'$ to $\mathbb{N}$ defined by:

$$(\varphi \cup \psi)(x) = \begin{cases} \varphi(x) & \text{if } x \in A, \\ \psi(x) & \text{if } x \in A'. \end{cases}$$

As usual, given two functions $\varphi, \psi : A \to \mathbb{N}$, define their sum $\varphi + \psi$ as: for any $x \in A$, set $(\varphi + \psi)(x) = \varphi(x) + \psi(x)$. We say that $\phi \leq \psi$ if, for any $x \in A$, $\phi(x) \leq \psi(x)$. We denote by $\varnothing$ the empty function.

We often will need sets of consecutive indices, for $n, m \in \mathbb{N}$, we use the notation,

$$\begin{aligned} [n] &= \{1, \ldots, n\}, \\ [-m] &= \{-m, \ldots, -1\}, \text{ and} \\ [-m, n] &= \{-m, \ldots, -1, 0, 1, \ldots, n\}. \end{aligned}$$

2.2 H-coloring, variants and parameterization

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 fixed graph H and 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 will refer to the vertices of C as the *weighted vertices*. Given a partial weight assignment (C, K) on H together with an (H, G)-list L, a mapping $\chi : V(G) \to V(H)$ is a *restrictive list H-coloring* of (G, L) and (C, K), if χ is a list H-coloring of (G, L) such that for all $a \in C$, we have $|\chi^{-1}(a)| = K(a)$. In Figure 1 we give examples of restrictive H-colorings of a graph G.

We consider the parameterization of the restrictive list H-coloring problem in

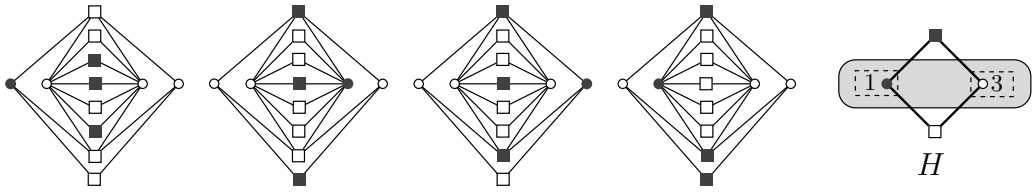

Fig. 1. A graph G a partial weight assignment (C, K) on H, and some of its restrictive H-colorings. Notice that there are $4 \times 2^8 = 1024$ restrictive H-colorings of G and (C, K).

G	
g	# of connected components
n	# of vertices of G ($\|V(G)\|$)

(H, C, K)	
γ	# of connected components f H
h	# of vertices of H ($\|V(H)\|$)
c	# of weighted vertices ($\|C\|$)
s	# of non weighted vertices ($\|V(H) - C\|$)
k	total weight ($\sum_{c \in C} K(c)$)

Table 2
Notation and parameters used in the analysis of our algorithms.

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 weight assignment and refer to it as a *partially weighted graph*. For a partially weighted graph (H, C, K), we use the notation $S = V(H) - C$, $h = |V(H)|$, $c = |C|$, $s = |S|$ and $k = \sum_{c \in C} K(c)$.

All the running times of the algorithms of this paper are expressed in terms of measures that depend either on the size of the main part ($|V(G)|$, g) or the parameterized part (h, c,k). As our objective is to design parameterized algorithms, the super-polynomial part of their running times will depend exclusively on the parameterized part. To facilitate the reading of the paper, in Table 2 we give a summary of the values that appear in the analysis of the running times of our algorithms.

A *list* (H, C, K)*-coloring* of (G, L) is a restrictive H-coloring of (G, L) and (C, K). For a fixed partially weighted graph (H, C, K), given a graph G and an (H, G)-list L, the (H, C, K)*-coloring* problem asks whether there exists a list (H, C, K)-coloring of (G, L).

The proposed parameterization allow us to capture some well known param-

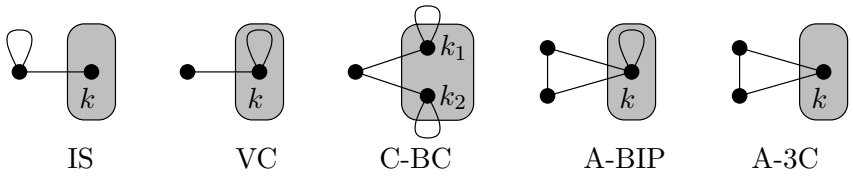

Fig. 2. Five partially weighted graphs (H, C, K). The first two correspond to the parameterized independent set and vertex cover and the last two to the k-almost bibartite and the k-almost 3-coloring problem.

eterized problems as particular cases. In Figure 2 we present three examples, where the (H, C, K)-coloring problem corresponds to the parameterized independent set problem (IS), the parameterized vertex cover problem (VC), and the k-almost bipartite subgraph problem (A-BIP) [28]. The shadowed regions represent the set C. This setting also allows us to consider new problems. For example, the problem of determining the existence of a bipartite subgraph of size k such that the removal of all of its vertices leaves a complete graph. This property holds if the complement of the given graph can be mapped to the third partially weighted graph in Figure 2 (C-BC), for some decomposition of k into a pair k_1, k_2 with $k_1 + k_2 = k$. Another interesting problem is the *almost 3-coloring problem* which asks whether a given graph is 3-colorable under the restriction that one of the colors can be used only to color k vertices, this problem corresponds to the (A-3C) partially weighted graph in Figure 2.

We are interested in counting the number of list (H, C, K)-colorings of a given graph and (H, G) list. We refer to this counting problem as the *list* $\#(H, C, K)$*-coloring* problem. We use the notation $\mathcal{H}_{(H,C,K)}(G, L)$ for the set of all list (H, C, K)-colorings of (G, L). Thus, the list $\#(H, C, K)$-coloring problem asks for the cardinality of $\mathcal{H}_{(H,C,K)}(G, L)$. We often use $\mathcal{H}(G, L)$ when (H, C, K) is clear from the context.

In contrast to the H-coloring, if the input graph G has more than one connected component, in general an (H, C, K)-coloring cannot be obtained as the disjoint union of (H, C, K)-colorings for their connected components. This fact is expressed by the next two lemmata, whose proof is straightforward.

Lemma 2.1 *Let (C, K_i), $i = 1, \ldots, t$, be partial weight assignments of a fixed graph H and let G be a graph with connected components G_i, $i = 1, \ldots, g$. If, for any i, $1 \leq i \leq s$, χ_i is an (H, C, K_i)-coloring of G_i, then $\chi^* = \chi_1 \cup \cdots \cup \chi_t$ is an $(H, C, K_1 + \cdots + K_g)$-coloring of G.*

Lemma 2.2 *Let (H, C, K) and (H', C', K') be two partially weighted graphs, let G and G' be two disjoint 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 $G \cup G'$.*

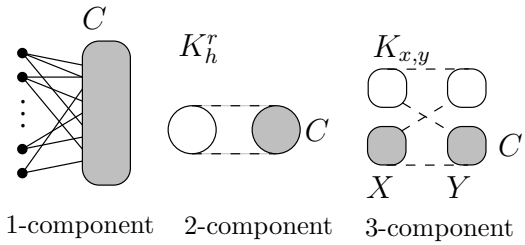

Fig. 3. Components for a simple partial weighted assignments.

2.3 Types of components

Now we recall the definition of some types of partially weighted graphs on connected graphs, an illustration of these types is given in Figure 3.

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

A partially weighted graph (H, C, K) is said to be *simple* when, for each connected component H' of H, the partial weight assignment $(H', C \cap H', K|_{H'})$ is either a 1-component, a 2-component or a 3-component.

Observe that for the partially weighted graphs given in Figure 2, VC and C-BC, are 1-components (and thus simple), while IS, A-BIP, and A-3C are not simple partially weighted graphs.

In our algorithm we assume that (H, C, K) is preprocessed once to compute the connected components of H, and classify them according to the above three types. The overall additional computation can be performed in $O(h^2)$ time.

2.4 Kernelizations

We recall some of the results in [8] that we will use later to construct the compactors.

First we consider an equivalence relation on the vertices of a connected graph G. Let (H, C, K) be a partially weighted graph. Given a graph G, together with an (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 $v \in V(G)$, P_v denotes the set $\{u \mid u \sim v\}$ and for any $Q \in \mathcal{P}$, we select a representative vertex $v_Q \in Q$. Observe that, as G has no loops, the vertices in any equivalence class form an independent set. Furthermore, if there is an edge joining two vertices in two different classes, then the subgraph induced by those classes is a complete bipartite graph.

We say that $R \subseteq V(G)$ is a *closed set* of the partition $\mathcal{P}$, if for any $v \in R$ we have $P_v \subseteq R$. In [8] there is given a construction that associates to a connected graph G and a partially weighted graph (H, C, K) a closed set, the construction depends on whether (H, C, K) is a 1, 2 or 3-component. These closed sets are used in [8] to define a kernel graph. In the next section we use them to define a compactor.

Given a connected 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 with 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'. This splitting corresponds to the classical kernelization for Vertex Cover as given for example in [2].

Lemma 2.3 ([8]) *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 we have*

- *$R = R_1 \cup R_2$ is a closed set.*
- *If either $|R_1| > k$ or $|R_2| > k^2 + k$, then there are no (H, C, K)-colorings of (G, L).*
- *It is possible to decide in $O(kn)$ steps whether $|R_1| \leq k$ and $|R_2| \leq k^2 + k$ and, if so, compute the set R.*

When $H = K_h^r$, for any partially weighted graph (H, C, K), there is 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 $G' = (V(G), \emptyset)$.

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 $N_1 = \{v \in V(G) \mid L(v) \cap S = \emptyset\}$ and $N_2 = V(G) - N_1$.

Lemma 2.4 ([8]) *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 we have*

- *N_1 is a closed set.*
- *If $|N_1| > k$, then there are no (H, C, K)-colorings of (G, L).*
- *It is possible to decide in $O(hn)$ steps whether $|N_1| \leq k$ and, if so, compute the set N_1.*

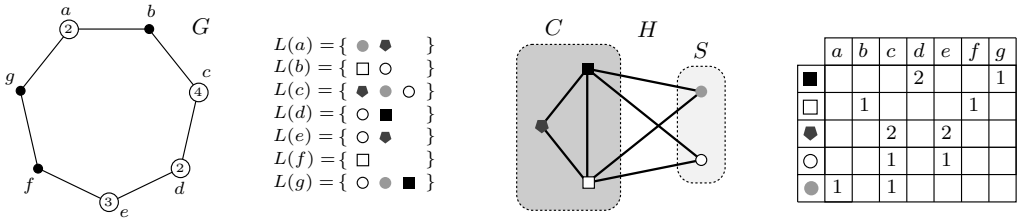

Fig. 4. An example of a (H, C, K)-coloring of the tribal graph G. The individuals of G are denoted by black vertices. In the table representing the (H, C, K)-coloring, empty positions correspond to 0's.

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 $M_3 = \{v \in V(G) \mid L(v) \cap S \neq \emptyset\}$, $M_1 = X(G) - M_3$ and $M_2 = Y(G) - M_3$.

Lemma 2.5 ([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 we have*

- *$M = M_1 \cup M_2$ is a closed set.*
- *If $|M_1| + |M_2| > k$, then there are no (H, C, K)-colorings of (G, L).*
- *In $O(hn)$ steps it is possible to decide whether $|M_1| + |M_2| > k$, if so, compute the set M*

3 Tribal graphs

We introduce some vertex-weighted graphs and extend the notion of list (H, C, K)-coloring to them. These concepts will play a significant role later.

A *tribal graph* is a graph G together with a vertex 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 $I(G) = V(G) - T$ *individuals* of G. For sake of readability, we denote a tribal graph as $\widetilde{G}$, avoiding the explicit reference to the vertex weight assignment.

A *list (H, C, K)-coloring* of a tribal graph $\widetilde{G}$ and an $(H, \widetilde{G})$ list L is a mapping $w : V(\widetilde{G}) \times V(H) \to \{0, \ldots, k\}$ where:

(1) For all $v \in V(\widetilde{G})$ and $a \in S$, we have $w(v,a) \leq 1$.
(2) For all $v \in V(\widetilde{G})$ and $a \in C$, we have $w(v,a) \leq K(a)$.
(3) For all $v \in V(\widetilde{G})$, we have $1 \leq \sum_{a \in H} w(v,a) \leq p(v)$.
(4) If $\{v,u\} \in E(\widetilde{G})$ then, for all $a,b \in H$ with $w(v,a) > 0$ and $w(u,b) > 0$, we have $\{a,b\} \in E(H)$.
(5) For all $a \in C$, we have $\sum_{v \in V(\widetilde{G})} w(v,a) = K(a)$,
(6) For all $v \in V(\widetilde{G})$ and $a \in V(H)$ with $w(v,a) > 0$, we have $a \in L(v)$.

An example of a list (H,C,K)-coloring of a tribal graph is given in Figure 4. Notice that if $C = \emptyset$, the above definition coincides with the concept of list H-coloring. Moreover, in the case that, for any $v \in V(\widetilde{G})$, $p(v) = 1$ 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 $w(v,a) = 1$. As usual, a *(H,C,K)-coloring of a tribal graph* is a list (H,C,K)-coloring for the particular case in which $L(u) = V(H)$, for any $u \in V(G)$.

Given a list (H,C,K)-coloring of a tribal graph $\widetilde{G}$ w, for any $c \in C$ define $\pi(w,c) = \sum_{v \in V(\widetilde{G})} w(v,c)$. Notice that $\pi(w,c)$ is the total number of preimages that vertex c gets in the coloring represented by w.

The following result assumes that a tribal graph $\widetilde{G}$ and an $(H,\widetilde{G})$-list are represented by a data structure that holds an array that, for each vertex v, keeps a circular linked list of neighboring vertices, an integer keeping $p(v)$, and a boolean array of size h representing $L(v)$. We also use $\widetilde{n}$ ($\widetilde{m}$) to represent the number of vertices (edges) of a tribal graph $\widetilde{G}$.

Lemma 3.1 *Let (H,C,K) be a partially weighted graph. Let $\widetilde{G}$ be a tribal graph and let L be a $(H,\widetilde{G})$-list. Given $w : V(\widetilde{G}) \times V(H) \to \{0,\ldots,k\}$ we can check whether w is a list (H,C,K)-coloring of $\widetilde{G}$ in time $O((h\widetilde{n} + hr\widetilde{m})$. Furthermore,*

$$|\{w : V(\widetilde{G}) \times V(H) \to \{0,\ldots,k\}\}| \leq k^{(h+1)\widetilde{n}},$$

and the set $\{w : V(\widetilde{G}) \times V(H) \to \{0,\ldots,k\}\}$ can be enumerated using $(\widetilde{n}+h)$ steps per element of the set.

Proof. It is straightforward to see that we can check in time $O(h\widetilde{n} + hr\widetilde{m})$ the 6 conditions on the definition of list (H,C,K)-coloring for a tribal graph.

The number of mappings of $V(\widetilde{G}) \times V(H)$ to $\{0,\ldots,k\}$ is at most $((k+1)^c 2^s)^{\widetilde{n}}$, as for each vertex we have to select c weights in the range $0,\ldots,k$ with total sum k, and s weights in the range $\{0,1\}$. Furthermore, it is straightforward to see that all the mappings can be generated in order, with a cost of $\widetilde{n} + h$ steps per element. □

In the remaining of the section, we assume that a connected graph G is given.

We associate to G a tribal graph $\widetilde{G}$ using the equivalence relation $\mathcal{P}$. Our aim is to show that the set $\mathcal{H}(\widetilde{G}, L)$ is a compactor for $\mathcal{H}(G, L)$. Although $\widetilde{G}$ is different for each type of component, the construction follows in all cases the same basic step: the selection of an adequate set of classes all of whose elements will be maintained in the tribal graph. Observe that $V(\widetilde{G})$ can be seen as a subset of $V(G)$, therefore we keep the name L for the $(H, \widetilde{G})$ list $L|_{V(\widetilde{G})}$.

Let R be a closed set. The *tribal graph associated to* (G, R) is defined to be

$$\widetilde{G} = (G[R \cup \{v_Q \mid Q \cap R = \emptyset\}], p),$$

where $p(v) = 1$ when $v \in R$, and $p(v_Q) = \min\{|Q|, k + s\}$ for any Q with $Q \cap R = \emptyset$.

In our compactor construction we use the closed set associated to a graph G, depending whether (H, C, K) is a 1,2 or 3-component, as described in the previous section. We also need some additional properties regarding representability of the complements of the closed sets.

For a subset $U \subseteq V(G)$ let $\mathcal{N}(U) = \{N_G(u) \mid u \in U\}$. We say that subset $U \subseteq V(G)$ is (l, t)-*representable* if each element $M \in \mathcal{N}(U)$ can be represented by a sorted list of numbers $\{\alpha_1, \ldots, \alpha_{l_M}\} \subseteq \{1, \ldots, l\}$ such that

(1) for all $M \in \mathcal{N}(U)$, $l_M \leq l$, and
(2) given $u \in U$ the list representing $N_G(u)$ can be computed in time t.

Lemma 3.2 *Let (H, C, K) be a partially weight assignment and let G be a connected graph given together with an (H, G)-list L. Then,*

1. *If $H - C$ is edgeless, let (R_1, R_2, R_3) be the k-splitting of G and set $R = R_1 \cup R_2$. Then, $\overline{R}$ is (k, k)-representable. Furthermore, the tribal graph $\widetilde{G}$ associated to $(G, R, \mathcal{P})$ has at most $k^2 + 2k + 2^{k+h}$ vertices and $O(k^3 + k2^{k+h})$ edges.*
2. *If $H = K_h^r$, let (N_1, N_2) be the list-splitting of (G, L). Then $\overline{N}_1$ is $(1, 1)$-representable and the tribal graph $\widetilde{G}$ associated to $(G, N_1, \mathcal{P})$ has at most $k + 2^h$ vertices and no edges.*
3. *If $H = K_{xy}$, let (M_1, M_2, M_3) be the bipartite list partition of (G, L), and set $M = M_1 \cup M_2$. Then, $\overline{M}$ is $(1, 2)$-representable and the tribal graph $\widetilde{G}$ associated to $(G, M, \mathcal{P})$ has at most $k + 2^{h+1}$ vertices and $(k + 2^{h+1})^2$ edges.*

Proof. To prove case 1, we have to take into account that for any $v \in R_3$ we have $N_G(v) \subseteq R_1$. Moreover, in the case that R_1 has at most k vertices, R_1 can be represented by $\{1, \ldots, |R_1|\}$. Using those labels we can represent a subset of R_1 with a sorted list of numbers in the range $\{1, \ldots, |R_1|\}$. Using

radix sort, the previous description allows us to compute, for any given vertex v, the list representing $N_G(v)$ in time k.

In case 2 we may assume that G has no edges. This fact simplifies the equivalence classes as we only have one type of neighborhood in G, the empty one. Thus, $\overline{R}$ is $(1,1)$-representable.

For case 3, notice that we can assume that G is a complete bipartite graph, therefore the number of different neighborhoods on G is two, and any set of vertices of G is $(2,1)$ representable. □

4 The counting algorithm for connected graphs

In the previous section we have associated to a connected graph G a tribal graph $\widetilde{G}$ that depends on the type of (H,C,K). In this section, we prove that, in the three cases, $\mathcal{H}(\widetilde{G},L)$ is a compactor for $\mathcal{H}(G,L)$. For doing so we use the two properties, first that R is a closed set, and second that $\overline{R} = V(G) - R$ is (l,t)-representable. Observe that from Lemma 3.2 we know that the values of l and t depend only on the parameter. Our aim is to show that, in the case that G is connected and (H,C,K) is a 1, 2 or 3-component, the following algorithm solves the list $\#(H,C,K)$-coloring, when selecting the adequate choices for R, which depend on the type of component.

```
function Count-list-colorings(H,C,K,G,L)
    (R,A,B) := Closed-set(H,C,K,G,L);
    if R[0] = 0 then return 0; end if
    GTR := Tribal(H,C,K,G,L,R,A,B);
    ν := Count-exten(H,C,K,G,L,GTR);
    output (ν)
end
```

In the above algorithm `R` is an array indicating the elements in the closed set R. `R`[0] holds a zero in the case that the function computing `R` detects that there are no list (H,C,K)-colorings of (G,L). `A` and `B` are auxiliary arrays used to maintain a double link between the possible neighbors of vertices not in R and a compact labeling of them, as defined in Lemma 3.2. The relabeling is kept in array `A` while array `B` is used to keep the reverse relation. The algorithms given in [8] can be easily adapted to implement the adequate Closed-set function the return in addition to `R` the arrays `A` and `B` without increase in their complexity.

In the following subsections, we show some basic properties of compactors that follow from the definition of the associated tribal graph. Once this is

done, we analyze the complexity of the algorithms Tribal, which outputs the associated tribal graph, and Count-exten, which computes the number of solutions. Finally, using the definition of the closed set for the cases of 1, 2 and 3-components, we conclude that the list $\#(H, C, K)$-coloring, when G is connected and (H, C, K) is a 1, 2, or 3-component, has a fixed parameter algorithm.

4.1 Property (3) of the compactor

In this section we establish the relationship between $\mathcal{H}(G, L)$ and $\mathcal{H}(\widetilde{G}, L)$ by showing that the third condition of the definition of compactor holds for $\mathcal{H}(\widetilde{G}, L)$. In order to obtain this result, we need the fact that R is a closed set and the definition of $\mathcal{P}$ given at the beginning of Section 2.4.

Lemma 4.1 *Let (H, C, K) be a partially weighted graph. Given a graph G together with an (H, G)-list L and a closed set R, let $\widetilde{G}$ be the* tribal *graph associated to $(G, R, \mathcal{P})$. Then, there is a surjective function $\mathcal{M} : \mathcal{H}(G, L) \to \mathcal{H}(\widetilde{G}, L)$.*

Proof. We first associate to each list (H, C, K)-coloring χ of (G, L) a list (H, C, K)-coloring $\mathcal{M}(\chi) = w_\chi$ of $(\widetilde{G}, L)$.

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

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

For $v \in I(\widetilde{G})$ and $a \in V(H)$,

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

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 Section 11). Furthermore, for $v \in T(\widetilde{G})$, we have that $1 \leq \sum_{a \in V(H)} w(v, a)$. Also, 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 $\sum_{a \in V(H)} w(v, a) \leq \min\{|P_v|, k + s\} = p(v)$, thus 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 $(\widetilde{G}, L)$ 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 $(\widetilde{G}, L)$. We define a mapping

$$\chi : V(G) \to V(H)$$

as follows: for every vertex $v \in I(\widetilde{G})$, 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 $\{A(v, a) \mid a \in V(H)\}$, so that, first $|A(v, a)| = w(v, a)$ and second, $\cup_{a \in V(H)} A(v, a) = P_v$, notice that some of them may be empty. 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 the sum $\sum_{a \in C} w(v, a) = k + s > k$. Therefore, there is a $b \in S$ such that $w(v, b) = 1$. We partition P_v into h disjoint sets $\{A(v, a) \mid a \in V(H)\}$ and a set B, such 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 filled by some representative of its tribe, and that both vertices have the same neighborhood and the same list, it follows that the χ defined is a list (H, C, K)-coloring of (G, L). Furthermore, it is straightforward to check that $\chi_w = w$. □

4.2 Computing $\widetilde{G}$

In this section, we present a generic algorithm for computing the tribal graph $\widetilde{G}$ and analyze its complexity. We provide an implementation of the function Tribal that takes as input an instance of the list (H, C, K)-coloring and a closed set R such that $\overline{R}$ is (l, t)-representable, and it produces a structure corresponding to the tribal graph $\widetilde{G}_{k+s}$ associated to G with individuals R. We associate to each vertex $v \in V(\widetilde{G})$ a data structure formed by a boolean array of size h holding a representation of $L(v)$, an integer to keep $p(v)$, and a circular list formed by the neighbors of v.

Function Tribal uses a binary rooted tree that has a leaf for each possible equivalence class containing an element in $\overline{R} \subseteq V(G)$. Each leaf holds a counter. Each tree edge holds a label. In the path from the root to a leaf, the sequence of labels is split in two parts. The first part corresponds to the sorted list of numbers that represent the elements in $\mathcal{N}(\overline{R})$, according to the (l, t)-representation. The second part is a binary string with length h that represents the subset of H corresponding to the list of the vertices in the class. We refer

to those binary strings as the *mask* of the corresponding set. The array X keeps tracks of the vertices in G that will be used as representatives of the corresponding classes in $\tilde{G}$.

```
function Tribal(H, C, K, G, L, R, A, B)
    begin
        Set T = ∅, U = NIL
        for all v ∈ N do
            if v ∈ R then X[v] = 1 else X[v] = 0 end if
        end for
        for all v ∉ R do
            α = Neigh(G, v, A);
            β = Mask(v, G, L);
            if Is(T, α, β)
                then Increment(T, α, β, k + s)
                else Insert(T, α, β, 1); Insert(U, v, α, β); X[v] = 1
            end if
        end for;
        GTR = Graph(U ∪ R);
        while Next(U) ≠ NIL do
            (u, α, β) = Next(U);
            Create u's adjacency list from α, X and B
            AddW(GTR, u, Value(T, α, β))
        end for;
        for all u ∈ R do
            Create u's adjacency list from G, X and B
            AddW(GTR, u, 1)
        end for;
        return GTR
    end
```

Function Neigh implements the construction of a representation α of the neighborhood of a vertex not in R by a sorted list of numbers in time t. Function Mask constructs the string β representing the set $L(v)$.

We use the following functions on the tree data-structure. Function Is determines whether a vertex equivalent with v has been inserted in the tree T. Function Inserts inserts a leaf in the binary search tree, identified by (α, β), with weight 1. Function Increment increments by one the value hold in the specified leaf provided it does not overpass $k + s$. Function Value returns the value held at the leaf determined by (α, β).

Function Graph creates a graph without edges on the specified vertices of G, maintaining the same assigned list $L(u)$ to the nodes. Function AddW assigns the given weight to the vertex.

The next result is obtained taking into account the previous description of the function Tribal and the fact that $\overline{R}$ is (l,t)-representable.

Lemma 4.2 *Let (H,C,K) be a partially weighted graph. Given a graph G together with a (H,G)-list L and a closed set R such that $\overline{R}$ is (l,t)-representable we have that the function* Tribal *computes the tribal graph $\widetilde{G}$ associated to (G,R) in $O(n(t+l+h+r_{\geq k})+N(l+h)+kr_{<k})$ steps, where $N=|\mathcal{N}(\overline{R})|$ and $r_{\geq k}$ and $r_{<k}$ are, respectively the vertices with degree k or more or the vertices with degree less than k in R.*

4.3 Property (4) of the compactor

We show how to compute the number of extensions of a given list (H,C,K)-coloring w of $(\widetilde{G},L)$ to a list (H,C,K)-coloring of (G,L). According to the results in Lemma 4.1 we want to compute

$$W(w,G)=|\mathcal{M}^{-1}(w)|=|\{\chi \mid w_\chi = w\}|.$$

We make use of 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 $\tau=\sum_{a\in C} w(v,a)$.

Precomputing the values $1,\ldots,k!$ and using repeated squaring we get the following result.

Lemma 4.3 *Let (H,C,K) be a partially weighted graph. Given a graph G together with an (H,G)-list L and a closed set R, let $\widetilde{G}$ be the* tribal *graph associated to $(G,R,\mathcal{P})$ and let w be a list (H,C,K)-coloring of $(\widetilde{G},L)$. Then, for any $v\in T(\widetilde{G})$ and given w, $\widetilde{G}$ and $|P_v|$, the value $W(w,v)$ can be computed in $O(k+\log n)$.*

The second step is to show how to compute $W(w,G)$ using $W(w,v)$.

Lemma 4.4 *Let (H,C,K) be a partially weighted graph. Given a graph G together with an (H,G)-list L and a closed set R, let $\widetilde{G}$ be the* tribal *graph associated to $(G,R,\mathcal{P})$ and let w be a list (H,C,K)-coloring of $(\widetilde{G},L)$. Then,*

$$W(w,G)=\prod_{v\in T(\widetilde{G})} W(w,v).$$

Proof. For any $v\in T(\widetilde{G})$ and for any list (H,C,K) coloring χ that extends w, it holds that for any $a\in C$, $|\chi^{-1}(a)\cap P_v|=w(v,a)$. Furthermore, for any

$a \in S$ with $w(v,a) = 1$, $|\chi^{-1}(a) \cap P_v| \geq 1$. To fulfill these conditions, we have to select, in all possible ways, the elements that fill C with elements in P_v, which gives the factor

$$\binom{|P_v|}{\tau} \frac{\tau!}{\prod_{a \in C} w(v,a)!},$$

where $\tau = \sum_{a \in C} w(v,a)$. Next, we have to choose one vertex to cover any of the selected places in S, which gives the additional factor

$$\binom{|P_v| - \tau}{\sigma} \sigma!,$$

where $\sigma = \sum_{a \in S} w(v,a)$. The remaining members of the tribe can be placed in any of the σ positions in S, giving an additional factor of $\sigma^{|P_v|-\tau-\sigma}$. The lemma follows noticing that the total number of extensions is the product of the number of extensions for each particular tribe. $\square$

Observe that if we can show that the number of nodes in $\widetilde{G}$ is small (it depends only on k and h), then the previous results implies that condition (4) in the definition of compactor holds.

Now we provide an implementation of the function Count-extent that takes as input an instance of the list (H, C, K)-coloring and a tribal graph and computes $|\mathcal{H}(G, L)|$.

```
function Count-exten(H, C, K, G, L, GTR)
    begin
        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
```

Observe that function Count-extent is a brute force algorithm that considers all possible tribal mappings. For $w \in \mathcal{H}(G, L)$, Count-extent computes the number of extensions. The algorithm returns the overall sum of all those quantities.

Lemma 4.5 *Let (H, C, K) be a partially weighted graph. Given a graph G together with an (H, G)-list L and a closed set R, let $\widetilde{G}$ be the tribal graph associated to G and R. Then, algorithm Count-exten outputs the number of*

list (H, C, K)-colorings of (G, L) within

$$O\left(\left(\widetilde{n}\log n + h^2\widetilde{m} + (h+k)\widetilde{n}\right)k^{(h+1)\widetilde{n}}\right)$$

steps.

Proof. Consider the function Count-exten, that given the associated tribal graph, computes the number of list (H, C, K)-coloring of the graph G. Its correctness follows from Lemmata 3.1, 4.1 and 4.3.

From Lemma 3.1 we know that the number of candidate mappings of $V(\widetilde{G}) \times V(H)$ in $[k]$ is at most $k^{(h+1)\widetilde{n}}$. We can check in time $O(h^2\widetilde{m}+h\widetilde{n})$ whether $w \in \mathcal{H}(\widetilde{G}, L)$. Furthermore, the candidate mappings can be generated in $O(\widetilde{n}+h)$ time per mapping. According to Lemma 4.3, the computation of the number of extensions of a particular coloring can be done in time $O((k+\log n)\widetilde{n}$. □

4.4 Counting list (H, C, K)-colorings

Combining all the pieces in algorithm Count-List-Colorings and using Lemmata 2.3, 2.4, 2.5, 3.2, 4.2, and 4.5 we set the following three counting results,

Theorem 4.6 *Let (H, C, K) be a partially 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, the adaptation of algorithm Count-List-Colorings outputs the number of list (H, C, K)-colorings of (G, L) within time*

$$O\left((k+h)n + 2^{k+h}k^{(h+1)(k^2+2k+2^{k+h})}\log n + kh^2 2^{k+h}k^{(h+1)(k^2+2k+2^{k+h})}\right).$$

Theorem 4.7 *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 $G' = (V(G), \emptyset)$, let (N_1, N_2) be the list-splitting of (G', L), and let $\widetilde{G}$ be the tribal graph associated to (G', N_1). Then, the adaptation of algorithm Count-List-Colorings outputs the number of list (H, C, K)-colorings of (G, L) within time*

$$O\left(hn + (k+2^h)k^{(h+1)(k+2^h)}\log n + (h+k)(k+2^h)k^{(h+1)(k+2^h)}\right).$$

Theorem 4.8 *Let (H, C, K) be a partially weighted graph with $H = K_{xy}$. Given a connected bipartite graph G together with a (H, G)-list L, let $G' = (X(G), Y(G), X(G) \times Y(G))$, let (M_1, M_2, M_3) be the bipartite list splitting of (G', L), and let $\widetilde{G}$ be the tribal graph associated to $(G', M_1 \cup M_2)$. Then, the adaptation of algorithm Count-List-Colorings outputs the number of list (H, C, K)-colorings of (G, L) within time*

$$O\left((h+k)n + (k+2^{h+1})k^{(h+1)(k+2^{h+1})}\log n + (h^2k^2 2^{2h})k^{(h+1)(k+2^{h+1})}\right).$$

5 Counting list (H, C, K)-colorings for simple weight assignments

In this section we provide a fixed parameter algorithm for counting the number of list (H, C, K)-coloring when the connected components of H are 1, 2 or 3-components. The graph H is decomposed as the disjoint union of γ components, $H_{-\rho}, \ldots, H_{\eta}$, for $\rho, \eta \in \{0, \ldots, h\}$, 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.

Notice, 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 $\{G_1, \ldots, G_g\}$. Given a (H, G)-list L, we often have to speak of a list coloring of a component G_i, we 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 $\widetilde{G}_{i\iota}$ be relabeled so that if $n_{i\iota} = |V(\widetilde{G}_{i\iota})|$, then $V(\widetilde{G}_{i\iota}) = [n_{i\iota}]$. Furthermore, we assume that for any i, $1 \leq i < n_{i\iota}$, $p_{i\iota}(i) \leq p_{i\iota}(i+1)$.

For any $i \in [g]$, $\iota \in [-\rho, \eta]$, and $w : [n_{i\iota}] \times V(H_\iota) \to [k]$ set $\beta(i, \iota, w) = 1$ if w is a list H_ι-coloring of $\widetilde{G}_{i\iota}$, otherwise $\beta(i, \iota, w) = 1$.

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

$$\begin{aligned} i \sim_\iota j \text{ iff } & n_{i\iota} = n_{j\iota} \\ & \text{and } \forall v \in [n_{i\iota}] p_{i\iota}(v) = p_{j\iota}(v) \\ & \text{and } \forall\, w : [n_{i\iota}] \times V(H_\iota) \to [k] \;\; \beta(i, \iota, w) = \beta(j, \iota, w). \end{aligned}$$

We say that two components of G are *equivalent* if they are equivalent 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 $i \in [g]$, let Q_i denote the equivalence class of i.

Let $\mathcal{Q}$ be the partition induced in $[g]$ by $\sim$. Taking into account that the number of vertices $\widetilde{n}$ that appears in one tribal graph is bounded by $(k + h)(k^2 + 2k + 2^{k+h})$ (the case of 1-components), we have at most

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

equivalence classes. Each class can be represented by an integer

$$z^Q = min\{|Q|, k^{(k+h)(k^2+2k+2^{k+h})}\}$$

and a sequence of γ triples $(n_\iota^Q, p_\iota^Q, \beta_\iota^Q)$, formed by an integer, a tribal weight, and a binary string of length at most $k^{h(k+s)(k^2+2k+2^{k+h})}$, representing respectively, the size of the vertex set, the size of the tribes, and the associated mask.

We need further definitions to establish the form and properties of the compactor. Recall that when w is a list (H, C, K)-coloring of a tribal graph $\widetilde{G}$, for any $c \in C$, $\pi(w, c) = \sum_{v \in V(\widetilde{G})} w(v, c)$. For any $Q \in \mathcal{Q}$ and $\iota \in [-\rho, \eta]$, define

$$B_\iota^0(Q) = \{w : [n_\iota^Q] \times V(H_\iota) \to [k] \mid \beta_\iota^Q(w) = 1 \text{ and } \forall c \in C\ \pi(w, c) = 0\}$$
$$B_\iota^1(Q) = \{w : [n_\iota^Q] \times V(H_\iota) \to [k] \mid \beta_\iota^Q(w) = 1 \text{ and } \exists c \in C\ \pi(w, c) > 0\}$$

and the classes of functions

$$\mathcal{F}_0(Q) = \bigcup_{\iota \in [-\rho, \eta]} B_\iota^0(Q) \text{ and}$$
$$\mathcal{F}_1(Q) = \bigcup_{\iota \in [-\rho, \eta]} B_\iota^1(Q).$$

Finally, define

$$\begin{aligned}\mathcal{F}(G, L) = \{((A_Q, o_Q, B_Q))_{Q \in \mathcal{Q}} \mid\ & \forall Q \in \mathcal{Q} \\ & A_Q \subseteq \mathcal{F}_1(Q) \text{ and } o_Q : A_Q \to [k] \\ & \text{and } B_Q \subseteq \mathcal{F}_0(Q) \\ & \text{and } \forall a \in C \sum_{Q \in \mathcal{Q}} \sum_{w \in A_Q} o_Q(w)\, \pi(w, a) = K(a) \\ & \text{and } |Q| \geq |B_Q| + \sum_{w \in A_Q} o_Q(w)\}\end{aligned}$$

In the next result we prove that the set $\mathcal{F}(G, L)$ satisfies properties (1)–(3) of the definition of compactor.

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 there is a surjective mapping from $\mathcal{H}(G, L)$ into $\mathcal{F}(G, L)$. Furthermore, $|\mathcal{F}(G, L)| \leq \mathsf{sz}(k, h)$ and the elements of $\mathcal{F}(G, L)$ can be enumerated using time $O(\mathsf{tm}(k, h))$ per element, for suitably selected functions sz and tm.*

Proof. First, to each list (H, C, K)-coloring χ of (G, L) we associate a member of $\mathcal{F}(G, L)$. Given a list (H, C, K)-coloring χ of (G, L), set χ_i to be the restriction of χ to G_i. Set $\chi(i) \in [-\rho, \eta]$ be such that $\chi(G_i) \subseteq H_{\chi(i)}$. Let w_i be the coloring of $\widetilde{G}_{i\chi(i)}$ in $H_{\chi(i)}$ constructed from χ_i according to Lemma 4.1 together with the suitable relabeling of the vertices.

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

$$\begin{aligned}\chi(Q) &= \{w_i \mid i \in Q\},\\ \chi^+(Q) &= \{w_i \in \chi(Q) \mid \sum_{c\in C} \pi(w_i, c) \geq 0\} \text{ and}\\ \chi^-(Q) &= \{w_i \in \chi(Q) \mid \sum_{c\in C} \pi(w_i, c) = 0\}.\end{aligned}$$

For any $w \in \chi^+(Q)$ set $o_Q(w) = |\{w_i = w \mid i \in Q\}|\}$. 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. Notice that the last condition in the definition of the elements in $\mathcal{F}(G, L)$ insures the existence of sufficient components in Q to get at least one element for each 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 4.1, and let φ be its disjoint union. From the definition of $\mathcal{F}(G, L)$, and Lemmata 4.1, 2.1, and 2.2, we get that for all $a \in C$, $|\varphi^{-1}(a)| = k(a)$. Furthermore, φ is a list H-coloring of $(\tilde{G}, L)$ as a consequence of the definition of equivalence between components and its definition.

Finally notice that, from the bound on the number of classes, the fact that the number of elements in any tribal graph associated to a component is bounded by a function of k, and the fact that each tribal coloring has range $[k]$, we have that $\mathcal{F}(G, L)$ has size bounded by a function of k and h. The algorithmic cost of generating the last element comes from standard techniques: we combine the generation of all the subsets of functions and all the combinations of tuples and then check for the properties that define those elements in $\mathcal{F}(G, L)$. As the size of the representation is bounded by a function of k and h, the claim follows. □

It remains to show property (4) of the compactor. We want to compute, for any sequence $\sigma \in \mathcal{F}(G, L)$, the number of list (H, C, K)-colorings χ of (G, L), such that $\sigma_\chi = \sigma$. Observe that this computation can be done as the product of the number of extensions of the triple associated to each class Q.

Assume that $\sigma \in \mathcal{F}(G, L)$ and fix an equivalence class Q. We consider first all

the ways to assign the colorings represented by the triple (A_Q, o_Q, B_q) to all the connected components of G present in Q. Once we have one such assignation we wish to compute the number of colorings of a component $G_i \in Q$ that the assigned tribal coloring represents. Notice that the same tribal coloring w used as coloring for a $\widetilde{G}_{i\iota}$, can be extended in a different number ways to a coloring of G_i than when considered as a coloring for $\widetilde{G}_{j\iota}$.

We give a recursive definition that allows us to compute the number of extensions of the colorings represented by (A_Q, o_Q, B_q). We have to define an assignment of components in Q to functions in $A_Q \cup B_Q$, satisfying the following two properties: any coloring in $w \in A_Q$ must be used $o_Q(w)$ times and any coloring $w \in B_Q$ must be used at least once. To simplify the presentation we replace A_Q by a multiset A^Q in which any function $w \in A_Q$ appears $o_Q(w)$ times. We also make use of the function $W(w, G)$, as defined in Lemma 4.3. Without loss of generality we assume that $Q = \{G_1, \ldots, G_{|Q|}\}$. We work on triples (r, M, N) where $0 \leq r \leq |Q|$ represents the first r connected component of G in the class Q, $M \subseteq A^Q$ (a multiset), and $N \subseteq B_Q$. We define $T(r, M, N)$ recursively as follows:

$$T(0, M, N) = \begin{cases} 1 & \text{if } M = N = \emptyset, \\ 0 & \text{otherwise,} \end{cases}$$

and, for $r > 0$,

$$\begin{aligned} T(r, M, N) = &\sum_{w \in M} T(r-1, M - \{w\}, N) W(w, G_r) \\ &+ \sum_{w \in N} T(r-1, M, N) W(w, G_r) \\ &+ \sum_{w \in N} T(r-1, M, N - \{w\}) W(w, G_r). \end{aligned}$$

Lemma 5.2 *Let (H, C, K) be a simple partial weight assignment. Given a graph G together with a (H, G)-list L and a set (A_Q, o_Q, B_Q) of colorings which can be used on the components in class Q, then $T(|Q|, A^Q, B_Q)$ is the number of colorings χ of $(G_i)_{i \in Q}$ for which $\sigma_\chi = ((A_Q, o_Q, B_Q))$.*

Proof. The correctness follows from the shape of the tribal colorings. If we take an element in the compactor, any (H, C, K)-colorings of G represented by this element is obtained by an adequate selection of the mapping associated to a connected component of G. In this selection, the last connected component of G must be mapped using one of the allowed colorings. If the mapping belongs to the set B_Q, we have two cases, either the mapping has been used previously, which corresponds to the second term in the recurrence, or or the mapping has not been used previously, which corresponds to the third term in the recurrence. Finally, observe that the number of mappings is the product of the number of extensions of the tribal colorings. □

The function $T(r, M, N)$ computes the number of ways of extending a particular element of the sequence $\sigma \in \mathcal{F}(G, L)$ to a list coloring. Taking into account the disjointness of the classes in the partition of connected components, the result can be used to compute the total number of colorings that extend σ.

Theorem 5.3 *Let (H, C, K) be a simple partial weight assignment. Given a graph G together with a (H, G)-list L,*

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

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.

Proof. The computation of the number of list (H, C, K)-colorings for simple partially weighted graph follows three main phases:

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

$$\sum_{i=1}^{g} \gamma n_i(h + k) + \gamma(k^3 + k2^{k+h}) = \gamma(h + k)n + g(hk^3 + hk2^{k+h})$$

steps. In the above formula we have taken the most costly computation for the 1-component.

(2) The second step is to compute the values of the β function. At the end of the phase we will end up with information for each class $Q \in \mathcal{Q}$: the value $|Q|$ and its mask, and the sets $\mathcal{F}_0(Q)$ and $\mathcal{F}_0(Q)$. All those computations can be done using only the graphs $\widetilde{G}_{i\iota}$. 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 k and h.

(3) The last phase is implemented by an enumeration of $\mathcal{F}(G, L)$. For each element and class, we use dynamic programming to compute a tabke holding all the values $T(r, M, N)$. Notice that the size of the table is a function of the parameters. Therefore the unique cost that depends on n is the computation of the function W that requires $\log n$ time. Thus, the overall contribution is $f_2(k, h) \log n$ plus some additive factors that only depend on the parameters k and h.

□

6 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. We can redefine all the parameterized

versions in the same way, replacing the notion of (H, C, K)-coloring for that of $(H, C, \leq K)$-coloring as it was done in [6]. 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 the same way we consider the $\#(H, C, \leq K)$-coloring and the list $\#(H, C, \leq K)$-coloring problems corresponding to their counting versions.

In a similar way, we can define the $\leq$-*equivalent* denoted by $(H, C, K) \sim_{\leq} (H', C', K')$. We use $\mathcal{H}^{\leq}_{(H,C,K)}(G)$ to denote the set of $(H, C, \leq K)$-colorings of graph G, and $\mathcal{H}^{\leq}_{(H,C,K)}(G, L)$ to denote the set of list $(H, C, \leq K)$-colorings of a graph G and an (H, G)-list L.

Let us call algorithm Count-List-Colorings-leq the variation of the algorithm Count-List-Colorings in which the condition $w \in \mathcal{H}(\widetilde{G}_{k+s}, L)$ is replaced by the condition $w \in \mathcal{H}^{\leq}(\widetilde{G}_{k+s}, L)$. Observe that, as we were looking to all the possible mappings, this includes the ones that do not cover completely the weighted vertices. Furthermore, the time bounds are the same as before.

By replacing the corresponding line in the derived counting algorithms and using the adequate compactors, we can design a counting algorithm for the cases in which G is connected and (H, C, K) is simple.

Theorem 6.1 *Let (H, C, K) be a partially weighted graph. Given a connected graph G and a (H, G)-list L, then the adaptation of algorithm Count-List-Colorings-leq outputs the number of list $(H, C, \leq K)$-colorings of (G, L) within time*

- $O\left((k+h)n + 2^{k+h}k^{(h+1)(k^2+2k+2^{k+h})}\log n + kh^2 2^{k+h}k^{(h+1)(k^2+2k+2^{k+h})}\right)$ *when $H - C$ is edgeless.*
- $O\left(hn + (k+2^h)k^{(h+1)(k+2^h)}\log n + (h+k)(k+2^h)k^{(h+1)(k+2^h)}\right)$ *when $H = K^r_h$.*
- $O\left((h+k)n + (k+2^{h+1})k^{(h+1)(k+2^{h+1})}\log n + (h^2k^2 2^{2h})k^{(h+1)(k+2^{h+1})}\right)$ *when $H = K_{xy}$.*

Finally, we redefine the compactor $\mathcal{F}^{\leq}(G, L)$ for the case of simple partial weight assignments. We use the notation given in Section 4 and reestate the unique change in the definition of compactor.

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

Then, using similar arguments as those in Lemma 4.1 we have

Lemma 6.2 *Let (H, C, K) be a simple partially weighted graph. Given a graph*

G together with a (H,G)-list L, then there is a surjective mapping from the set $\mathcal{H}^{\leq}(G,L)$ into the set $\mathcal{F}^{\leq}(G,L)$.

Finally, taking into account that the structure of the assignment of list-$(H,C,\leq K)$ coloring of the tribal graph to the components is similar, using the same dynamic programming schema as in Theorem 5.3, we get

Theorem 6.3 *Let (H,C,K) be a simple partial weight assignment. Given a graph G together with a (H,G)-list L, $|\mathcal{H}^{\leq}(G,L)|$ 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.*

7 Conclusions and open problems

We have shown that for the family of simple partially weighted graphs the list (H,C,K)-coloring and the list $\#(H,C,K)$-coloring problems each have an efficient fixed parameter algorithm. In doing so, we have introduced the notion of compactor and provided a novel technique to develop efficient fixed parameter algorithms for counting problems.

Two lines of future research can be highlighted. The first one is to explore the possibility of designing algorithms for the counting version of other parameterized problems (for recent work in this direction, see [27]). The second one is to continue in the analysis of the parameterized complexity of further families of partially weighted graphs.

For the second line of research it is worth mentioning the way in which we obtained the class of simple partially weighted graphs. We started from the characterization of the graphs H for which the $\#H$-coloring is solvable in polynomial time. According to [16] those are the graphs whose connected components are either a complete reflexive graph or a complete bipartite irreflexive graph. The 2 and 3-components were obtained by fixing number of the preimages of some of the vertices in such components. The 1-components were obtained from a complete bipartite graph, allowing the removal of some edges joining restricted vertices to non restricted vertices and the addition of edges between restricted vertices. For the case of 2-components, removing loops in the restricted part allows to capture problems like independent set which is not in FPT (see Figure 2). It remains open to explore whether the removal of some loop edges in the non restricted part gives rise to more FPT counting problems. Observe that this extended 2-component includes the fourth partially weighted graph given in Figure 2 (A-BIP) that describes the k-almost bipartite subgraph problem whose decision version belongs to FPT [28]. It remains also open to study the effect on the 3-components of the addition and removal of edges in the parameterized complexity of the corresponding prob-

lems. To this latest class belongs the fifth partially weighted graph given in Figure 2 (A-3C), which is obtained adding edges between the parameterized and non parameterized part of a 3-component. As already mentioned before, the parameterized complexity of the almost 3-coloring problem also remains open.

Acknowledgement

We thank an anonymous referee for the careful comments which allowed us to improve the presentation of this paper.

References

[1] K. A. Abrahamson, R. G. Downey, and M. R. Fellows. Fixed-parameter tractability and completeness. IV. On completeness for W[P] and PSPACE analogues. *Annals of Pure and Applied Logic*, 73(3):235–276, 1995.

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

[3] T.H. Cormen, C.E. Leisserson, R.L. Rivest, and C. Stein *Introduction to Algorithms*, 2nd ed. MIT Press and McGraw Hill, 2001

[4] 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*. Lecture Notes in Computer Science, volume 2136, pages 304–315, Springer, 2001.

[5] 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.

[6] J. Díaz, M. Serna, and D. Thilikos. Fixed parameter 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.

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

[8] J. Díaz, M. Serna, and D. M. Thilikos. Efficient algorithms for parameterized H-colorings. In M. Klazar, J. Kratochvil, M. Loebl, J. Matousek, R. Thomas, and P. Valtr, editors. *Topics in Discrete Mathematics*, pages 373–406. Algorithms and Combinatorics 26, Springer, 2006.

[9] J. Díaz, M. Serna, and D. M. Thilikos. Complexity issues on Bounded Restrictive H-colorings. *Discrete Mathematics*, 307(16):2082-2093, 2007.

[10] R. Downey and M. Fellows. Fixed-parameter tractability and completeness. III. Some structural aspects of the W hierarchy. In *Complexity theory*, pages 191–225. Cambridge Univ. Press, Cambridge, 1993.

[11] R. G. Downey and M. R. Fellows. Fixed-parameter tractability and completeness. In *Proceedings of the Twenty-first Manitoba Conference on Numerical Mathematics and Computing (Winnipeg, MB, 1991)*, volume 87, pages 161–178, 1992.

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

[13] 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.

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

[15] M. Dyer and C. Greenhill. Corrigendum: The complexity of counting graph homomorphisms. *Draft*, 2004.

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

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

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

[19] H. Fernau. Parameterized Algorithmics: A Graph-Theoretic Approach. Habilitationsschrift (submitted). Universität Tübingen, April 2005.

[20] J. Flum and M. Grohe. The parameterized complexity of counting problems. *SIAM Journal on Computing*, 33(4):892–922,2004.

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

[22] P. Hell and J. Nešetřil. Counting list homomorphisms and graphs with bounded degrees. In J. Nešetřil and P. Winkler, editors, *Graphs, Morphisms and Statistical Physics*, DIMACS series in Discrete Mathematics and Theoretical Computer Science, vol 63, pp. 105–112. American Mathematical Society, 2004.

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

[24] Rolf Niedermeier. *Invitation to fixed-parameter algorithms*, volume 31 of *Oxford Lecture Series in Mathematics and its Applications*. Oxford University Press, Oxford, 2006.

[25] C. McCartin. Parameterized counting problems. In K. Diks and W. Rytter, editors, *Mathematical Foundations of Computer Science 2002, 27th International Symposium (MFCS 2002)*. Lecture Notes in Computer Science, volume 2420, pages 556–567, Springer, 2002.

[26] C. McCartin. *Contributions to Parameterized Complexity*. PhD Thesis, Victoria University, Wellington, 2003.

[27] N. Nishimura, P. Ragde and D. M. Thilikos. Parameterized counting algorithms for general graph covering problems. In F. K. H. A. Dehne, A. Lpez-Ortiz, J. R. Sack, editors, *Algorithms and Data Structures, 9th International Workshop (WADS 2005)*. Lecture Notes in Computer Science, volume 3608, pages 99–109, Springer, 2005.

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