Parameterized Complexity of Happy Coloring Problems[*]

Akanksha Agrawal[†1], N. R. Aravind[2], Subrahmanyam Kalyanasundaram[2], Anjeneya Swami Kare[‡3], Juho Lauri[4], Neeldhara Misra[5], and I. Vinod Reddy[§6]

[1]Ben-Gurion University of the Negev, Israel
[2]Indian Institute of Technology Hyderabad, India
[3]University of Hyderabad, India
[4]Nokia Bell Labs, Ireland
[5]Indian Institute of Technology Gandhinagar, India
[6]Indian Institute of Technology Bhilai, India

Abstract

In a vertex-colored graph, an edge is *happy* if its endpoints have the same color. Similarly, a vertex is *happy* if all its incident edges are happy. Motivated by the computation of homophily in social networks, we consider the algorithmic aspects of the following Maximum Happy Edges (k-MHE) problem: given a partially k-colored graph G and an integer ℓ, find an extended full k-coloring of G making at least ℓ edges happy. When we want to make ℓ vertices happy on the same input, the problem is known as Maximum Happy Vertices (k-MHV). We perform an extensive study into the complexity of the problems, particularly from a parameterized viewpoint. For every $k \geq 3$, we prove both problems can be solved in time $2^n n^{O(1)}$. Moreover, by combining this result with a linear vertex kernel of size $(k+\ell)$ for k-MHE, we show that the edge-variant can be solved in time $2^\ell n^{O(1)}$. In contrast, we prove that the vertex-variant remains W[1]-hard for the natural parameter ℓ. However, the problem does admit a kernel with $O(k^2\ell^2)$ vertices for the combined parameter $k+\ell$. From a structural perspective, we show both problems are fixed-parameter tractable for treewidth and neighborhood diversity, which can both be seen as sparsity and density measures of a graph. Finally, we extend the known NP-completeness results of the problems by showing they remain hard on bipartite graphs and split graphs. On the positive side, we show the vertex-variant can be solved optimally in polynomial-time for cographs.

1 Introduction

Analyzing large networks is of fundamental importance for a constantly growing number of applications. In particular, how does one mine e.g., social networks to provide valuable insight? A basic observation concerning the structure of social networks is *homophily*, that is, the principle that we tend to share characteristics with our friends. Intuitively, it seems believable our friends are similar to us in terms of their age, gender, interests, opinions, and so on. In fact, this observation is well-known in sociology (see e.g., [16, 27, 26]). For example, imagine a network of supporters in a country with a two-party system. In order to check whether there is homophily by political stance (i.e., a person tends to befriend a person with similar political beliefs), we could count the number of edges between two people of opposite beliefs. If there were no such edges, we would observe homophily in an extreme sense. It is characteristic of social networks that they evolve over time: links tend to be added between people that share some characteristic. But given a snapshot of the network, how extensively can homophily be present? For instance, how far can an extreme ideology spread among people some of whom are "politically neutral"?

[*]The present manuscript is a combination of the works [1, 3, 28], two of which appeared in the proceedings of the 28th International Workshop on Combinatorial Algorithms (IWOCA).
[†]This work was done while the author was a Ph.D. student at University of Bergen, Norway.
[‡]This work was done while the author was a Ph.D. student at IIT Hyderabad, India.
[§]This work was done while the author was a Ph.D. student at IIT Gandhinagar, India.

We abstract these questions regarding the computation of homophily as follows. Consider a vertex-colored graph $G = (V, E)$. We say an edge is *happy* if its endpoints have the same color (otherwise, the edge is *unhappy*). Similarly, a vertex is *happy* if it and all its neighbors have the same color (otherwise, the vertex is *unhappy*). Equivalently, a vertex is happy when all of its incident edges are happy. Let $S \subseteq V(G)$, and let $c : S \to [k]$ be a partial vertex-coloring of G. A coloring $\tilde{c} : V(G) \to [k]$ is an *extended full coloring* of c if $\tilde{c}|_S = c$, i.e, $\tilde{c}(v) = c(v)$ for all $v \in S$. In this work, we consider the following coloring problems.

Weighted Maximum Happy Edges (Weighted MHE)
Input: A graph G, integers k, a vertex subset $S \subseteq V(G)$, (partial) coloring $c : S \to [k]$, and a weight function $w : E(G) \to \mathbb{N}$.
Output: A coloring $\tilde{c} : V(G) \to [k]$ such that $\tilde{c}|_S = c$ maximizing the total weight of the happy edges.

Weighted Maximum Happy Vertices (Weighted MHV)
Input: A graph G, integers k, a vertex subset $S \subseteq V(G)$, (partial) coloring $c : S \to [k]$, and a weight function $w : V(G) \to \mathbb{N}$.
Output: A coloring $\tilde{c} : V(G) \to [k]$ such that $\tilde{c}|_S = c$ maximizing the total weight of the happy vertices.

Less generally, we typically consider the unweighted variants of the problems obtained by letting the weight of each edge or vertex be one. We refer these problems as Maximum Happy Edges (MHE) and Maximum Happy Vertices (MHV) respectively. Moreover, when k is fixed and not part of the input, we refer the weighted variants of the problems as Weighted k-MHE and Weighted k-MHV and the unweighted variants of the problems as k-MHE and k-MHV respectively.

1.1 Previous Work

Zhang and Li [30] proved that for every $k \geq 3$, the problems k-MHE and k-MHV are NP-complete. However, when $k = 2$, they gave algorithms running in time $O(\min\{n^{2/3}m, m^{3/2}\})$ and $O(mn^7 \log n)$ for 2-MHE and 2-MHV, respectively. Towards this end, the authors used max-flow algorithms (2-MHE) and minimization of submodular functions (2-MHV). Moreover, the authors presented approximation algorithms with approximation ratios 1/2 and $\max\{1/k, \Omega(\Delta^{-3})\}$ for k-MHE and k-MHV, respectively, where Δ is the maximum degree of the graph. Later on, Zhang, Jiang, and Li [29] gave improved algorithms with approximation ratios 0.8535 and $1/(\Delta+1)$ for k-MHE and k-MHV, respectively. In [2], a subset of the current authors proved that both problems are solvable in polynomial time for trees.

Perhaps not surprisingly, the happy coloring problems are tightly related to cut problems. Indeed, the k-MHE problem is a generalization of the following Multiway Uncut problem [25]. In this problem, we are given a weighted undirected graph $G = (V, E)$ and a terminal set $S = \{s_1, s_2, \ldots, s_k\} \subseteq V(G)$. The goal is to find a partition of $V(G)$ into classes $V_1, \ldots, V_k$ such that each class contains exactly one terminal and the total weight of the edges not cut by the partition is maximized. We obtain the Multiway Uncut problem as a special case of k-MHE, when each color is used to precolor exactly one vertex. We also mention that the complement of the Multiway Uncut problem is the Multiway Cut problem that has been studied before (see e.g., [13, 11]). There are known (parameterized) algorithms for the Multiway Cut problem with the size of the cut ℓ as the parameter. In this regard, the fastest known algorithm runs in $O^*(1.84^\ell)$ time [10].

1.2 Our Results

We perform an extensive study of the complexity of the two problems, particularly from a parameterized perspective. For the summary below, we recall that we use k to denote the number of colors, and ℓ to denote the target total weight of the number of happy vertices.

- In Section 3, we consider exact exponential-time algorithms for the happy coloring problems. The naive brute force runs in $k^n n^{O(1)}$ time, but we show that for every $k \geq 3$, there is an algorithm running in time $O^*(2^n)$, where n is the number of vertices in the input graph. Moreover, we prove that this is not optimal for every k by giving an even faster $O^*(1.89^n)$-time algorithm for both 3-MHE and 3-MHV.

- In Section 4, we show that the decision variant of WEIGHTED MHE admits a small kernel of size $k + \ell$. The ingredients of the kernel are a polynomial-time algorithm for WEIGHTED MHE when the uncolored vertices induce a forest together with simple reduction rules. By combining the algorithm of Section 3 with the kernel, we obtain an algorithm running in time $2^{\ell} n^{O(1)}$ for k-MHE.
- In Section 5 we prove that the decision variant of MHV is W[1]-hard for the natural parameter ℓ. On a positive side, we proceed to show in Section 6 that the problem does admit a kernel of size $O(k^2\ell^2)$ for the combined parameter $k + \ell$. The kernel is obtained using the annotation strategy (see e.g., [9]) and explicit reduction rules.
- In Section 7, we turn our attention to structural parameters. In particular, we consider two measures of sparsity and density, namely treewidth and neighborhood diversity. Our main result here is that both MHV and MHE are FPT parameterized by treewidth and the number of colors. Similarly, both problems are FPT parameterized by neighborhood diversity.
- In Section 8, we extend the known hardness results of the problems by proving that the decision variants of both MHE and MHV remain NP-complete for bipartite graphs and also for split graphs. On the positive side, we show MHV can be solved in polynomial-time for cographs.

2 Preliminaries

We denote the set of natural numbers by $\mathbb{N}$. For notational convenience, we write $[k]$ for $k \in \mathbb{N}$ to denote the set $\{1, 2, \ldots, k\}$. We use $-\infty$ to denote minus infinity and use the convention that for any $n \in \mathbb{N}$, we have $-\infty + n = -\infty$ and $-\infty + -\infty = -\infty$. Let $f : X \to Y$ be a function. For $y \in Y$, by $f^{-1}(y)$ we denote the set $\{x \in X \mid f(x) = y\}$. For $X' \subseteq X$, by $f|_{X'}$ we denote the function $f|_{X'} : X' \to Y$ such that $f_{X'}(x) = f(x)$, for all $x \in X'$. For an ordered set $R = X \times Y$, a function $f : R \to Z$, and an element $r = (x, y) \in R$, we slightly abuse the notation to denote $f(r) = f((x, y))$ by $f(x, y)$.

Sometimes, we write $f(n) = O^*(g(n))$ if $f(n) = O(g(n)\text{poly}(n))$, where $\text{poly}(n)$ represents any polynomial in n.

Graph Theory We use standard terminology from the book of Diestel [14] for graph-theoretical terms that are not explicitly defined here. For a graph G, by $V(G)$ and $E(G)$ we denote the vertex and edge sets of G, respectively. For a graph G and a vertex $v \in V(G)$, by $N_G(v)$ we denote the set $\{u \in V(G) \mid (v, u) \in E(G)\}$ and by $N_G[v]$ we denote the set $N_G(v) \cup \{v\}$. For $S \subseteq V(G)$, by $N_G(S)$ we denote the set $(\cup_{u \in S} N_G(u)) \setminus S$. We drop the subscript G from $N_G(v)$, $N_G[v]$ and $N_G(S)$ when the context is clear. For a vertex subset $S \subseteq V(G)$, by $G[S]$ we denote the subgraph of G induced by S, i.e., the graph with vertex set S and edge set $\{(u, v) \in E(G) \mid v, u \in S\}$. By $G - S$ we denote the graph $G[V(G) \setminus S]$.

For vertices $u, v \in V(G)$, *identifying* u and v in G results in the following graph G'. We have $V(G') = (V(G) \setminus \{u, v\}) \cup \{u^\star\}$ and $E(G') = E(G[V(G) \setminus \{u, v\}]) \cup \{(u^\star, w) \mid w \in (N_G(u) \cup N_G(v)) \setminus \{u, v\}\}$, where $u^\star$ is a vertex that is not in $V(G)$. Moreover, we refer to $u^\star$ as the resulting vertex after identification and the operation is said to identify u with v in G.

A *coloring* of a graph G with $k \in \mathbb{N}$ colors is a function $\varphi : V(G) \to [k]$. A partial coloring of G with k colors is a function $c : S \to [k]$, where $S \subseteq V(G)$. We will refer to a partial coloring as a coloring when the context is clear. A coloring $\tilde{c} : V(G) \to [k]$ is said to extend a partial coloring $c : S \to [k]$ if $\tilde{c}|_S = c$.

Parameterized Complexity A *parameterized problem* is a language $L \subseteq \Sigma^* \times \mathbb{N}$, where Σ is a fixed, finite alphabet. For an instance $(x, p) \in \Sigma^* \times \mathbb{N}$, we call p the *parameter*. The parameterized problem L is *fixed-parameter tractable* (FPT) when there is an algorithm $\mathcal{A}$, a computable function $f : \mathbb{N} \to \mathbb{N}$, and a constant r such that, given $(x, p) \in \Sigma^* \times \mathbb{N}$, the algorithm $\mathcal{A}$ correctly decides whether $(x, p) \in L$ in time bounded by $f(p) \cdot |x|^r$. An equivalent way of proving a problem is FPT is by constructing a *kernel* for it. A kernel for a parameterized problem (x, p) is a polynomial-time algorithm $\mathcal{B}$ that returns an equivalent instance (x', p') of L such that $|x'| \leq g(p)$, for some computable function $g : \mathbb{N} \to \mathbb{N}$. Here, we say two instances are *equivalent* if the first is a YES-instance iff the second is a YES-instance. Given a parameterized problem, it is natural to ask whether it admits a kernel, and moreover whether that kernel is small.[1] By small, we typically mean a polynomial kernel, or even a linear kernel (i.e., $g(p) = O(p)$).

Kernelization is often discovered through *reduction rules.* A reduction rule is a polynomial-time transformation of an instance (x, p) to another instance of the same problem (x', p') such that $|x'| < |x|$ and $p' \leq p$. A reduction rule is *safe* when the instances are equivalent. For more on parameterized complexity, we refer the interested reader to [12, 19].

Special Graph Classes. We now define some of the special graph classes considered in this paper. A graph is *bipartite* if its vertex set can be partitioned into two disjoint sets such that no two vertices in same set are adjacent. A graph is a *split graph* if its vertex set can be partitioned into a clique and an independent set. Split graphs do not contain C_4, C_5 or $2K_2$ as induced subgraphs. *Cographs* are P_4-free graphs, that is, they do not contain any induced paths on four vertices..

Definition 1. *A tree decomposition of a graph G is a pair $(\mathcal{T}, \mathcal{X} = \{X_t \mid t \in V(\mathcal{T})\})$, where an element $X \in \mathcal{X}$ is a subset of $V(G)$, called a bag, and $\mathcal{T}$ is a rooted tree satisfying the following properties:*

1. *$\cup_{X \in \mathcal{X}} X = V(G)$;*
2. *For every $(u, v) \in E(G)$, there exists $X \in \mathcal{X}$ such that $u, v \in X$;*
3. *For all $t_1, t_2, t_3 \in V(\mathcal{T})$ if t_2 lies on the unique path between t_1 and t_3 in $\mathcal{T}$ then $X_{t_1} \cap X_{t_3} \subseteq X_{t_2}$.*

Let $(\mathcal{T}, \mathcal{X})$ be a tree decomposition of a graph G. We refer to the vertices of the tree $\mathcal{T}$ as *nodes.* Note that since $\mathcal{T}$ is a rooted tree, we have a natural parent-child and ancestor-descendant relationship among nodes in $\mathcal{T}$. A *leaf* node or a *leaf* of $\mathcal{T}$ is a node with degree exactly one in $\mathcal{T}$ which is different from the root node. All the nodes of $\mathcal{T}$ which are neither the root node or a leaf will be called *non-leaf* nodes. The *width* of the tree decomposition $(\mathcal{T}, \mathcal{X})$ is defined to be $\max_{X \in \mathcal{X}}(|X| - 1)$. The *treewidth* of a graph G, denoted by $\mathsf{tw}(G)$, is the minimum of the widths of all its tree decompositions. We use the following structured tree decomposition in our algorithm.

Definition 2. *A tree decomposition $(\mathcal{T}, \mathcal{X} = \{X_t \mid t \in V(T)\})$ with root node as r of G is called a nice tree decomposition if the following conditions are satisfied.*

1. *$X_r = \emptyset$ and $X_\ell = \emptyset$ for every leaf node ℓ in $\mathcal{T}$;*
2. *Every non-leaf node t of $\mathcal{T}$ is of one of the following type:*
    - **Introduce node:** *The node t has exactly one child t' in $\mathcal{T}$ and $X_t = X_{t'} \cup \{v\}$, where $v \notin X_{t'}$.*
    - **Forget node:** *The node t has exactly one child t' in $\mathcal{T}$ and $X_t = X_{t'} \setminus \{v\}$, where $v \in X_{t'}$.*
    - **Join node:** *The node t has exactly two children t_1, t_2 in $\mathcal{T}$ and $X_t = X_{t_1} = X_{t_2}$.*

Lemma 3 ([12, 23]). *If a G has a tree decomposition $(\mathcal{T}, \mathcal{X})$ of width at most w then there is a nice tree decomposition of G of width at most w. Moreover, given a tree decomposition $(\mathcal{T}, \mathcal{X})$ of G of width at most w, in time $O(w^2 \cdot \max(|V(\mathcal{T})|, |V(G)|))$ we can compute a nice tree decomposition of G of width at most w with at most $O(w|V(G)|)$ nodes.*

3 Exact Exponential-Time Algorithms for Happy Coloring

In this section, we consider the happy coloring problems from the viewpoint of exact exponential-time algorithms. Every problem in NP can be solved in time exponential in the input size by a brute-force algorithm. For Weighted MHE (Weighted MHV), such an algorithm goes through each of the at most k^n colorings, and outputs the one maximizing the total weight of the happy edges (vertices). It is natural to ask whether there is an algorithm that is considerably faster than the $k^n n^{O(1)}$-time brute force approach. In what follows, we show that brute-force can be beaten. Let us introduce the following more general problem.

[1]This is abuse of notation: as is commonly done, we call the output of the kernel also a kernel.

Max Weighted Partition
Input: An n-element set N, integer d, and functions $f_1, f_2, \dots, f_d : 2^N \to [-M, M]$ for some integer M.
Question: A d-partition $(S_1, S_2, \dots, S_d)$ of N that maximizes $f_1(S_1) + f_2(S_2) + \cdots + f_d(S_d)$.

Using an algebraic approach, the following has been shown regarding the complexity of the problem.

Theorem 4 (Björklund, Husfeldt, Koivisto [4]). *The* Max Weighted Partition *problem can be solved in $3^n d^2 M \cdot n^{O(1)}$ time and polynomial space. In exponential space, the time can be improved to $2^n d^2 M \cdot n^{O(1)}$.*

In the following, we observe that the weighted variants of both problems can be reduced to Max Weighted Partition. This results in an algorithm that is considerably faster than one running in time $k^n n^{O(1)}$.

Lemma 5. Weighted MHE *and* Weighted MHV *reduce in polynomial time to* Max Weighted Partition.

Proof. Consider the claim for an instance $I = (G, w, k, S, c)$ of Weighted MHE. To construct an instance of Max Weighted Partition, let $N = V(G) \setminus S$, where S is the set of precolored vertices, let $d = k$, and let $M = \sum_{uv \in E(G)} w(uv)$. Define $f_i = \sum_{uv \in E(G[S_i \cup c^{-1}(i)])} w(uv)$, i.e., f_i sums the weights of the edges uv that range over the edge set of the subgraph induced by the union of S_i and $c^{-1}(i)$, the vertices precolored with color i. Thus, a partition $(S_1, \dots, S_k)$ maximizing $f_1(S_1) + \cdots + f_k(S_k)$ maximizes the weight of happy edges.

Finally, consider the claim for an instance $I = (G, w, k, S, c)$ of Weighted MHV. Now, we define $f_i = \sum_{v \in S_i : \forall y \in N(v) : y \in (S_i \cup c^{-1}(i))} w(v)$, i.e., f_i sums the weights of the vertices v for which it holds that v and each neighbor y of v are all colored with color i. Also, we let $M = \sum_{v \in V(G)} w(v)$, but otherwise the argument is the same as above. □

For some NP-complete problems, the fastest known algorithms run in $O^*(2^n)$ time, but we do not necessarily know whether (under reasonable complexity-theoretic assumptions) they are optimal. Indeed, could one have an algorithm that runs in $O^*((2-\varepsilon)^n)$ time, for any $\varepsilon > 0$, for either Weighted MHE or Weighted MHV? We prove that at least for some values of k this bound can be achieved. For this, we recall the following result.

Theorem 6 (Zhang and Li [30]). *For $k = 2$, k-*MHE *and k-*MHV *are solvable in $O(\min\{n^{2/3}m, m^{3/2}\})$ and $O(mn^7 \log n)$ time, respectively.*

We are ready to proceed with the following.

Lemma 7. *For $k = 3$, k-*MHE *and k-*MHV *can be solved in time $O^*(1.89^{n'})$, where n' is the number of uncolored vertices in the input graph.*

Proof. First, consider the claim for an instance $I = (G, S, c)$ of k-MHE. Consider a partition $\mathcal{S} = (S_1, S_2, S_3)$ of the uncolored vertices into $k = 3$ color classes that maximizes the number of happy edges. Also, denote by C_i for $i \in [k]$ the set of vertices precolored with color i. In $V(G) \setminus (S_3 \cup C_3)$, by the optimality of $\mathcal{S}$, it must be the case that $S_1 \cup C_1$ and $S_2 \cup C_2$ have a minimum number of crossing edges. Thus, we can proceed as follows. Observe that in any optimal solution $\mathcal{S}$, there exists $S_i \in \mathcal{S}$ such that $|S_i| \le n'/3$. The number of subsets of size at most $n'/3$ is $2^{H(1/3)n'} < 1.89^{n'}$, using the well-known bound $2^{H(1/3)} < 1.89$, where $H(\cdot)$ is the binary entropy function (for a proof, see e.g., [18, Lemma 3.13]). Thus, we guess S_i by extending it in all possible at most $1.89^{n'}$ ways. Then, for every such partial coloring, we solve an instance of 2-MHE on the remaining graph $G[V(G) \setminus (S_1 \cup C_1)]$ in polynomial time by Theorem 6. Combining the bounds, we obtain an algorithm running in time $O^*(1.89^{n'})$ for 3-MHE.

For 3-MHV, as above we guess the vertices colored i in all possible $1.89^{n'}$ ways. Let us first consider the case $i = 1$. Notice that the vertices in the neighborhood of the vertices colored 1, i.e., $N(S_1 \cup C_1)$ cannot be made happy. We add two new vertices v_2 and v_3 which are precolored 2 and 3 respectively. We add edges from vertices v_2 and v_3 to all the vertices in $N(S_1 \cup C_1)$. Now we solve an instance of 2-MHV on the graph $G[V(G) \cup \{v_2, v_3\} \setminus (S_1 \cup C_1)]$. Repeating this for $i = 2$ and $i = 3$, we get the desired algorithm for 3-MHV. □

By Lemma 7 and by combining Theorem 4 with Lemma 5, we arrive at the following.

Theorem 8. *For every $k \geq 3$, Weighted k-MHE and Weighted k-MHV can be solved in time $O^*(2^{n'})$. When $k = 3$, the problems k-MHE and k-MHV are solvable in time $O^*(1.89^{n'})$, where n' is the number of uncolored vertices in the input graph.*

4 A Linear Kernel for Weighted MHE

In this section, we consider the following decision version of the Weighted MHE.

Weighted DMHE **Parameter:** $k + \ell$
Input: A graph G, integers k and ℓ, a vertex subset $S \subseteq V(G)$, (partial) coloring $c : S \to [k]$, and a weight function $w : E(G) \to \mathbb{N}$.
Question: Does there exist a coloring $\tilde{c} : V(G) \to [k]$ such that $\tilde{c}|_S = c$ and the sum of the weights of the happy edges is at least ℓ?

In particular, we prove that Weighted DMHE has a kernel of size $k + \ell$. Our strategy to obtain the kernel consists of two parts: first, we will show that there is a polynomial-time algorithm for Weighted MHE when the uncolored vertices induce a forest. Then, to leverage this algorithm, we apply a set of reduction rules that shrink the instance considerably, or solve it directly along the way.

4.1 Polynomial Time Algorithm for Subproblems of Weighted MHE

We show that the Weighted MHE problem is polynomial-time solvable when the uncolored vertices $V(G) \setminus S$ induce a tree, where S is the set of precolored vertices. When $V(G) \setminus S$ induces a forest, we run the algorithm for each component in $V(G) \setminus S$ independently. The approach we present is based on dynamic programming, and inspired by the algorithm given in [2].

We define edges *touching* a subtree to be those edges that have at least one endpoint in the subtree. We choose any vertex $r \in V(G) \setminus S$ as the root of the tree induced by $V(G) \setminus S$. The vertices of this rooted tree are processed according to its post-order traversal. At each node, we keep k values. The k values are defined as follows, for $1 \leq i \leq k$:

- $T_v[i]$: The maximum total weight of the happy edges touching the subtree T_v, when the vertex v is colored with color i.

We also define the following expressions:

- $T_v[*]$: The maximum total weight of the happy edges touching the subtree T_v, i.e.,

$$T_v[*] = \max_{i=1}^{k}\{T_v[i]\}. \tag{1}$$

- $T_v[\bar{\imath}]$: The maximum total weight of the happy edges touching the subtree T_v, when the vertex v is colored with a color other than i, i.e.,

$$T_v[\bar{\imath}] = \max_{j=1, j\neq i}^{k}\{T_v[j]\}. \tag{2}$$

If W_p is the total weight of the happy edges in the initial partial coloring, $W_p + T_r[*]$ gives us the maximum total weight of the happy edges in G. Now, we explain how to compute the values $T_v[i]$ for $1 \leq i \leq k$ and for each $v \in V(G) \setminus S$. When we say *color-i* vertices, we mean the vertices precolored with color i.

For a leaf vertex $v \in V(G) \setminus S$, let $v_1, v_2, \ldots, v_x$ be the color-i neighbors of v in G. Then,

$$T_v[i] = \sum_{j=1}^{x} w(vv_j). \tag{3}$$

If there are no color-i neighbors for v, then $T_v[i]$ is set to 0.

For a non-leaf vertex $v \in V(G) \setminus S$, let $v_1, v_2, \ldots, v_x$ be the color-i neighbors of v in G and let $u_1, u_2, \ldots, u_d$ be the children of v in $V(G) \setminus S$. Then,

$$T_v[i] = \sum_{j=1}^{x} w(vv_j) + \sum_{j=1}^{d} \max\{(w(vu_j) + T_{u_j}[i]), T_{u_j}[\bar{i}]\}. \tag{4}$$

This naturally leads to an algorithm listed as Algorithm 1.

Algorithm 1 Algorithm for a special case of Weighted MHE

Input: A weighted undirected graph G with $S \subseteq V(G)$ precolored vertices under a partial vertex-coloring $c : S \to [k]$, $V(G) \setminus S$ induces a tree, and a vertex $r \in V(G) \setminus S$ as the root of the tree.

Output: Maximum total weight of the happy edges in G.

```
1: $M_p \leftarrow 0$
2: for all happy edge $uv$ in the precoloring do
3:     $M_p \leftarrow M_p + w(uv)$
4: end for
5: for all $v \in V(G) \setminus S$ in post-order do
6:     if $v$ is a leaf vertex in $V(G) \setminus S$ then
7:         for $i = 1$ to $k$ do
8:             $T_v[i] \leftarrow 0$
9:             for all $vu \in E(G)$ such that $u \in S$ and $c(u) = i$ do
10:                $T_v[i] \leftarrow T_v[i] + w(vu)$
11:            end for
12:        end for
13:    else
14:        for $i = 1$ to $k$ do
15:            $T_v[i] \leftarrow 0$
16:            for all $vu \in E(G)$ such that $u \in S$ and $c(u) = i$ do
17:                $T_v[i] \leftarrow T_v[i] + w(vu)$
18:            end for
19:            for all child $u$ of $v$ in $V(G) \setminus S$ do
20:                $T_v[i] \leftarrow T_v[i] + \max\{(w(vu) + T_u[i]), T_u[\bar{i}]\}$
21:            end for
22:        end for
23:    end if
24: end for
25: return $(M_p + T_r[*])$
```

The running time of the algorithm is $O(k(m+n))$. The correctness of the values $T_v[i]$, for $1 \le i \le k$ and for each $v \in V(G) \setminus S$, implies the correctness of the algorithm. The following theorem is proved by induction on the size of the subtrees.

Theorem 9. *Algorithm 1 correctly computes the values $T_v[i]$ for every $v \in V(G) \setminus S$ and $1 \le i \le k$.*

Proof. We prove the theorem by using induction on the size of the subtrees. For a leaf vertex v, the algorithm correctly computes the values $T_v[i]$ for $1 \le i \le k$. For a non-leaf vertex v, let $u_1, u_2, \ldots, u_d$ be the children of v in $V(G) \setminus S$. By induction, all the k values associated with each child u_j of v are correctly computed. Moreover, $T_v[i]$ is the sum of two quantities (see Equation 4), the first quantity is correct because it is the sum of the weights of the happy edges from v to S. If $T_v[i]$ is not correct, it will contradict the correctness of $T_{u_j}[*]$ for some child u_j of v. So, the second term in the $T_v[i]$ is correct. Hence, the algorithm correctly computes the values $T_v[i]$ for every v in $V(G) \setminus S$ and $1 \le i \le k$. $\square$

4.2 Reduction Rules Combined with the Algorithm: a Kernel

In this subsection, we assume the edge weights of the Weighted DMHE instance are positive integers. The kernel will also work for real weights that are at least 1. We present the following simple reduction rules.

Rule 1. *If G contains an isolated vertex, delete it.*

Rule 2. *If both endpoints of an edge $uv \in E(G)$ are colored, remove uv. Furthermore, if $c(u) = c(v)$, decrement ℓ by the weight on uv.*

Lemma 10. *Rule 2 is correct.*

Proof. As both endpoints of uv are colored, the existence of the edge uv does not further contribute to the value of the optimal solution. Moreover, if the edge is already happy under c, we can safely decrement ℓ. $\square$

Rule 3. *Identify every color class C_i induced by the partial coloring c into a single vertex. Let $e_1, \dots, e_r$ be the (parallel) edges between two vertices u and v. Delete each edge in $e_1, \dots, e_r$ except for e_1, and update $w(e_1) = w(e_1) + w(e_2) + \cdots + w(e_r)$.*

Lemma 11. *Rule 3 is correct.*

Proof. Let G' be the resulting graph after the application of Rule 3. Because Rule 2 does not apply, each color class C_i forms an independent set. Thus, G' contains no self-loops.

Fix a color i, and consider an uncolored vertex $v \in V(G) \setminus C_i$. Denote by $N_i(v)$ the neighbors of v with color i, and denote by $E[X, Y]$ the set of edges whose one endpoint is in X and the other in Y. Depending on the color v gets in an extended full coloring of c, either all edges in $E[\{v\}, N_i(v)]$ are happy or all are unhappy. Hence, we can safely replace these edges with a single weighted edge. $\square$

Theorem 12. *The problem* Weighted DMHE *admits a kernel on $k + \ell$ vertices.*

Proof. Let (G, w, k, S, c) be a reduced instance of Weighted DMHE. We claim that if G has more than $k + \ell$ vertices, then we have YES-instance. The proof follows by the claims below.

Claim 1. *The weight of each edge is at most ℓ.*

Proof. If an edge uv has $w(uv) \geq \ell$ and at least one of u and v is uncolored, we make uv happy and output YES. On the other hand, any unhappy edge (with any weight) has been removed by Rule 2. $\blacksquare$

Claim 2. *The number of precolored vertices in G is at most k.*

Proof. Follows directly from Rule 3. $\blacksquare$

Claim 3. *The number of uncolored vertices in G is at most $\ell - 1$.*

Proof. Let H be the graph induced by the uncolored vertices, i.e., $H = G[V(G) \setminus \cup_{i \in [k]} C_i]$. We note the following two cases:

- If any of the connected components of H is a tree, then we apply the procedure described in Section 4.1 for that component, and decrement the parameter ℓ accordingly.
- If $w(E(H)) \geq \ell$, then we color all the vertices in H by the same color making all the edges in H happy. So the case where $w(E(H)) \geq \ell$ is a YES-instance.

After the application of the above, every component of H contains a cycle, and $|E(H)| < \ell$. So in each component of H, the number of vertices is at most the number of edges. Consequently, we have $|V(H)| \leq |E(H)| < \ell$. Hence the number of uncolored vertices is at most $\ell - 1$. $\blacksquare$

Clearly, all of the mentioned rules can be implemented to run in polynomial time. Moreover, as we have bounded the number of precolored and uncolored vertices, the claimed kernel follows. $\square$

By combining Theorem 12 with Theorem 8, we have the following corollary.

Corollary 13. *The* Weighted DMHE *problem can be solved in time $O^*(2^\ell)$. For the special case of $k = 3$, the problem* Weighted DMHE *admits an algorithm running in time $O^*(1.89^\ell)$.*

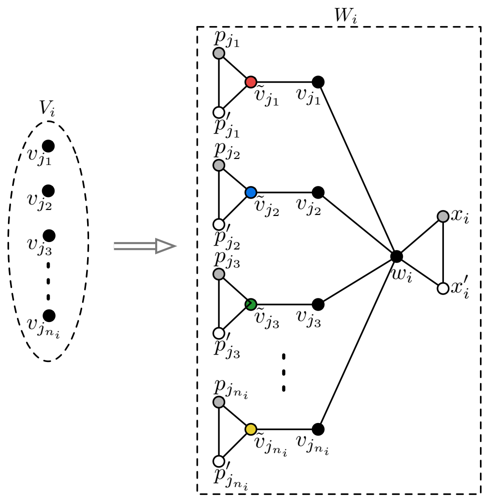

Figure 1: Construction of vertex selection gadget.

5 W[1]-hardness of Maximum Happy Vertices for the Natural Parameter

In this section, we consider the following decision version of the MHV.

DMHV	**Parameter:** ℓ
Input: A graph G, integers k and ℓ, a vertex subset $S \subseteq V(G)$, (partial) coloring $c : S \to [k]$.	
Question: Does there exist a coloring $\tilde{c} : V(G) \to [k]$ such that $\tilde{c}|_S = c$ and the number of happy vertices is at least ℓ?	

We show that DMHV parameterized by the number of happy vertices is W[1]-hard. We give a parameterized reduction from Multicolored Independent Set (MIS) which is known to be W[1]-hard [17]. The problem MIS is formally defined below.

Multicolored Independent Set (MIS)	**Parameter:** t
Input: A t-partite graph G with a partition $V_1, V_2, \dots, V_t$ of $V(G)$.	
Question: Does there exist $X \subseteq V(G)$, such that for each $i \in [t]$, $|X \cap V_i| = 1$ and $G[X]$ is an independent set?	

Intuitively, given an instance $(G, V_1, V_2, \dots, V_t)$ of MIS, for each V_i we create a vertex selection gadget, W_i which ensures that exactly one vertex from V_i can be happy in any valid coloring. Furthermore, the selected set of vertices from V_is form a set of happy vertices in the instance of DMHV created. We now move to the formal description of the reduction.

Let $(G, V_1, V_2, \dots, V_t)$ be an instance of MIS. We create an instance (G', k, ℓ, S, c) of DMHV as follows. Let $n = |V(G)|$ and $V(G) = \{v_i \mid i \in [n]\}$. Initially, we have $V(G') = V(G)$ and $E(G') = \{(u, v) \in E(G) \mid u \in V_i, v \in V_j, i, j \in [t] \text{ and } i \neq j\}$. We now describe the vertex selection gadget W_i, for $i \in [t]$. For each $v_i \in V(G)$, we add three vertices $\tilde{v}_i, p_i, p'_i$ to $V(G')$, and add the edges $(v_i, \tilde{v}_i)$, $(\tilde{v}_i, p_i)$, $(\tilde{v}_i, p'_i)$, and (p_i, p'_i) to $E(G')$. Furthermore, we add $\tilde{v}_i, p_i$, and p'_i to S, and set $c(\tilde{v}_i) = i$, $c(p_i) = n + 1$, and $c(p'_i) = n + 2$. For $i \in [t]$, we add three vertices w_i, x_i, x'_i to $V(G')$ and add all the edges in $\{(u, w_i) \mid u \in V_i\} \cup \{(w_i, x_i), (w_i, x'_i), (x_i, x'_i)\}$ to $E(G')$. Furthermore, we add x_i and x'_i to S, and set $c(x_i) = n + 1$ and $c(x'_i) = n + 2$. Here, the vertices x_i and x'_i are added to ensure that w_i can never be a happy vertex in any coloring of G', and w_i is added to ensure that at most one vertex from V_i can be happy in any coloring of $V(G')$. We have $W_i = G'[V_i \cup \{\tilde{v}_j, p_j, p'_j \mid v_j \in V_i\} \cup \{w_i, x_i, x'_i\}]$. Notice that we have $S = \{\tilde{v}_j, p_j, p'_j \mid v_j \in V(G)\} \cup \{x_i, x'_i \mid i \in [t]\}$. Note that for each $u \in S$, we have described the value of $c(u)$, and we have $k = n + 2$. Finally, we set $\ell = t$, and the resulting instance of DMHV is (G', k, ℓ, S, c).

We state some lemmata which establish certain properties of the instance (G', k, ℓ, S, c) of DMHV that we created.

Lemma 14. *Let $\tilde{c}$ be a coloring that extends c to a coloring of G', and H be the set of happy vertices in G' with respect $\tilde{c}$. Then for all $u \in \{w_i \mid i \in [t]\} \cup S$ we have $u \notin H$.*

Proof. Consider $w_{i^*} \in \{w_i \mid i \in [t]\}$. Recall that by construction, w_{i^*} has two neighbors x_{i^*} and x'_{i^*} with $\tilde{c}(x_{i^*}) = c(x'_{i^*}) = n+1$ and $\tilde{c}(x'_{i^*}) = c(x'_{i^*}) = n+2$. Therefore, $w_i \notin H$. For each $u \in S$, by construction we have a vertex $v \in N_{G'}(u) \cap S$ such that $\tilde{c}(u) = c(u) \neq c(v) = \tilde{c}(v)$, therefore $u \notin H$. □

Lemma 15. *Let $\tilde{c}$ be a coloring that extends c to a coloring of G', and H be the set of happy vertices in G' with respect $\tilde{c}$. Then for all $i \in [t]$, we have $|V_i \cap H| \leq 1$.*

Proof. Consider $i \in [t]$, such that $|V_i \cap H| > 1$. Let $v_j, v_{j'}$ be two distinct $(j \neq j')$ vertices in $V_i \cap H$. Since $v_j \in H$, and $\tilde{c}(\tilde{v}_j) = c(\tilde{v}_j) = j$, we have $\tilde{c}(v_j) = j$. Since $w_i \in N_{G'}(v_j)$ therefore, we have $\tilde{c}(w_i) = j$. By similar arguments we have $\tilde{c}(v_{j'}) = j'$ and $\tilde{c}(w_i) = j'$. This implies that $j = j'$, a contradicting. □

We now state the main lemma of this section.

Lemma 16. *$(G, V_1, V_2, \ldots, V_t)$ is a YES-instance of* MIS *if and only if (G', k, ℓ, S, c) is a YES-instance of* DMHV*.*

Proof. In the forward direction, let $(G, V_1, V_2, \ldots, V_t)$ be a YES-instance of MIS and $X = \{v_{i^\star} \mid i \in [t]\}$ be one of its solutions. We construct a solution $\tilde{c}$ to DMHV in (G', k, ℓ, S, c) as follows. For each $u \in S$, we let $\tilde{c}(u) = c(u)$. For $i \in [t]$, for each $v \in V_i$, we let $\tilde{c}(v) = i^\star$ and $\tilde{c}(w_i) = i^\star$. This completes the description of $\tilde{c}$. Notice that for each $v_{i^\star} \in X$, we have that for all $v \in N_{G'}(v_{i^\star})$, $\tilde{c}(v) = i^\star$. This implies that $X \subseteq H$, where H is the set of happy vertices in G' with respect to $\tilde{c}$. Moreover, we have $|X| = t = \ell$. This concludes the proof in the forward direction.

In the reverse direction, let (G', k, ℓ, S, c) be a YES-instance of DMHV, $\tilde{c}$ be one of its solutions, and H be the set of happy vertices in G' with respect to $\tilde{c}$. We show that H is a solution to MIS in $(G, V_1, V_2, \ldots, V_t)$. By construction we have $V(G') = \cup_{i \in [t]} V(W_i)$. Lemma 14 implies that $H \cap (\{w_i \mid i \in [t]\} \cup S) = \emptyset$, and Lemma 15 implies that for each $i \in [t]$, we have $|H \cap V_i| \leq 1$. This together with the fact that $|H| \geq \ell$ implies that for each $i \in [t]$, we have $|H \cap V_i| = 1$. Therefore, we only need to show that H is an independent set in G. For $i \in [t]$, let $h_{i^\star}$ be the unique vertex in $H \cap V_i$. Consider $h_{i^\star}, h_{j^\star} \in H$, where $i, j \in [t]$ and $i \neq j$. Recall that $h_{i^\star}$ has a neighbor $\tilde{h}_{i^\star}$ in G' such that $\tilde{c}(\tilde{h}_{i^\star}) = i^\star$. Similarly, $h_{j^\star}$ has a neighbor $\tilde{h}_{j^\star}$ in G' such that $\tilde{c}(\tilde{h}_{i^\star}) = j^\star$. Therefore, we have $\tilde{c}(h_{i^\star}) = i^\star$ and $\tilde{c}(h_{j^\star}) = j^\star$, where $i^\star \neq j^\star$. This implies that $(h_{i^\star}, h_{j^\star}) \notin E(G')$, and hence by construction we have $(h_{i^\star}, h_{j^\star}) \notin E(G)$. Therefore, H is an independent set in G. □

Theorem 17. DMHV *when parameterized by the number of happy vertices is* W[1]*-hard.*

Proof. Follows from the construction of the instance (G', k, ℓ, S, c) of DMHV for the given instance $(G, V_1, V_2, \ldots, V_t)$ of MIS, Lemma 16, and W[1]-hardness of MIS. □

6 Kernelization Algorithm for Maximum Happy Vertices

In this section, we give a polynomial kernel for the problem DMHV. In fact, we give a kernel for an annotated version of DMHV, which we call Annotated DMHV (ADMHV). The problem is formally defined below.

Annotated DMHV (ADMHV) **Parameter:** $k + \ell$
Input: A graph G, integers k and ℓ, a vertex subsets $S, U \subseteq V(G)$, a (partial) coloring $c : S \to [k]$.
Question: Does there exist a coloring $\tilde{c} : V(G) \to [k]$ such that $\tilde{c}|_S = c$ and $|H \setminus U| \geq \ell$, where H is the set of happy vertices in G with respect to $\tilde{c}$?

Observe that DMHV is a special case of ADMHV, where $U = \emptyset$. Moreover, given an instance (G, k, ℓ, S, U, c) of ADMHV, in polynomial time we can construct an instance (G', k', ℓ, S', c') of DMHV such that $|V(G')| \in O(|V(G)|)$, $k' \in O(k)$, and $|S'| \in O(|S|)$ as follows. Initially, we have $G' = G$ and $c' = c$. We add two (new) vertices $u^\star, v^\star$ to $V(G')$, add the edge $(u^\star, v^\star)$ to $E(G')$, add $u^\star, v^\star$ to S, and set $c'(u^\star) = k+1$ and $c'(v^\star) = k+2$. Furthermore, we add the edges $\{(u, u^\star), (u, v^\star) \mid u \in U\}$ to $E(G')$ and set $k' = k+2$. It is easy to see that (G, k, ℓ, S, U, c) is a YES-instance of ADMHV if and only if (G', k', ℓ, S', c') is a YES-instance of DMHV. Therefore, to design a kernel for DMHV with $O(k^2\ell^2)$ vertices it is enough to design a kernel for ADMHV with $O(k^2\ell^2)$ vertices. Hereafter, the focus of this section will be to design a kernel with $O(k^2\ell^2)$ vertices for ADMHV.

Let (G, k, ℓ, S, U, c) be an instance of ADMHV. The kernelization algorithm applies the following reduction rules in the order in which it is stated. Furthermore, at each step we assume that none of the preceding reduction rules are applicable. When none of the reduction rules are applicable we argue that we get a kernel of the desired size.

Rule 4. *If $\ell \leq 0$, then return that (G, k, ℓ, S, U, c) is a YES-instance of* ADMHV.

Observe that if $\ell \leq 0$, then any coloring that extends c to a coloring of $V(G)$ is a valid solution to the instance (G, k, ℓ, S, U, c) of ADMHV, which implies that Rule 4 is safe.

Rule 5. *Let $v \in V(G) \setminus U$ be a vertex such that $N(v) \subseteq S$, for all $u, u' \in N(v)$ we have $c(u) = c(u')$, and one of the following conditions is satisfied. i) $v \notin S$; or ii) $c(v) = c(u)$, where $u \in N(v)$. Then delete v from G and decrease ℓ by one. The resulting instance is $(G - \{v\}, k, \ell - 1, S \setminus \{v\}, U, c|_{S\setminus\{v\}})$.*

Lemma 18. *Rule 5 is safe.*

Proof. Let (G, k, ℓ, S, U, c) be an instance of ADMHV and $v \in V(G) \setminus U$ be a vertex such that $N(v) \subseteq S$, and for all $u, u' \in N(v)$ we have $c(u) = c(u')$. Furthermore, let $(G', k, \ell - 1, S', U, c')$ be the instance resulting after application of the Rule 5, where $G' = G - \{v\}$, $S' = S \setminus \{v\}$, and $c' = c|_{S'}$. The proof of forward direction follows from the fact that $G' = G[V(G) \setminus \{v\}]$. In the reverse direction let $\tilde{c}'$ be a coloring that extends $c|_{S'}$ to a coloring of $V(G')$ such that the number of happy vertices in $V(G') \setminus U$ is at least $\ell - 1$. If $v \in S$, then the coloring $\tilde{c}$ obtained by extending $\tilde{c}'$ to a coloring of $V(G)$ with $\tilde{c}(v) = c(v)$ is a coloring that extends c to a coloring of $V(G)$ with at least ℓ happy vertices in $V(G) \setminus U$ in G. Otherwise, we have $v \notin S$. In this case, let $\tilde{c}$ be the coloring obtained by extending $\tilde{c}'$ to a coloring of $V(G)$ with $\tilde{c}(v) = c(u)$, where $u \in N_G(v)$. Observe that $\tilde{c}$ is a coloring that extends c to a coloring of $V(G)$, and there are at least ℓ happy vertices in $V(G) \setminus U$ in G with respect to $\tilde{c}$. This concludes the proof. $\square$

Rule 6. *Let $v \in S \setminus U$ be a vertex such that there exists $u \in N(v) \cap S$ with $c(v) \neq c(u)$. Then add v to the set U. The resulting instance is $(G, k, \ell, S, U \cup \{v\}, c)$.*

The safeness of Rule 6 follows from the fact that a vertex $v \in S \setminus U$ with $u \in N(v) \cap S$ such that $c(v) \neq c(u)$ can never be a happy vertex in any coloring of G that extends c to a coloring of $V(G)$.

Rule 7. *Let $v \in V(G) \setminus U$ be a vertex such that there exists $u, u' \in N(v) \cap S$ with $c(u) \neq c(u')$. Then add v to the set U. The resulting instance is $(G, k, \ell, S, U \cup \{v\}, c)$.*

The safeness of Rule 7 follows from the fact that a vertex v with $u, u' \in N(v) \cap S$ such that $c(u) \neq c(u')$ can never be a happy vertex in any coloring of G that extends c to a coloring of $V(G)$.

Next we consider the following sets. For $i \in [k]$, let $U_i = \{v \in U \cap S \mid c(v) = i\}$, and $U_R = U \setminus (\cup_{i\in[k]} U_i)$. We proceed with the following reduction rules.

Rule 8. *If there exists $i \in [k]$ such that there are distinct $u, v \in U_i$ then identify u, v in G to obtain the graph G' with $u^\star$ being the vertex resulting after identification. Furthermore, let $c' : (S \setminus \{u, v\}) \cup \{u^\star\} \to [k]$ be the coloring obtained from c with $c'|_{S\setminus\{u,v\}} = c$ and $c'(u^\star) = c(u)$. The resulting instance is $(G', k, \ell, (S \setminus \{u, v\}) \cup \{u^\star\}, (U \setminus \{u, v\}) \cup \{u^\star\}, c')$.*

Lemma 19. *Rule 8 is safe.*

Proof. Let $i \in [k]$ such that there are distinct $u, v \in U_i$, and G' be the graph obtained from G after identifying u and v with $u^\star$ being the resulting vertex after identification. Furthermore, let $U' = (U \setminus \{u, v\}) \cup \{u^\star\}$, $S' = (S \setminus \{u, v\}) \cup \{u^\star\}$, and $c' : S' \to [k]$ be the coloring obtained from c with $c'|_{S'\setminus\{u^\star\}} = c|_{S\setminus\{u,v\}}$ and $c'(u^\star) = c(u)$. We will show that (G, k, ℓ, S, U, c) is a YES-instance of ADMHV if and only if $(G', k, \ell, S', U', c')$ is a YES-instance of ADMHV.

In the forward direction let (G, k, ℓ, S, U, c) be a YES-instance of ADMHV, $\tilde{c}$ be one of its solutions, and $H \subseteq V(G) \setminus U$ be the set of happy vertices in G with respect to $\tilde{c}$. Notice that we have $|H| \geq \ell$.

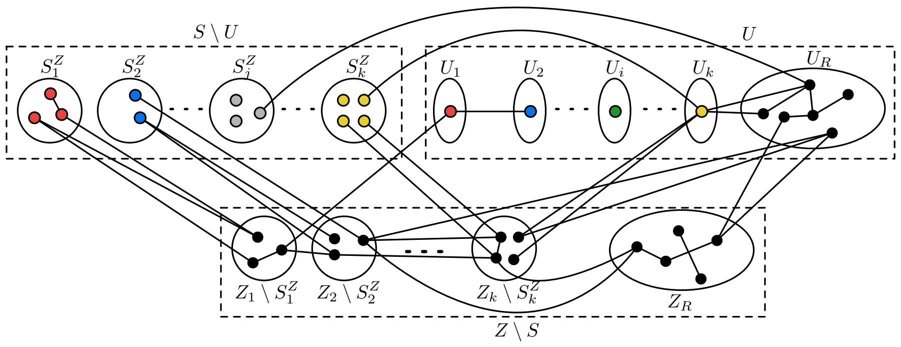

Figure 2: An illustration of partition of $V(G)$ into various sets.

Let $\tilde{c}' : V(G') \to [k]$ be the coloring obtained from $\tilde{c}$ with $\tilde{c}'(u^\star) = \tilde{c}(u)$ and $\tilde{c}'|_{V(G')\setminus\{u^\star\}} = \tilde{c}|_{V(G)\setminus\{u,v\}}$. Recall that $V(G') \setminus \{u^\star\} = V(G) \setminus \{u, v\}$, hence $\tilde{c}'$ is a coloring of G'. Furthermore, we have $\tilde{c}'|_{S'} = c'$. Hence, we only need to show that with respect to $\tilde{c}'$ in G' we have at least ℓ vertices in $V(G') \setminus U'$ that are happy. Observe that $H \subseteq V(G') \setminus U'$. We claim that all the vertices in H are happy in G' with respect to $\tilde{c}'$, which is enough to show that $(G', k, \ell, S', U', c')$ is a YES-instance of ADMHV. Consider a vertex $h \in H$. Recall that $N_{G'}(h) \setminus U' = N_G(h) \setminus U$. This together with the construction of $\tilde{c}'$ implies that for all $w, w' \in N_{G'}(h) \setminus U'$ we have $\tilde{c}'(w) = \tilde{c}'(w') = \tilde{c}'(h)$. For $w \in N_{G'}(h) \cap U'$ if $\tilde{c}'(w) \neq \tilde{c}'(h)$ then consider the following cases. If $w = u^\star$ then replacing w by u (or v) in G we have that $\tilde{c}(h) \neq \tilde{c}(w)$, contradicting that h is a happy vertex in G with respect to $\tilde{c}$. On the other hand if $w \neq u^\star$ then $w \in U$ and in G we have that $\tilde{c}(h) \neq \tilde{c}(w)$, a contradiction. This concludes the proof in the forward direction.

In the reverse direction let $(G', k, \ell, S', U', c')$ be a YES-instance of ADMHV, $\tilde{c}'$ be one of its solutions, and $H \subseteq V(G') \setminus U'$ be the set of happy vertices in G' with respect to $\tilde{c}'$. Notice that $|H| \geq \ell$. Let $\tilde{c} : V(G) \to [k]$ be the coloring obtained from $\tilde{c}'$ with $\tilde{c}(u) = \tilde{c}(v) = \tilde{c}'(u^\star)$ and $\tilde{c}|_{V(G)\setminus\{u,v\}} = \tilde{c}'|_{V(G')\setminus\{u^\star\}}$. Observe that we have $\tilde{c}|_S = c$. Hence, we only need to show that with respect to $\tilde{c}$ in G we have at least ℓ vertices in $V(G) \setminus U$ that are happy. Observe that $H \subseteq V(G) \setminus U$. We claim that all vertices in H are happy in G with respect to $\tilde{c}$, which is enough to show that (G, k, ℓ, S, U, c) is a YES-instance of ADMHV. Consider a vertex $h \in H$. Recall that $N_G(h) \setminus U = N_{G'}(h) \setminus U'$. This together with the construction of $\tilde{c}$ implies that for all $w \in N_G(h) \setminus U$ we have $\tilde{c}(w) = \tilde{c}(h)$. For $w \in N_G(h) \cap U$ if $\tilde{c}(w) \neq \tilde{c}(h)$ then consider the following cases. If $w \in \{u, v\}$ then replacing w by $u^\star$ in G' we have that $\tilde{c}'(h) \neq \tilde{c}'(w)$, contradicting that h is a happy vertex in G' with respect to $\tilde{c}'$. On the other hand if $w \notin \{u, v\}$ then $w \in U'$ and in G' we have that $\tilde{c}'(h) \neq \tilde{c}'(w)$, a contradiction. ☐

Hereafter, we assume that Rule 8 is not applicable and hence for each $i \in [k]$, we have $|U_i| \leq 1$. Let $Z = V(G) \setminus U$. For $i \in [k]$, let $S_i^Z = \{v \in S \cap Z \mid c(v) = i\}$ and $Z_i = (Z \cap N(U_i \cup S_i^Z)) \cup S_i^Z$. Furthermore, we let $Z_R = Z \setminus (\cup_{i\in[k]} Z_i)$. Observe that for $i, j \in [k]$, $i \neq j$ we have $Z_i \cap Z_j = \emptyset$ since Rule 6 and 7 are not applicable. Also, for each $v \in Z_R$, we have $N(v) \subseteq V(G) \setminus S$. We proceed with the following reduction rules.

Rule 9. *If there exists $i \in [k]$ such that $|Z_i| \geq \ell$ then return that (G, k, ℓ, S, U, c) is a YES-instance of* ADMHV.

Lemma 20. *Rule 9 is safe.*

Proof. Let (G, k, ℓ, S, U, c) be an instance of ADMHV such that $|Z_R| \geq \ell$. Consider the coloring $\tilde{c}$: $V(G) \to [k]$ with $\tilde{c}|_S = c$, and for all $v \in V(G) \setminus S$, $\tilde{c}(u) = i$. We claim that $\tilde{c}$ is a solution to ADMHV in (G, k, ℓ, S, U, c), where all the vertices in Z_i are happy with respect to $\tilde{c}$. Note that by definition $\tilde{c}$ extends c to a coloring of G therefore, we only need to show that vertices in Z_i are happy in G with respect to $\tilde{c}$. Consider a vertex $z \in Z_i$ and a vertex $v \in N(z)$. By the definition of $\tilde{c}$ and Z_i, we have $\tilde{c}(z) = i$. If $v \in S$ then we have $\tilde{c}(v) = c(v) = i$, since Rule 6 and 7 are not applicable. If $v \in V(G) \setminus S$ then by definition of $\tilde{c}$ we have $\tilde{c}(v) = i$. Therefore, z is a happy vertex in G with respect to $\tilde{c}$. This concludes the proof. ☐

Rule 10. *If $|Z_R| \geq \ell$ then return that (G, k, ℓ, S, U, c) is a YES-instance of* ADMHV.

Lemma 21. *Rule 10 is safe.*

Proof. Let (G, k, ℓ, S, U, c) be an instance of ADMHV such that $|Z_R| \geq \ell$. Consider the coloring $\tilde{c} : V(G) \to [k]$ with $\tilde{c}|_S = c$, and for all $v \in V(G) \setminus S$, $\tilde{c}(u) = 1$. We claim that $\tilde{c}$ is solution to ADMHV in (G, k, ℓ, S, U, c), where all the vertices in Z_R are happy with respect to $\tilde{c}$. Note that by definition $\tilde{c}$ extends c to a coloring of G therefore, we only need to show that vertices in Z_R are happy in G with respect to $\tilde{c}$. Consider a vertex $z \in Z_R$ and a vertex $v \in N(z)$. By the definition of $\tilde{c}$ we have $\tilde{c}(z) = 1$. By the construction of sets Z_R, we have $N(z) \subseteq V(G) \setminus S$ and by definition of $\tilde{c}$ it follows that $\tilde{c}(v) = 1$. Therefore, z is a happy vertex in G with respect to $\tilde{c}$. This concludes the proof. □

Notice that since Rule 7 is not applicable we have for each $i \in [k]$, $|U_i| = 1$. Furthermore, since Rule 9 is not applicable we have for each $i \in [k]$, $|Z_i| < \ell$, and since Rule 10 is not applicable we have $|Z_R| < \ell$. Therefore, we have $|Z \cup (\cup_{i \in [k]} U_i)| \leq k\ell + \ell - 1$. We now move to bounding the size of U_R, which will give us the desired kernel. To bound the size of U_R we employ the following marking scheme and argue that all the unmarked vertices can be deleted.

Marking Scheme for bounding $|U_R|$. We will denote the set of marked vertices by $M^\star \subseteq U_R$. For all $u, v \in V(G) \setminus U_R$ (not necessarily distinct) such that $N(u) \cap N(v) \cap U_R \neq \emptyset$, choose an arbitrary vertex in $w_{uv} \in N(u) \cap N(v) \cap U_R$ and add it to $M^\star$. That is we add a vertex in U_R to the marked set of vertices which is a common neighbor to vertices u and v.

We call a vertex in $U_R \setminus M^\star$ as an unmarked vertex. We now move to the reduction rule which deletes an unmarked vertex.

Rule 11. *If there exists $u \in U_R \setminus M^\star$ then delete u from G. The resulting instance is $(G - \{u\}, k, \ell, S, U \setminus \{u\}, c)$.*

Lemma 22. *Rule 11 is safe.*

Proof. Let (G, k, ℓ, S, U, c) be an instance of ADMHV, $u \in U_R \setminus M^\star$, $G' = G - \{u\}$, and $U' = U \setminus \{u\}$. Recall that $U_R \cap S = \emptyset$ and therefore, $u \notin S$. The resulting instance after deletion of u in G is (G', k, ℓ, S, U', c). We will show that (G, k, ℓ, S, U, c) is a YES-instance of ADMHV if and only if (G', k, ℓ, S, U', c) is a YES-instance of ADMHV.

In the forward direction let (G, k, ℓ, S, U, c) be a YES-instance of ADMHV, $\tilde{c}$ be one of its solutions, and $H \subseteq V(G) \setminus U$ be the set of happy vertices in G with $|H| \geq \ell$. Let $\tilde{c}' : V(G') \to [k]$ be the coloring obtained from $\tilde{c}$ with $\tilde{c}' = \tilde{c}|_{V(G')}$. Notice that since $G' = G[V(G) \setminus \{u\}]$, therefore it follows that all the vertices in H are happy in G' with respect to the coloring $\tilde{c}'$. This concludes the proof in the forward direction.

In the reverse direction let (G', k, ℓ, S, U', c) be a YES-instance of ADMHV, $\tilde{c}'$ be one of its solutions, and $H \subseteq V(G') \setminus U'$ be the set of happy vertices in G' with $|H| \geq \ell$. Let $\tilde{c} : V(G) \to [k]$ be the coloring obtained from $\tilde{c}'$ with $\tilde{c}|_{V(G')} = \tilde{c}'$ and $\tilde{c}(u)$ to be determined shortly. Since $u \notin M^\star$, for all $v, v' \in N(u) \cap Z$, we have a vertex $w_{vv'} \in M^\star \cap N(v) \cap N(v')$. Consider the set $H_u = H \cap N(u)$. If $H_u = \emptyset$, then we set $\tilde{c}(u) = 1$ (or any $i \in [k]$). Otherwise, we have $H_u \neq \emptyset$. Observe that for all $h, h' \in H_u$ we have $\tilde{c}'(h) = \tilde{c}'(h')$. Therefore, we set $\tilde{c}(u) = \tilde{c}'(h)$, where $h \in H_u$. The construction of $\tilde{c}$ implies that all the vertices in H are happy in G with respect to $\tilde{c}$. This concludes the proof. □

Once Rule 11 is not applicable we have $|U_R| \leq \binom{|Z|}{2} + |Z|$. Therefore, when none of the Rules 4 to 11 are applicable, we get the desired kernel. Hence, we obtain the following theorem.

Theorem 23. ADMHV *admits a kernel with $O(k^2\ell^2)$ vertices, where k is the number of colors in the coloring function and ℓ is the desired number of happy vertices.*

7 Structural Parameterization for Happy Coloring

In this section, we consider happy coloring from the standpoint of various structural parameters: treewidth, neighborhood diversity, vertex cover number, and deletion distance to a clique. We begin by considering treewidth. For the unweighted vertex-variant, we give an algorithm whose runtime is optimized to an extent, while a less optimized algorithm is given for the weighted edge-variant. Our main goal is merely to prove fixed-parameter tractability for treewidth. Indeed, we believe that the techniques presented can be used to derive e.g., a faster algorithm for the edge-variant.

7.1 Treewidth

In this subsection, we design a dynamic programming based FPT algorithm for Maximum Happy Vertices when parameterized by the treewidth of input graph and the number of colors in the precoloring of a subset of vertices. The problem is formally defined as follows.

MHV	**Parameter:** $k, \mathsf{tw}(G)$
Input: A graph G, an integer k, a vertex subset $S \subseteq V(G)$, and a (partial) coloring $c : S \subseteq V(G) \to [k]$.	
Output: A coloring $\tilde{c} : V(G) \to [k]$ such that $\tilde{c}|_S = c$ maximizing the number of happy vertices.	

Let (G, k, S, c) be an instance of MHV, $n = |V(G)|$ and $m = |E(G)|$. Without loss of generality we assume that for each $i \in [k]$, we have $c^{-1}(i) \neq \emptyset$, otherwise we can adjust the instance appropriately by adding isolated vertices. For each $i \in [k]$, we arbitrarily choose a vertex from $c^{-1}(i)$, which we denote by $v_i^\star$. We let $S^\star = \{v_i^\star \mid i \in [k]\}$, and $\mathcal{S}^\star = (\{v_1^\star\}, \{v_2^\star\}, \ldots, \{v_k^\star\})$. We start by computing a tree decomposition $(\bar{\mathcal{T}}, \bar{\mathcal{X}})$ of width at most $w \leq 6 \cdot \mathrm{tw}(G)$ in time $O(2^{O(\mathrm{tw}(G))}n)$, using the algorithm by Bodlaender et al. [8]. Using Lemma 3 we compute a nice tree decomposition $(\mathcal{T}', \mathcal{X}' = \{X'_t \mid t \in V(\mathcal{T}')\})$ of G with the root node as r' and width at most w, in time $O(\mathrm{tw}(G)^2 n)$. We modify the nice tree decomposition $(\mathcal{T}', \mathcal{X}')$ to obtain a more structured tree decomposition $(\mathcal{T}, \mathcal{X})$ with root node $r = r'$ as follows. We let $\mathcal{T} = \mathcal{T}'$, and $\mathcal{X} = \{X_t = X'_t \cup S^\star \mid X'_t \in \mathcal{X}'\}$, i.e., $(\mathcal{T}, \mathcal{X})$ is obtained from $(\mathcal{T}', \mathcal{X}')$ by adding all the vertices in $S^\star$ to each bag of $\mathcal{X}'$. Note that width of $(\mathcal{T}, \mathcal{X})$ is bounded by $w + k \leq 6 \cdot \mathrm{tw}(G) + k$. The purpose of adding all vertices in $S^\star$ to each bag is to ensure the subgraph induced by the subtree rooted at a node contains vertices of all k colors, which simplifies the proof. We note that the notion of introduce node, forget node, and join node naturally extends to the tree decomposition $(\mathcal{T}, \mathcal{X})$.

For a node $t \in V(\mathcal{T})$, by $\mathsf{desc}(t)$ we denote the set of nodes which are descendants of t (including t) in $\mathcal{T}$. Furthermore, for $t \in V(\mathcal{T})$, by G_t we denote the graph $G[V_t]$, where $V_t = \cup_{t' \in \mathsf{desc}(t)} X_{t'}$.

We now move to the description of the entries of the dynamic programming table. Consider a node $t \in V(\mathcal{T})$, and an ordered partition $\mathcal{P} = (P_1, P_2, \ldots P_k)$ of X_t into k sets. We call $\mathcal{P}$ a *valid* ordered partition if and only if for all $i \in [k]$, $c^{-1}(i) \cap X_t \subseteq P_i$. Note that for any valid ordered partition $\mathcal{P} = (P_1, P_2, \ldots, P_k)$, for all $i \in [k]$, we have $P_i \neq \emptyset$. This follows from the fact that $S^\star \subseteq X_t$.

For a valid ordered partition $\mathcal{P} = (P_1, P_2, \ldots P_k)$ of X_t, let $\mathcal{H} = \{(H_i, U_i) \mid H_i \uplus U_i = P_i \text{ and } i \in [k]\}$ be a set comprising of ordered pairs, which are partitions of the sets P_is into two sets. A tuple $\tau = (t, \mathcal{P}, \mathcal{H})$ is a *valid tuple* if $\mathcal{P}$ is a valid ordered partition. For a valid tuple $\tau = (t, \mathcal{P}, \mathcal{H})$, a coloring $c_\tau : V(G_t) \to [k]$ is called a *τ-good coloring* if all the following conditions are satisfied.

1. For all $i \in [k]$, we have $P_i \subseteq c_\tau^{-1}(i)$;
2. For all $i \in [k]$, all the vertices in H_i are happy in G_t with respect to c_τ;
3. $c_\tau|_{S \cap V(G_t)} = c|_{S \cap V(G_t)}$.

For every valid tuple $\tau = (t, \mathcal{P}, \mathcal{H})$, we have a table entry denoted by $\Pi(\tau)$ which is set to an element $z \in [|V(G_t)|] \cup \{-\infty\}$. Intuitively, $\Pi(\tau)$ is set to an element $z \in [|V(G_t)|] \cup \{-\infty\}$ which corresponds to the maximum of the number of happy vertices in G_t over all *τ-good colorings* (if it exists). Formally, the value of $\Pi(\tau)$ is determined as follows.

1. If there is no τ-good coloring of G_t then $\Pi(\tau) = -\infty$.
2. Otherwise, over all *τ-good colorings* of G_t, $\Pi(\tau)$ is set to the maximum of the number of happy vertices in $V(G_t) \setminus (\cup_{i \in [k]} U_i)$ of G_t over all such colorings.

Let $\mathbb{H}^\star$ be the set comprising of all $\mathcal{H} = \{(H_i, U_i) \mid H_i \uplus U_i = P_i \text{ and } i \in [k]\}$. Observe that $\max_{\mathcal{H} \in \mathbb{H}^\star} \Pi(r, \mathcal{S}^\star, \mathcal{H})$ is exactly the number of happy vertices in G maximized over all colorings that extends c to a coloring of $V(G)$. We now move to the description of how the values of $\Pi(\cdot)$ are computed. Since we have a structured form of tree decomposition we compute the value of each of the entries at node $t \in V(\mathcal{T})$ based on the entries of its children, which will be given by the recursive formula. For leaf nodes, we compute the values directly, which corresponds to the base case for the recursive formula. Therefore, by computing the formula in a bottom-up fashion we compute the value of $\Pi(r, \mathcal{S}^\star, \mathcal{H})$, for each $\mathcal{H} \in \mathbb{H}^\star$, and hence the value of $\max_{\mathcal{H} \in \mathbb{H}^\star} \Pi(r, \mathcal{S}^\star, \mathcal{H})$. We now move to the description of computing $\Pi(\tau)$, where $\tau = (t, \mathcal{P} = (P_1, \ldots, P_k), \mathcal{H} = \{(H_i, U_i) \mid H_i \uplus U_i = P_i \text{ and } i \in [k]\})$ is a valid tuple.

Leaf node. Suppose t is a leaf node. In this case, we have $X_t = S^\star$, and $\mathcal{P} = \mathcal{S}^\star$. Note that in this case there is exactly one τ-good coloring of G_t namely, $c|_{S^\star}$. Moreover, we can find the set of happy vertices H, in G_t with respect to $c|_{S^\star}$ by looking at the adjacencies between the vertices in $S^\star$. If there exist $i \in [k]$ such that $H_i \setminus H \neq \emptyset$ then we set $\Pi(\tau) = -\infty$. Otherwise, we set $\Pi(\tau) = |H \setminus (\cup_{i\in[k]} U_i)|$. The correctness of setting the values as described is justified by the uniqueness of τ-good coloring in G_t.

Introduce node. Suppose t is an introduce node. Let t' be the unique child of t in $\mathcal{T}$, and $X_t = X_{t'} \cup \{\tilde{v}\}$, where $\tilde{v} \notin X_{t'}$. Furthermore, let P_i be the set containing $\tilde{v}$, where $i \in [k]$. Recall that by the properties of tree decomposition, there is no $u \in N_{G_t}(\tilde{v}) \setminus X_t$, i.e., all the neighbors of $\tilde{v}$ in G_t are in X_t. Let $\mathcal{P}' = (P_1, P_2, \ldots, P_i \setminus \{\tilde{v}\}, \ldots, P_k)$, and $\mathcal{H}' = (\mathcal{H} \setminus \{(H_i, U_i)\}) \cup \{(H_i \setminus \{\tilde{v}\}, U_i \setminus \{\tilde{v}\})\}$. Finally, let $\tau' = (t', \mathcal{P}', \mathcal{H}')$. Note that τ' is a valid tuple. We start by considering the following simple cases where we can immediately set the value of $\Pi(\tau)$.

- Case 1: If $\tilde{v} \in H_i$ and there is $j \in [k] \setminus \{i\}$ such that $P_j \cap N_{G_t}(\tilde{v}) \neq \emptyset$ then we set $\Pi(\tau) = -\infty$ since for any τ-good coloring of G_t, $\tilde{v}$ is not a happy vertex.
- Case 2: If there is $j \in [k] \setminus \{i\}$ such that $H_j \cap N_{G_t}(\tilde{v}) \neq \emptyset$ then set $\Pi(\tau) = -\infty$. The correctness of this step is justified by the fact that for any τ-good coloring c_τ, a vertex in $H_j \cap N_{G_t}(\tilde{v})$ cannot be happy in G_t with respect to c_τ.

If none of the above cases are applicable then we (recursively) set the value of $\Pi(\tau)$ such that

$$\Pi(\tau) = \begin{cases} 1 + \Pi(\tau') & \text{if } \tilde{v} \in H_i; \\ \Pi(\tau') & \text{if } \tilde{v} \in U_i. \end{cases} \tag{5}$$

Forget node. Suppose t is a forget node. Let t' be the unique child of t in $\mathcal{T}$ such that $X_t = X_{t'} \setminus \{\tilde{v}\}$, where $\tilde{v} \in X_{t'}$. For $i \in [k]$, let $\mathcal{P}_i = (P_1, P_2, \ldots, P_i \cup \{\tilde{v}\}, \ldots, P_k)$, i.e., the ordered partition of $X_{t'} = X_t \cup \{\tilde{v}\}$ obtained from $\mathcal{P}$ by adding $\tilde{v}$ to the set P_i. Furthermore, for the partition (H_i, U_i) of P_i in $\mathcal{H}$ let $\mathcal{H}_{i1} = (\mathcal{H} \setminus \{(H_i, U_i)\}) \cup \{(H_i \cup \{\tilde{v}\}, U_i)\}$ and $\mathcal{H}_{i2} = (\mathcal{H} \setminus \{(H_i, U_i)\}) \cup \{(H_i, U_i \cup \{\tilde{v}\})\}$, i.e., $\mathcal{H}_{i1}$ and $\mathcal{H}_{i2}$ are obtained from $\mathcal{H}$ by adding $\tilde{v}$ to the set of happy and unhappy vertices, respectively. If for some $\tilde{i} \in [k]$, we have $\tilde{v} \in c^{-1}(\tilde{i})$ then we let $\mathbb{P} = \{\mathcal{P}_{\tilde{i}}\}$, otherwise we let $\mathbb{P} = \{\mathcal{P}_i \mid i \in [k]\}$. We set the value of $\Pi(\tau)$ such that

$$\Pi(\tau) = \max_{\mathcal{P}_i \in \mathbb{P}, j \in [2]} \{\Pi(t', \mathcal{P}_i, \mathcal{H}_{ij})\}. \tag{6}$$

Join node. Suppose t is a join node. Let t_1, t_2 be the two children of t in $\mathcal{T}$. Recall that by the definition of nice tree decomposition we have $X_t = X_{t_1} = X_{t_2}$. We set $\Pi(t, \mathcal{P}, \mathcal{H})$ such that

$$\Pi(t, \mathcal{P}, \mathcal{H}) = \Pi(t_1, \mathcal{P}, \mathcal{H}) + \Pi(t_2, \mathcal{P}, \mathcal{H}) - |\cup_{i\in[k]} H_i| \tag{7}$$

Correctness. In what follows, we prove the correctness of Equation 5, 6, and 7.

Equation 5 We prove that Equation 5 correctly computes the value of $\Pi(\tau)$ when the Cases 1 and 2 are not applicable. Consider a τ-good coloring c_τ of G_t that maximizes the number of happy vertices in $V(G_t) \setminus (\cup_{i\in[k]} U_i)$, and let $H \subseteq V(G_t) \setminus (\cup_{i\in[k]} U_i)$ be the set of happy vertices in G_t with respect to c_τ. Also, let $c_{\tau'} = c_\tau|_{V(G_{t'})}$. Notice that $c_{\tau'}$ is a τ'-good coloring of $G_{t'}$. Since $G_{t'} = G_t - \{\tilde{v}\}$ therefore, all the vertices in $H \setminus \{\tilde{v}\}$ are happy in $G_{t'}$ with respect to $c_{\tau'}$. This implies that $\Pi(\tau') \geq |H \setminus \{\tilde{v}\}|$. If $\tilde{v} \in H_i$ then it must hold that $\tilde{v} \in H$ since c_τ is a τ-good coloring, and hence, we have $\Pi(\tau') \geq \Pi(\tau) - 1$. Otherwise, $\tilde{v} \in U_i$, and therefore $\tilde{v} \notin H$. This implies that $\Pi(\tau') \geq \Pi(\tau)$.

For the other direction consider a τ'-good coloring of $G_{t'}$ that maximizes the number of happy vertices in $V(G_{t'}) \setminus (\cup_{i\in[k]} U_i)$, and let $H \subseteq V(G_{t'}) \setminus (\cup_{i\in[k]} U_i)$ be the set of happy vertices in $G_{t'}$ with respect to $c_{\tau'}$. Let c_τ be a coloring of G_t such that $c_\tau|_{V(G_t \setminus \{v^\star\})} = c_{\tau'}$ and $c_\tau(\tilde{v}) = i$. Since Case 2 is not applicable therefore, all the vertices in $N_{G_t}(\tilde{v}) \cap H$ must belong to P_i. This implies that all the vertices in H are happy in G_t with respect to c_τ. Consider the case when $\tilde{v} \in H_i$. Since Case 1 is not applicable therefore, $N_{G_t}(\tilde{v}) \subseteq P_i$. This implies that for all $u \in N_{G_t}(\tilde{v})$, we have $c_\tau(u) = c_\tau(\tilde{v})$. Therefore, all the vertices in $H \cup \{\tilde{v}\}$ are happy in G_t with respect to c_τ. Moreover, c_τ is a τ-good coloring of G_t. Therefore, in this case we that $\Pi(\tau) \geq \Pi(\tau') + 1$. Next, consider the case when $\tilde{v} \in U_i$. Observe that c_τ is a τ-good coloring of G_t. This together with the fact that all the vertices in H are happy in G_t with respect to c_τ, implies that $\Pi(\tau) \geq \Pi(\tau')$.

Equation 6 We prove that Equation 6 correctly computes the value of $\Pi(\tau)$. Consider a τ-good coloring c_τ of G_t that maximizes the number of happy vertices in $V(G_t) \setminus (\cup_{i \in [k]} U_i)$, and let $H \subseteq V(G_t) \setminus (\cup_{i \in [k]} U_i)$ be the set of happy vertices in G_t with respect to c_τ. Let $\tilde{i} = c_\tau(\tilde{v})$. Furthermore, let $\tilde{j} = 1$ if $\tilde{v} \in H$ and $\tilde{j} = 2$, otherwise. Consider the tuple $\tau' = (t', \mathcal{P}_{\tilde{i}}, \mathcal{H}_{\tilde{i}\tilde{j}})$. By definition of τ' it holds that c_τ is a τ'-good coloring of $G_{t'} = G_t$. Furthermore, all the vertices in H are happy in $G_{t'}$ with respect to c_τ. This implies that $\Pi(\tau) \leq \max_{\mathcal{P}_i \in \mathbb{P}, j \in [2]}\{\Pi(t', \mathcal{P}_i, \mathcal{H}_{ij})\}$.

For the other direction consider $\mathcal{P}_i \in \mathbb{P}$, $j \in [2]$, and let the corresponding tuple be $\tau' = (t', \mathcal{P}_i, \mathcal{H}_{ij})$. Let $c_{\tau'}$ be a τ'-good coloring of $G_{t'}$ that maximizes the number of happy vertices in $V(G_{t'}) \setminus (\cup_{(H'_i, U'_i) \in \mathcal{H}_{ij}} U'_i)$, and let $H \subseteq V(G_{t'}) \setminus (\cup_{(H'_i, U'_i) \in \mathcal{H}_{ij}} U'_i)$ be the set of happy vertices in $G_{t'}$ with respect to $c_{\tau'}$. Since $G_t = G_{t'}$, therefore all the vertices in H are happy in G_t with respect to $c_{\tau'}$. Furthermore, by the definition of sets $\mathbb{P}$, $\mathcal{P}_i$, and $\mathcal{H}_{ij}$ it follows that $c_{\tau'}$ is a τ-good coloring of G_t. Therefore, $\Pi(\tau) \geq \max_{\mathcal{P}_i \in \mathbb{P}, j \in [2]}\{\Pi(t', \mathcal{P}_i, \mathcal{H}_{ij})\}$.

Equation 7 The proof that Equation 7 correctly computes $\Pi(\tau)$ follows from the fact that G_{t_1} and G_{t_2} are subgraphs of G_t, and in $G_t - X_t$ there is no edge in G_t between a vertex in $G_{t_1} - X_t$ and a vertex in $G_{t_2} - X_t$.

This concludes the description and the correctness proof for the recursive formulas for computing the values $\Pi(\cdot)$. We now move to the runtime analysis of the algorithm.

Runtime Analysis. Let (G, k, S, c) be an instance of MHV. In time $O(2^{O(\text{tw}(G))}n)$, we compute a nice tree decomposition $(\mathcal{T}', \mathcal{X}')$ of G, with r as the root node, and of width at most $w \leq 6 \cdot \text{tw}(G)$. Furthermore, the number of nodes in $\mathcal{T}$ is bounded by $O(wn)$. We then obtain a more structured tree decomposition $(\mathcal{T}, \mathcal{X})$, by adding $S^\star$ to each bag of $\mathcal{X}'$. For each node in $\mathcal{T}$ we have at most $k^{w+1}2^{k+w+1}$ many table entries. Here, we get a factor of k^{w+1} in the number of table entries instead of k^{k+w+1} because for a node $t \in \mathcal{T}$, we only consider valid ordered partition of X_t, and therefore, we do not guess the set for vertices in $X_t \cap S$. Using the recursive formula we can compute each value of $\Pi(\cdot)$ in time $O(2^{O(k + w \log k)}n^{O(1)})$. At this point of time, we cannot guarantee the runtime which linearly depends on n because we need to check the adjacency among vertices for setting the value of certain entries of the table, which using the straightforward implementation will require quadratic dependence on n. Nonetheless, we can start by computing a data structure for the graph G of treewidth at most w in time $w^{O(1)}n$ that allows performing adjacency queries in time $O(w)$ (for instance using [7] or Exercise 7.16 in [12]). Thus using this data structure we can compute all the entries of the table in time $O(2^{O(k+w \log k)}n) \in O(2^{O(k+\text{tw}(G) \log k)}n)$, which gives us the desired running time with linear dependence on n.

Theorem 24. *Let (G, k, S, c) be an instance of* MHV. *Then in time $O(2^{O(k+\text{tw}(G) \log k)}n)$ we can find the maximum of the number of happy vertices over all colorings that extent c to a coloring of $V(G)$. Here, n is the number of vertices in G.*

We note here that using the standard backtracking technique together with the fact that we have a partition of vertices into at most k parts which extends c, we can construct a coloring which achieves the maximum number of happy vertices.

We then turn our attention to the edge-variant. For convenience, we give a less detailed algorithm with the goal of only proving fixed-parameter tractability for treewidth. We believe it is possible to optimize the runtime further, but do not pursue this further. The problem is defined as follows.

WEIGHTED MHE **Parameter:** $k, \mathsf{tw}(G)$
Input: A graph G, integers k, a vertex subset $S \subseteq V(G)$, (partial) coloring $c : S \to [k]$, and a weight function $w : V(G) \to \mathbb{N}$.
Output: A coloring $\tilde{c} : V(G) \to [k]$ such that $\tilde{c}|_S = c$ maximizing the total weight of the happy edges.

Theorem 25. Weighted MHE *problem can be solved in time $k^{tw(G)} \cdot n^{O(1)}$, where n is the number of vertices of the input graph and $tw(G)$ is its treewidth.*

Proof. Let (G, w, k, S, c) be an instance of Weighted MHE, let $(\mathcal{T}, \mathcal{X})$ be a nice tree decomposition of G, and let r be the root of $\mathcal{T}$. For a node $t \in V(\mathcal{T})$, by $\mathsf{desc}(t)$ we denote the set of nodes which are descendants of t (including t) in $\mathcal{T}$. Furthermore, for $t \in V(\mathcal{T})$, by G_t we denote the graph $G[V_t]$, where $V_t = \cup_{t' \in \mathsf{desc}(t)} X_{t'}$.

For every node t of $\mathcal{T}$ we set up a table Π_{t} indexed by all possible extended full k-colorings of X_t. Intuitively, an entry of Π_{t} indexed by $f : X_t \to [k]$ gives the total weight of edges happy in $G[V_t]$ under f. It holds that an optimal solution is given by $\max_f\{\Pi_{\mathrm{t}}[f]\}$. In what follows, we detail the construction of the tables Π_{t} for every node t. The algorithm processes the nodes of $\mathcal{T}$ in a post-order manner, so when processing t, a table has been computed for all children of t.

- **Leaf node.** Let t be a leaf node and $X_t = \{v\}$. Obviously, $G[V_t]$ is edge-free, so we have $\Pi_{\mathrm{t}}[f] = 0$. Hence, Π_{t} is computed in $O(k)$ time.
- **Introduce node.** Suppose t is an introduce node. Let t' be the unique child of t in $\mathcal{T}$, and $X_t = X_{t'} \cup \{\tilde{v}\}$, where $\tilde{v} \notin X_{t'}$. It is not difficult to see that we set $\Pi_{\mathrm{t}}[f] = \Pi_{\mathrm{t'}}[f|_{X_t}] + \sum_{p \in N_h(\tilde{v})} w(p\tilde{v})$, where $N_h(\tilde{v})$ denotes the neighbors of $\tilde{v}$ colored with the same color as $\tilde{v}$. It follows Π_{t} can be computed in time $O(k^{tw(G)+1})$.
- **Forget node.** Suppose t is a forget node. Let t' be the unique child of t in $\mathcal{T}$ such that $X_t = X_{t'} \setminus \{\tilde{v}\}$, where $\tilde{v} \in X_{t'}$. Observe that the graphs $G[V_t]$ and $G[V_t']$ are the same. Thus, we set $\Pi_{\mathrm{t}}[f]$ to the maximum of $\Pi_{\mathrm{t'}}[f']$ where $f'|_{X_t} = f$. Since there are at most k such colorings f' for each f, we compute Π_{t} in time $O(k^{tw(G)+2})$.
- **Join node.** Suppose t is a join node. Let t_1, t_2 be the two children of t in $\mathcal{T}$. Recall that by the definition of nice tree decomposition we have $X_t = X_{t_1} = X_{t_2}$. The properties of a tree decomposition guarantee that $V(G[V_{t_1}]) \cap V(G[V_{t_2}]) = X_t$, and that no vertex in $V(G[V_{t_1}]) \setminus X_t$ is adjacent to a vertex in $V(G[V_{t_2}]) \setminus X_t$. Thus, we add together weights of happy edges that appear in $G[V_{t_1}]$ and $G[V_{t_2}]$, while subtracting a term guaranteeing we do not add weights of edges that are happy in both subgraphs. Indeed, we set $\Pi_{\mathrm{t}}[f] = \Pi_{\mathrm{t_1}}[f] + \Pi_{\mathrm{t_2}}[f] - q$, where q is the total weight of the edges made happy under f in X_t. The table $\Pi_{\mathrm{t}}[f]$ can also be computed in time $O(k^{tw(G)+2})$.

To summarize, each table Π_{t} has size bounded by $k^{tw(G)+1}$. Moreover, as each table is computed in $O(k^{tw(G)+2})$ time, the algorithm runs in $k^{tw(G)} \cdot n^{O(1)}$ time, which is what we wanted to show. ☐

7.2 Neighborhood Diversity

The polynomial-time solvability of a problem on bounded treewidth graphs implies the existence of a polynomial-time algorithm also for other structural parameters that are polynomially upper-bounded in treewidth. For instance, one such parameter is the *vertex cover number*, i.e., the size of a smallest vertex cover that a graph has. However, graphs with bounded vertex cover number are highly restricted, and it is natural to look for less restricting parameters that generalize vertex cover (like treewidth). Another parameter generalizing vertex cover is *neighborhood diversity*, introduced by Lampis [24]. Let us first define the parameter, and then discuss its connection to both vertex cover and treewidth.

Definition 26. *In an undirected graph G, two vertices u and v have the same type if and only if $N(u) \setminus \{v\} = N(v) \setminus \{u\}$.*

Definition 27 (Neighborhood diversity [24]). *A graph G has neighborhood diversity t if there exists a partition of $V(G)$ into t sets $P_1, P_2, \ldots, P_t$ such that all the vertices in each set have the same type. Such a partition is called a type partition. Moreover, it can be computed in linear time.*

Note that all the vertices in P_i for every $i \in [t]$ have the same neighborhood in G. Moreover, each P_i either forms a clique or an independent set in G.

Neighborhood diversity can be viewed as representing the simplest of dense graphs. If a graph has vertex cover number d, then the neighborhood diversity of the graph is not more than $2^d + d$ (for a

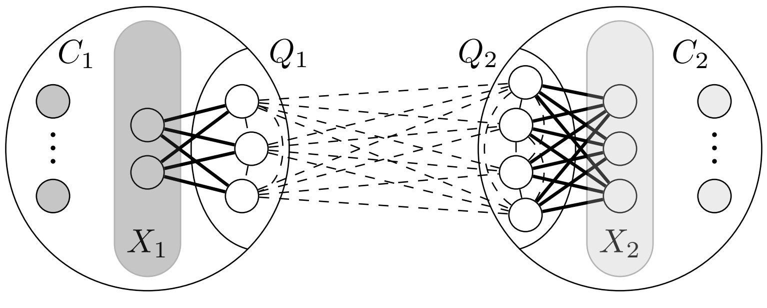

Figure 3: A set of a type partition, where each vertex in $Q_1 \cup Q_2$ has the same type. The dashed edges appear exactly when $Q_1 \cup Q_2$ induces a clique. The set Q_1 forms a complete bipartite graph with both X_1 and X_2; likewise for Q_2 (edges omitted for brevity).

proof, see [24]). Hence, graphs with bounded vertex cover number also have bounded neighborhood diversity. However, the converse is not true since complete graphs have neighborhood diversity 1. Paths and complete graphs also show that neighborhood diversity is incomparable with treewidth. In general, some NP-hard problems (some of which remain hard for treewidth), are rendered tractable for bounded neighborhood diversity (see e.g., [21]).

We proceed to present algorithms for MHE and MHV for graphs of bounded neighborhood diversity. Consider a type partition of a graph G with t sets, and an instance of $I = (G, k, S, c)$ of MHE (MHV). If a set contains both precolored and uncolored vertices, we split the set into two sets: one containing precisely the precolored vertices and the other precisely the uncolored vertices. After splitting each set, the number of sets is at most $2t$. For convenience, we say a set is *uncolored* if each vertex in it is uncolored; otherwise the set is *precolored*. Let the uncolored sets be $P_1, P_2, \ldots, P_t$. In what follows, we discuss how vertices in these sets are colored in an optimal solution. We say a set is *monochromatic* if all of its vertices have the same color.

MHE **Parameter:** neighborhood diversity t
Input: A graph G, an integer k, a vertex subset $S \subseteq V(G)$, and a (partial) coloring $c : S \subseteq V(G) \to [k]$.
Output: A coloring $\tilde{c} : V(G) \to [k]$ such that $\tilde{c}|_S = c$ maximizing the number of happy edges.

Lemma 28. *There is an optimal extended full coloring for an instance I of* MHE *such that each uncolored set P_i for $1 \leq i \leq t$ is monochromatic.*

Proof. Consider any optimal extended full coloring for an instance I. Suppose the vertices in a set P_i belong to more than one color class. Let Q_1 and Q_2 be the (disjoint and non-empty) sets of vertices of P_i belonging to color classes C_1 and C_2, respectively. Let X_1 and X_2 be the neighbors of the vertices in Q_1 and Q_2 in color classes C_1 and C_2, respectively, as shown in Figure 3. Without loss of generality, let us assume that $|X_1| \leq |X_2|$. By recoloring vertices in Q_1 with the color of C_2, we retain an optimal solution without disturbing the colors of other vertices. If $|E(Q_1, Q_2)|$ is the number of edges between Q_1 and Q_2, the gain in the number of happy edges by recoloring Q_1 is $|E(Q_1, Q_2)| + |Q_1|(|X_2| - |X_1|)$, which is strictly positive if P_i is a clique and non-negative if P_i is an independent set.

In conclusion, we have shown that every optimal extended full coloring makes each P_i inducing a clique monochromatic. Moreover, there is an optimal extended full coloring making each P_i inducing an independent set monochromatic. □

The previous lemma is combined with the algorithm of Theorem 8 to obtain the following.

Theorem 29. *For any $k \geq 1$,* MHE *can be solved in time $O^*(2^t)$, where t is the neighborhood diversity of the input graph.*

Proof. First we construct a weighted graph H from G as follows: merge each uncolored set into a single vertex. Within a precolored set (i.e., a set that is not uncolored), merge vertices of the same color. This merging operation may create parallel edges and self-loops in H. Discard all self-loops in H. Replace all parallel edges with a single weighted edge with weight equivalent to the number edges between the corresponding vertices. Edges between the vertices in G that are merged to the same vertex are treated as happy, as there is an optimal extended full coloring where the merged vertices are colored the same by Lemma 28. Clearly, H has at most $t + kt$ vertices in which t vertices are uncolored.

Now, MHE on G is converted to an instance of Weighted MHE on H. By using Theorem 8, we can solve the instance of MHE on G in time $O^*(2^t)$. □

Using similar arguments, we get the following results for MHV as well.

MHV **Parameter:** neighborhood diversity t
Input: A graph G, an integer k, a vertex subset $S \subseteq V(G)$, and a (partial) coloring $c : S \subseteq V(G) \to [k]$.
Output: A coloring $\tilde{c} : V(G) \to [k]$ such that $\tilde{c}|_S = c$ maximizing the number of happy vertices.

Lemma 30. *There is an optimal extended full coloring for an instance I of* MHV *such that each uncolored set P_i for $1 \leq i \leq t$ is monochromatic.*

Proof. Consider any optimal extended full coloring for an instance I. Suppose the vertices in a set P_i belong to more than one color class. Let Q_1 and Q_2 be the (disjoint and non-empty) sets of vertices of P_i belonging to color classes C_1 and C_2, respectively. Let X_1 and X_2 be the neighbors of the vertices in Q_1 and Q_2 in color classes C_1 and C_2, respectively, as shown in Figure 3. Without loss of generality, let us assume that $|X_1| \leq |X_2|$. By recoloring vertices in Q_1 with the color of C_2, we retain an optimal solution without disturbing the colors of other vertices. If $|E(Q_1, Q_2)|$ is the number of edges between Q_1 and Q_2, the gain in the number of happy edges by recoloring Q_1 is $|E(Q_1, Q_2)| + |Q_1|(|X_2| - |X_1|)$, which is strictly positive if P_i is a clique and non-negative if P_i is an independent set.

In conclusion, we have shown that every optimal extended full coloring makes each P_i inducing a clique monochromatic. Moreover, there is an optimal extended full coloring making each P_i inducing an independent set monochromatic. □

Theorem 31. *For any $k \geq 1$,* MHV *can be solved in time $O^*(2^t)$, where t is the neighborhood diversity of the input graph.*

Proof. First we construct a weighted graph H from G as follows: merge each uncolored set into a single vertex. Within a precolored set (i.e., a set that is not uncolored), merge vertices of the same color. This merging operation may create parallel edges and self-loops in H. Discard all self-loops in H. Replace all parallel edges with a single weighted edge with weight equivalent to the number edges between the corresponding vertices. Edges between the vertices in G that are merged to the same vertex are treated as happy, as there is an optimal extended full coloring where the merged vertices are colored the same by Lemma 28. Clearly, H has at most $t + kt$ vertices in which t vertices are uncolored.

Now, MHV on G is converted to an instance of Weighted MHV on H. By using Theorem 8, we can solve the instance of MHV on G in time $O^*(2^t)$. □

7.3 Distance to Clique

Let us then consider another natural parameter, the distance to clique.

Theorem 32. MHE *and* MHV *are both FPT when parameterized by the size of a clique modulator of the input graph.*

Proof. Let (G, k, S, c) denote an instance of MHV. We will use Q to refer to $V(G) \setminus S$, the set of vertices that are not precolored by c. Further, let $X \subseteq V(G)$ such that $C = G \setminus X$ is a clique. We use d to denote $|X|$.

If $k > d + 1$, then there exists at least two vertices u and v in C such that $c(u) \neq c(v)$, which implies that no vertex of C is happy. First we guess the partition (H, U) of X in $O(2^d)$ time, where H and U denote the happy and unhappy vertices of X in an optimal coloring c'.

Let $H = (H_1, \ldots, H_t)$ be the partition of H such that all vertices in set H_i, $i \in [t]$, are colored with the same color by c'. We can guess the correct partition in $O(d^d)$ time. Note that $N(H_i) \cap N(H_j) = \emptyset$. Indeed, suppose not, and let $v \in N(H_i) \cap N(H_j)$. Then, note that irrespective of how v is colored, at least one vertex in H_i or H_j is not happy, a contradiction.

Since H_i is happy, there are no two vertices u and v in the set $H_i \cup N(H_i)$ such that $c(u) \neq c(v)$. For each $i \in [t]$, if at least one vertex is precolored in $H_i \cup N(H_i)$ then assign the same color to all vertices of $H_i \cup N(H_i)$. For the sets $H_i \cup N(H_i)$, which do not have any precolored vertices, assign a color which is not used so far to color any H_i. At the end arbitrarily color the remaining vertices. For each possible partition H and U of X and each possible partition $H_1, \ldots, H_t$ of H, count the number of happy vertices and the optimal coloring c' is the coloring which maximizes the number of happy vertices.

If $k \leq d+1$, then some of the clique vertices can be happy, but this only happens when all the clique vertices are colored by the same color. Since there are at most k colors we can guess the correct coloring of the clique in $O(k)$ time. It remains to color the set X. Since $k \leq d+1$, we can guess the coloring of X in $O(d^{d+1})$ time.

We now turn to the MHE problem.

Overall Approach. We begin with an informal discussion of the main steps in our algorithm. First, we show that if every vertex in the clique $G \setminus X$ is precolored by c, then an optimal coloring c' can be found in FPT time by guessing the behavior of c' on the modulator and then reducing the remaining problem to finding a maximum-weight matching in an auxiliary bipartite graph. On the other hand, suppose the clique is not entirely precolored. Then we split into two cases based on whether the uncolored part is large or small. If the uncolored part is small, then we can combine it with the modulator X and think of the union as a larger modulator, and we are back to the previous case. The key idea of our algorithm lies in analyzing the remaining case: here we show that if the uncolored part of the clique is large enough, then it "makes sense" for any optimal coloring to color all of it with the same color. With this fact at hand, we can now guess the common color employed on the large uncolored portion of the clique and then proceed as in the first case. We now turn to a formal description of these ideas.

Precolored Clique. First, we give a procedure to color X when all the vertices in the clique $C = G \setminus X$ are precolored. Without loss of generality, let no vertex of X be precolored (otherwise we adapt our argument to work with the uncolored part of X). To begin with, we guess $X = (X_1, \ldots, X_t)$, which is the partition of X such that all vertices in X_i are colored in the same way by c'.

Let $w[i,j]$ denote the number of edges in $G[X_i \cup C]$ that will be happy if all vertices of X_i are colored j. Consider an auxiliary weighted bipartite graph, denoted by $D = ((A, B), E(G))$ with edge weights given by $w : E(G) \to [|E(G)|]$. This graph is constructed as follows. The vertex set A contains one vertex for every set X_i and the vertex set B contains one vertex for every color used in the clique. The weight of the edge $a_i b_j$ is simply $w[i,j]$. We find a matching M of maximum weight in D. It is easy to see that any such matching saturates A, since $|B| \geq |A|$, the weights are positive, and all edges are present. We now color all vertices in X_i based on the matching M. In particular, if M matches a_i to b_j, then we color all vertices in X_i with color j. At the end of this phase, all vertices in X have been colored.

Now, we turn to the general situation. Let C_U be the set of uncolored vertices in the clique. We say that a color j is *active* if there exists a vertex $v \in X$ such that $c(v) = j$, while a color is *dormant* otherwise. In other words, a dormant color is a color that is employed only on vertices outside the modulator. We use q to denote the number of active colors, and define $\ell = q + 1$. Note that $\ell \leq d + 1$.

Few Uncolored Vertices. If $|C_U| \leq 2d\ell = O(d^2)$, then $X' = X \cup C_U$ is a modulator to clique C' of size at most $O(d^2)$, i.e., $G \setminus (X \cup C_U)$ is a clique. Since all the vertices of the clique $G \setminus X'$ are precolored, the vertices of X' can be colored using the procedure described above. So without loss of generality we assume that $|C_U| > 2d\ell$.

Many Uncolored Vertices. We now argue that if C_U is "sufficiently large", then there exists an optimal coloring c' that uses one color among all vertices of C_U. We first begin with a technical claim, which shows how we can modify a coloring using two colors on C_U into one that uses just one, while maintaining at least as many happy edges as the original coloring. This will be useful later.

Claim 4. *Let (G, k, S, c, X) be an instance of MHE, and let C_U denote the subset of $G \setminus X$ that is not colored by the precoloring c. Let $c^\star$ be a coloring extending c that uses two colors amongst the vertices of C_U, and let η denote the number of edges that are happy with respect to $c^\star$. If $|C_U| > 4d$, there also exists an coloring c' extending c that: (i) assigns the same color to all vertices of C_U, and (ii) makes at least η edges happy.*

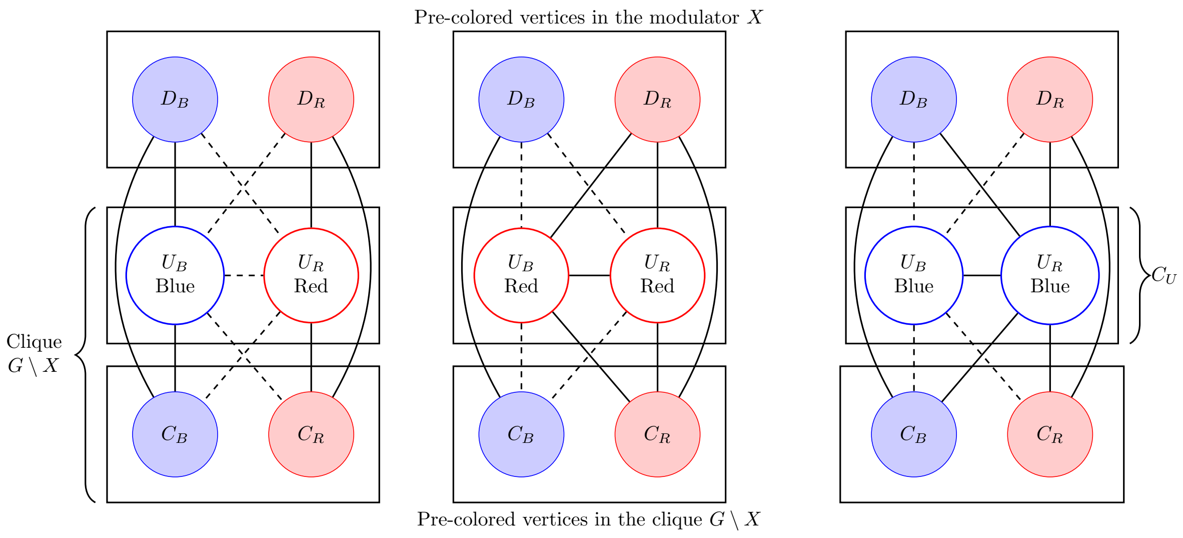

Figure 4: A given coloring using two colors in C_U (left). The two figures (center, right) show the effect of switching one of the colors to the others. The solid edges denote happy edges while the dashed edges are the unhappy ones. The edge set labels have been omitted for clarity.

Proof. We argue by contradiction. Let the two colors employed by $c^\star$ among the vertices of C_U be red and blue. For the rest of this discussion, assume that all vertices of G are colored according to $c^\star$, the given coloring. We let D_R and U_R denote the set of red vertices in the modulator X and C_U, and use d_R and u_R, respectively, to denote their sizes. Further, let C_R denote the set of precolored vertices in the clique that are colored red, and let c_R denote the size of C_R. We use analogous notation (i.e., D_B, U_B, and C_B for the sets; and d_B, u_B, c_B for the sizes) for the blue vertices. See Figure 4 for an illustration. Finally, let

- e_R (respectively, e_B) denote the number of edges in $G[D_R]$ (respectively, $G[D_B]$);
- e_{RR} denote the number of edges between D_R and U_R;
- e_{RB} denote the number of edges between D_R and U_B;
- e_{BR} denote the number of edges between D_B and U_R;
- e_{BB} denote the number of edges between D_B and U_B;
- f_{RR} denote the number of edges between D_R and C_R; and
- f_{BB} denote the number of edges between D_B and C_B.

For convenience, we also set

$$\Delta := \binom{u_R}{2} + \binom{u_B}{2} + \binom{c_R}{2} + \binom{c_B}{2} + f_{BB} + f_{RR} + e_R + e_B.$$

The number of happy edges whose endpoints are either both red or both blue, with respect to the coloring $c^\star$, is given by

$$\alpha = e_{RR} + e_{BB} + c_R \cdot u_R + c_B \cdot u_B + \Delta.$$

Without loss of generality, assume that $c_R \geq c_B$. Further, note that since $u_R + u_B = |C_U| \geq 4d$, either $u_R > 2d$ or $u_B > 2d$. We consider these cases individually.

Case 1: $u_R > 2d$**.** Consider an alternate coloring c', which is identical to $c^\star$ except that it colors all vertices in U_B red. The number of happy edges with respect to c' is given by:

$$\beta = u_R \cdot u_B + e_{RR} + e_{RB} + c_R \cdot u_R + c_R \cdot u_B + \Delta.$$

Since $c^\star$ is an optimal coloring and the number of happy edges that do not involve the colors red or blue are exactly the same between the colorings $c^\star$ and c', we must have $\alpha \geq \beta$, which in turn implies

$$e_{BB} + c_B \cdot u_B \geq u_R \cdot u_B + e_{RB} + c_R \cdot u_B.$$

Since $c_R \geq c_B$, we have that

$$e_{BB} \geq u_R \cdot u_B + e_{RB},$$

implying that $d_B \geq u_R$. However, since $u_B > 2d$ and $d_B \leq d$, this is a contradiction.

Case 2: $u_B > 2d$**.** Consider again the alternate coloring c' described above, and recall that we had

$$e_{BB} + u_B \cdot c_B \geq u_R \cdot u_B + e_{RB} + c_R \cdot u_B.$$

Note that $e_{BB} \leq d \cdot u_B$ and $u_R \cdot u_B + e_{RB} \geq 0$, we simplify the above to

$$d \cdot u_B + u_B \cdot c_B \geq c_R \cdot u_B,$$

implying that $c_R \leq c_B + d$. Now consider an alternate coloring c'', which is identical to $c^\star$ except that it colors all vertices in U_R blue. The number of happy edges with respect to c'' is given by

$$\gamma = u_R \cdot u_B + e_{BB} + e_{BR} + c_B \cdot u_B + c_B \cdot u_R + \Delta.$$

Again, by the optimality of $c^\star$, we have $\alpha \geq \gamma$, which in turn implies

$$e_{RR} + c_R \cdot u_R \geq u_R \cdot u_B + e_{BR} + c_B \cdot u_R.$$

Note that $e_{RR} \leq d \cdot u_R$ and $e_{BR} \geq 0$, so the expression above simplifies to

$$d \cdot u_R + c_R \cdot u_R \geq u_R \cdot u_B + c_B \cdot u_R,$$

implying that

$$d + c_R \geq u_B + c_B \implies u_B \leq c_R - c_B + d \leq c_B - c_B + 2d = 2d.$$

The last inequality follows from the conclusion we obtained above just before defining the coloring c''. However, note that the situation above, which is that $u_B \leq 2d$, contradicts the case that we are in, and this concludes the argument. □

We now turn to the main claim regarding the coloring of large cliques.

Claim 5. *Let (G, k, S, c, X) be an instance of MHE as described above, and let C_U denote the subset of $G \setminus X$ that is not colored by the precoloring c. If $|C_U| > 2d\ell$, then there exists an optimal coloring c' extending c that assigns the same color to all vertices of C_U.*

Proof. We argue by induction on ℓ. Recall that $\ell = q + 1$, where q is the number of active colors. The base case is when $\ell = 1$. Herein, the precoloring only employs dormant colors. Thus the claim is that there exists an optimal coloring which uses at most one dormant color among vertices in C_U. Let c' be an optimal coloring extending c. If c' colors vertices of C_U with one color, there is nothing to prove. On the other hand, suppose the coloring uses two dormant colors on the vertices of C_U, say red and green. Let p_r and p_g be the number of red and green vertices in the precoloring (note that all of these vertices belong to the clique), while we use q_r and q_g to denote the number of red and green vertices in C_U with respect to c'. Without loss of generality, let $p_r \geq p_g$. Let c'' denote the coloring that is the same as c' except that it uses the color red for all vertices in C_U that were colored green by c'. In other words, we started with c' and "switched" from green to red in C_U to obtain c''. Note that the number of happy edges whose endpoints are either both green or both red are in c' and c'' are, respectively, given by $h' := p_r \cdot q_r + p_g \cdot q_g$ and $h'' := p_r \cdot q_r + p_r \cdot q_g$, since there are no red or green vertices in the modulator. Note that $h'' \geq h'$ since $p_r \geq p_g$. Repeating this argument for any additional dormant colors, we can overwrite the use dormant colors to a point where we obtain an optimal coloring that uses at most one dormant color.

We now turn to the induction step. Assume that the induction hypothesis holds for all values $\ell' < \ell$. Let c' be an optimal coloring extending c. If c' colors vertices of C_U with one color, we are done. Otherwise, let $V_1, \ldots, V_p$ be the color classes of C_U with respect to c', and let these color classes be listed in decreasing order of size (i.e., V_p is the smallest color class amongst $V_1, \ldots, V_p$). We also use ω_i to denote the color of the vertices in V_i. Note that we may assume that $p \leq \ell$. Indeed, otherwise we are in a situation where the coloring on C_U makes use of more than one dormant color, and we can modify c' to a coloring that makes use of exactly one dormant color by the argument described for the base case. Without loss of generality, let $p = \ell$.

Now, if $|V_p| \leq 2d$, then

$$\left| \bigcup_{i=1}^{p-1} V_i \right| \geq 2d\ell - 2d = 2d(\ell - 1), \text{ since } \left| \bigcup_{i=1}^{p} V_i \right| \geq 2d\ell$$

while if $|V_p| > 2d$, then

$$\left| \bigcup_{i=1}^{p-1} V_i \right| \geq 2d(\ell - 1), \text{ since } |V_i| \geq |V_p| \ \forall i \in [p-1].$$

In either situation, we apply the induction hypothesis to the subgraph H of G that consists of all vertices not colored with color ω_p according to c'. Let $c^\star$ be the projection of the precoloring c onto H, and let $k^\star$, $S^\star$ be defined analogously. Also, let $X^\star := X \setminus \{v \in X \mid c'(v) = \omega_i\}$. Note that $X^\star$ continues to be a clique modulator for the graph H and that $|X^\star| \leq d$. It is straightforward to check that the induction hypothesis holds in the instance $(H, k^\star, S^\star, c^\star, X^\star)$, since the uncolored clique in this instance is given by $\bigcup_{i=1}^{p-1} V_i$, whose size bound was argued above.

By the induction hypothesis, there exists an optimal coloring $c^\star$ of $H^\star$ that uses one color on the uncolored vertices of $H^\star \setminus X^\star$. Now consider the coloring c'' defined such that

$$c''(v) = \begin{cases} c^\star(v) & \text{if } v^\star \in V(H^\star), \\ c'(v) & \text{otherwise.} \end{cases}$$

It is easy to verify that c'' has at least as many happy edges as c' and has the property that it uses two colors on the vertices of C_U. Using Claim 4 (which applies since $\ell \geq 2$), we can now adapt c'' to arrive at a coloring with at least as many happy edges as c'' but that uses one color on the vertices of C_U. This completes the argument. □

To summarize, the case when $|C_U| \leq 2d\ell = O(d^2)$ is easy, since we can guess optimal coloring of X in time $d^{O(d^2)}$ and then it can be easily extended to color the entire graph. If $|C_U| > 2d\ell$, from Claim 5 we know that all vertices of C_U get the same color, which we guess in $O(k)$ time. Now we need to color X such that the number of happy edges is maximized. This can be done by simply applying the procedure described in the first case, where all vertices of the clique are precolored. The running time of the algorithm is $O(k(d^{d^2})(m\sqrt{n}\log n)(n+m))$, where $O(m\sqrt{n}\log n)$ is the time needed to compute a maximum matching on a bipartite graph [15]. □

8 Happy Coloring on Special Graph Classes

We begin this section by proving hardness of both DMHE and DMHV for bipartite graphs and split graphs. Afterwards, we show the vertex-variant can be solved optimally on cographs.

8.1 Bipartite and Split Graphs

Theorem 33. DMHE *is* NP-complete *on the class of bipartite and split graphs.*

Proof. We first consider the case of bipartite graphs. We reduce from DMHE on general graphs.

We let (G, k, ℓ, S, c) be an instance of DMHE. Construct a bipartite graph $(H = (A, B), E(G))$ as follows. For every vertex $v \in V(G)$, we introduce a vertex $a_v \in A$. For every edge $e \in E(G)$, we introduce a vertex $b_e \in B$, and if $e = (u, v)$, then b_e is adjacent to a_u and a_v. The precoloring function c' mimics c on A, that is, for every $u \in S$, $c'(a_u) = c(u)$. We use X to denote $\{a_u \mid u \in S\} \subseteq A$. Let $\ell' = m + \ell$. Thus our reduced instance is (H, k, ℓ', X, c').

We now argue the equivalence. First, consider the forward direction. If $\tilde{c}$ is a total coloring of $V(G)$ that makes ℓ edges happy, then we define a coloring $c^\star$ for H as follows: color $c^\star(a_v) := \tilde{c}(v)$ for all $a_v \in A$. For every edge $e = (u, v) \in E(G)$, color $b_e \in B$ according to $\tilde{c}(u)$. Note that for all edges e in G that are happy with respect $\tilde{c}$, two edges (namely (b_e, a_v) and (b_e, a_u)) are happy with respect to $c^\star$. Corresponding to all unhappy edges, H has one happy edge with respect to $c^\star$. Therefore, the total number of happy edges in $c^\star$ is $2\ell + (m - \ell) = m + \ell$.

In the reverse direction, let $c^\star$ be a coloring of H that makes at least $m+\ell$ edges happy. Now consider the coloring $\tilde{c}$ obtained as follows: $\tilde{c}(u) = c^\star(a_u)$. We argue that at least ℓ edges are happy in G with respect to $\tilde{c}$. Indeed, suppose not. Without loss of generality, assume that only $(\ell - 1)$ edges are happy with respect to $\tilde{c}$. Then in H, there are at most $(\ell - 1)$ vertices in B that can have two happy edges incident on them, and therefore the total number of happy edges is at most $2(\ell-1)+(m-\ell+1) = m+\ell-1$, which contradicts our assumption about the total number of happy edges in H with respect to $c^\star$.

We now turn to the case of split graphs. The construction is similar to the case of bipartite graphs. Construct a split graph $(H = (A, B), E(G))$ as follows. Let (G, k, ℓ, S, c) be an instance of DMHE. Let $T := \binom{m}{2} + 1$. For every vertex $v \in V(G)$, we introduce T copies of the vertex $a_v \in A$, denoted by $a_v[1], \ldots, a_v[T]$. For every edge $e \in E(G)$, we introduce a vertex $b_e \in B$, and if $e = (u, v)$, then b_e is adjacent to all copies of a_u and a_v. Finally, we add all edges among vertices in B, thereby making $H[B]$ a clique.

The precoloring function c' mimics c on A across all copies, that is, for every $u \in S$, $c'(a_u) = c(u)$ for all copies of a_u. We use X to denote $\{a_u[i] \mid u \in S, 1 \leq i \leq T\}$ and for $1 \leq i \leq T$, $A_i := \{a_u[i] \mid u \in V(G)\}$. Let $\ell' = T(m + \ell)$. Thus our reduced instance is (H, k, ℓ', X, c').

We now argue the equivalence of these instances. First, consider the forward direction. If $\tilde{c}$ is a total coloring of $V(G)$ that makes ℓ edges happy, then we define a coloring $c^\star$ for H as follows: color $c^\star(a_v) := \tilde{c}(v)$ for all copies of $a_v \in A$. For every edge $e = (u, v) \in E(G)$, color $b_e \in B$ according to $\tilde{c}(u)$. Note that for all edges e in G that are happy with respect $\tilde{c}$, $2T$ edges (namely (b_e, a_v) and (b_e, a_u) across all copies) are happy with respect to $c^\star$. Corresponding to all unhappy edges, H has T happy edges with respect to $c^\star$. Therefore, the total number of happy edges in $c^\star$ is at least $2T\ell + T \cdot (m - \ell) = T \cdot (m + \ell)$.

In the reverse direction, let $c^\star$ be a coloring of H that makes at least $T(m+\ell)$ edges happy. We argue that there must be at least one copy A_i of $\{a_v \mid v \in V(G)\}$ for which the number of happy edges with one endpoint in B and one in A_i is at least $(m + \ell)$. Indeed, suppose not. Then consider the following partition of the edges in H: let E_0 be all edges with both endpoints in B, and let E_i be all edges with one endpoint in B and the other endpoint in A_i. For the sake of contradiction, we have assumed that the number of happy edges in E_i is less than $(m + \ell)$ for all $i \in [T]$. Note that the total number of edges in B is $\binom{m}{2}$. Therefore, the the number of edges happy with respect to $c^\star$ is at most:

$$\binom{m}{2} + T \cdot (m + \ell - 1) = T \cdot (m + \ell) + \binom{m}{2} - T < T(m + \ell),$$

where the last step follows by substituting for $T = \binom{m}{2} + 1$. This leads to the desired contradiction.

Having identified one set E_i that has at least $(m + \ell)$ happy edges, the argument for recovering a coloring $\tilde{c}$ for G that makes at least ℓ edges happy is identical to the case of bipartite graphs. □

Theorem 34. DMHV *is* NP-complete *on the class of bipartite and split graphs.*

Proof. We first consider the case of split graphs. Let $I = (G, k, \ell, S, c)$ be an instance of DMHE, and let us in polynomial time construct an instance $I' = (G', k, \ell, S, c')$ of DMHV. We can safely (and crucially) assume at least two vertices of G are precolored (in distinct colors), for otherwise the instance is trivial. We construct the split graph $G' = (C \cup B, E' \cup E'')$, where

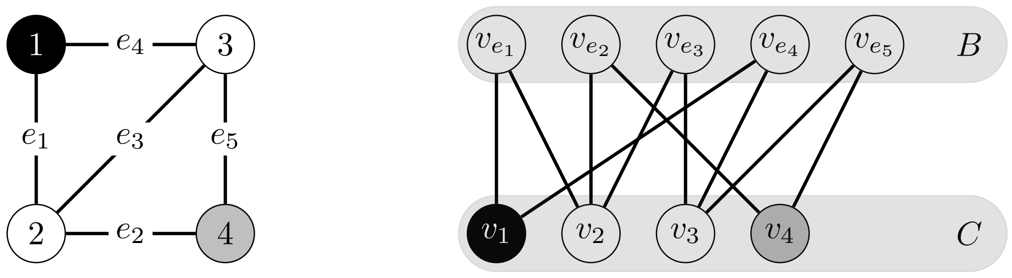

Figure 5: **(a)** A graph G of an instance of DMHE, where white vertices correspond to uncolored vertices. **(b)** The graph G transformed into a split graph G' by the construction of Theorem 34. The edges between the vertices in C are not drawn.

- $C = \{v_x \mid x \in V(G)\}$,
- $B = \{v_e \mid e \in E(G)\}$,
- $E' = \{v_e v_x \mid e \text{ is incident to } x \text{ in } G\}$, and
- $E'' = \{v_x v_{x'} \mid x, x' \in V(G)\}$.

That is, C forms a clique and B an independent set in G', proving G' is a split graph. In particular, observe that the degree of each vertex v_e is two. To complete the construction, we retain the precoloring, i.e., set $c'(v_x) = c(x)$ for every $x \in V(G)$. The construction is illustrated in Figure 5.

We claim that I is a YES-instance of DMHE iff I' is a YES-instance of DMHV. Suppose ℓ edges can be made happy in G by an extended full coloring of c. Consider an edge $e \in E(G)$ whose endpoints are colored with color i. To make ℓ vertices happy in G', we give v_e and its two neighbors the color i. For the other direction, suppose ℓ vertices are happy under an extended full coloring of c'. As at least two vertices in C are colored in distinct colors, it follows that all the happy vertices must be in B. Furthermore, the vertices in B correspond to precisely the edges in $E(G)$, so we are done.

For bipartite graphs, the construction is a modification of the construction for split graphs. Modify the split graph G' by deleting the edges between the vertices in C, i.e., let $G' = (C \cup B, E')$. For each $v_x \in C$, add a path $S_{v_x} = \{v_x^1, v_x^2, v_x^3\}$ along with the edges $v_x v_x^1$ and $v_x^3 v_x$. In other words, each v_x forms a 4-cycle with the vertices in S_{v_x}. Clearly, we have that G' is bipartite as it contains no odd cycles. Arbitrarily choose three distinct colors from $[k]$, and map them bijectively to S_{v_x}. Observe that by construction, none of the vertices in S_{v_x} can be happy under any c' extending c. This completes the construction. Correctness follows by the same argument as in case of split graphs. $\square$

8.2 Cographs

We now turn to the MHV problem on the class of cographs. We use the notion of modular decomposition to solve this problem on cographs.

A set $M \subseteq V(G)$ is called *module* of G if all vertices of M have the same set of neighbors in $V(G) \setminus M$. The *trivial modules* are $V(G)$, and $\{v\}$ for all v. A prime graph is a graph in which all modules are trivial. The modular decomposition of a graph is one of the decomposition techniques which was introduced by Gallai [20]. The *modular decomposition* of a graph G is a rooted tree M_G that has the following properties:

1. The leaves of M_G are the vertices of G.
2. For an internal node h of M_G, let $M(h)$ be the set of vertices of G that are leaves of the subtree of M_G rooted at h. ($M(h)$ forms a module in G).
3. For each internal node h of M_G there is a graph G_h (*representative graph*) with $V(G_h) = \{h_1, h_2, \ldots, h_r\}$, where $h_1, h_2, \ldots, h_r$ are the children of h in M_G and for $1 \le i < j \le r$, h_i and h_j are adjacent in G_h iff there are vertices $u \in M(h_i)$ and $v \in M(h_j)$ that are adjacent in G.
4. G_h is either a clique, an independent set, or a prime graph and h is labeled *Series* if G_h is clique, *Parallel* if G_h is an independent set, and *Prime* otherwise.

James et al. [22] gave the first polynomial-time algorithm for modular decomposition which runs in $O(n^4)$ time.

Theorem 35. MHV *is polynomial-time solvable on the class of cographs.*

Proof. The modular decomposition tree of cographs has only parallel and series nodes. Let G be a cograph whose modular decomposition tree is M_G. Without loss of generality we assume that the root r of tree M_G is a series node, otherwise G is not connected and the number of happy vertices in G is equal to the sum of the happy vertices in each connected component. Let the children of r be x and y. Further, let the cographs corresponding to the subtrees at x and y be G_x and G_y, respectively.

We assume that $k \geq 3$, for otherwise we use the polynomial-time algorithm for 2-MHV on general graphs to find the number of maximum happy vertices. Further, we assume that both G_x and G_y contain at least two vertices. Suppose, if G_x has only vertex v, then v is the universal (adjacent to all vertices) vertex in G. It is easy to see that in any optimal coloring $\tilde{c}$ of G, if a vertex u is happy then $\tilde{c}(u) = \tilde{c}(v)$ i.e, in any optimal coloring all happy vertices are colored with color of v. We can guess the color of v in $O(k)$ time.

We have the following three cases based on the number of colors present in G_x and G_y in the partial coloring $\tilde{c}$ of G.

Number of colors in G_x is zero. Let $k \geq 3$ be the number of colors used in G_y by $\tilde{c}$. It is easy to see that no vertex of G_x is happy as each vertex of G_x is adjacent to at least two vertices of different colors. Moreover, all the vertices in G_x need to be colored with a single color for otherwise the number of happy vertices becomes zero. We try over all possible $O(k)$ many ways of coloring G_x. In the end color all uncolored vertices in G_y with the color used in G_x.

Number of colors in G_x is one. Since $k \geq 3$, the number of colors (distinct from the color used in G_x) used in G_y by $\tilde{c}$ is at least two. An optimal coloring is obtained by coloring all the uncolored vertices of G with the color used in G_x.

Number of colors in G_x is at least two. We assume that the number of colors in G_y is at least two, for otherwise we can use one of the previous cases by interchanging G_x and G_y. In this case no vertex in G is happy, because every vertex of G is adjacent to at least two precolored vertices having different colors. □

9 Concluding Remarks

We studied the parameterized complexity of the main variants of the happy coloring problems. The results extend considerably our understanding of the complexity of the problems (see Subsection 1.2 for an overview).

Recent developments build on our results and clarify this landscape even further. In particular, the following results are established by Bliznets and Sagunov [6], directly extending our understanding of the problem in the context of structural parameters. The authors show that MHV is W[1]-hard with respect to parameters pathwidth, treewidth or clique-width, distance to cographs, and feedback vertex set number. Also, MHE is W[1]-hard with respect to the cluster vertex deletion number (a generalization of the clique deletion number, and hence a smaller parameter in general), while MHV is FPT with respect to this parameter. When parameterized by distance to clique, MHV admits a polynomial kernel.

In [5], several algorithmic lower bounds are established for these problems. Among the most relevant to our work are the following. Under reasonable complexity-theoretic assumptions, it is shown that there are no polynomial kernels for MHV parameterized by vertex cover, and no polynomial kernels for MHE under the following parameterizations: number of uncolored vertices, number of happy edges, and distance to almost any reasonable graph class (see the respective work for a precise definition). On the other hand, to the best of our knowledge, the question of whether MHV admits a linear kernel when parameterized by $k + \ell$, i.e., the number of colors and solution size remains open.

Acknowledgements We thank the anonymous reviewers for their helpful comments on this work, and in particular for pointing out a mistake in an earlier proof of Theorem 32.

References

[1] A. Agrawal. On the parameterized complexity of happy vertex coloring. In *Proceedings of the 28th International Workshop on Combinatorial Algorithms (IWOCA)*, volume 10765 of *Lecture Notes in Computer Science*, pages 103–115, 2017.

[2] N. R. Aravind, S. Kalyanasundaram, and A. S. Kare. Linear time algorithms for happy vertex coloring problems for trees. In *Proceedings of the 27th International Workshop on Combinatorial Algorithms (IWOCA)*, volume 9843 of *Lecture Notes in Computer Science*, pages 281–292. Springer, 2016.

[3] N. R. Aravind, S. Kalyanasundaram, A. S. Kare, and J. Lauri. Algorithms and hardness results for happy coloring problems. *CoRR*, abs/1705.08282, 2017.

[4] A. Björklund, T. Husfeldt, and M. Koivisto. Set Partitioning via Inclusion-Exclusion. *SIAM Journal on Computing*, 39(2):546–563, 2009.

[5] I. Bliznets and D. Sagunov. Lower bounds for the happy coloring problems. In *Proceedings of the 25th International Conference on Computing and Combinatorics (COCOON)*, volume 11653 of *Lecture Notes in Computer Science*, pages 490–502, 2019.

[6] I. Bliznets and D. Sagunov. On happy colorings, cuts, and structural parameterizations. In *Proceedings of the 45th International Workshop on Graph-Theoretic Concepts in Computer Science (WG)*, volume 11789 of *Lecture Notes in Computer Science*, pages 148–161, 2019.

[7] H. L. Bodlaender, P. S. Bonsma, and D. Lokshtanov. The fine details of fast dynamic programming over tree decompositions. In *Proceedings of the 8th International Symposium on Parameterized and Exact Computation (IPEC)*, volume 8246 of *Lecture Notes in Computer Science*, pages 41–53, 2013.

[8] H. L. Bodlaender, P. G. Drange, M. S. Dregi, F. V. Fomin, D. Lokshtanov, and M. Pilipczuk. A c$^{\mathrm{k}}$ n 5-approximation algorithm for treewidth. *SIAM Journal on Computing*, 45(2):317–378, 2016.

[9] H. L. Bodlaender, M. R. Fellows, M. A. Langston, M. A. Ragan, F. A. Rosamond, and M. Weyer. Quadratic kernelization for convex recoloring of trees. *Algorithmica*, 61(2):362–388, Oct 2011.

[10] Y. Cao, J. Chen, and J.-H. Fan. An $O^*(1.84^k)$ parameterized algorithm for the multiterminal cut problem. *Information Processing Letters*, 114(4):167–173, 2014.

[11] S. Chopra and M. R. Rao. On the multiway cut polyhedron. *Networks*, 21(1):51–89, 1991.

[12] M. Cygan, F. V. Fomin, Ł. Kowalik, D. Lokshtanov, D. Marx, M. Pilipczuk, M. Pilipczuk, and S. Saurabh. *Parameterized Algorithms.* Springer, 2015.

[13] E. Dahlhaus, D. S. Johnson, C. H. Papadimitriou, P. D. Seymour, and M. Yannakakis. The complexity of multiway cuts (extended abstract). In *Proceedings of the 24th Annual ACM Symposium on Theory of Computing (STOC)*, pages 241–251. ACM, 1992.

[14] R. Diestel. *Graph Theory, 4th Edition*, volume 173 of *Graduate texts in mathematics.* Springer, 2012.

[15] R. Duan, S. Pettie, and H.-H. Su. Scaling algorithms for weighted matching in general graphs. *ACM Trans. Algorithms*, 14(1):8:1–8:35, Jan. 2018.

[16] D. Easley and J. Kleinberg. *Networks, Crowds, and Markets: Reasoning About a Highly Connected World.* Cambridge University Press, 2010.

[17] M. R. Fellows, D. Hermelin, F. A. Rosamond, and S. Vialette. On the parameterized complexity of multiple-interval graph problems. *Theoretical Computer Science*, 410(1):53–61, 2009.

[18] F. V. Fomin and D. Kratsch. *Exact exponential algorithms.* Springer Science & Business Media, 2010.

[19] F. V. Fomin, D. Lokshtanov, S. Saurabh, and M. Zehavi. *Kernelization: theory of parameterized preprocessing*. Cambridge University Press, 2019.

[20] T. Gallai. Transitiv orientierbare graphen. *Acta Mathematica Hungarica*, 18(1):25–66, 1967.

[21] R. Ganian. Using neighborhood diversity to solve hard problems. *CoRR*, abs/1201.3091, 2012.

[22] L. O. James, R. G. Stanton, and D. D. Cowan. Graph decomposition for undirected graphs. In *Proceedings of the 3rd Southeastern Conference on Combinatorics, Graph Theory, and Computing*, pages 281–290, 1972.

[23] T. Kloks. *Treewidth, Computations and Approximations*, volume 842 of *Lecture Notes in Computer Science*. Springer, 1994.

[24] M. Lampis. Algorithmic meta-theorems for restrictions of treewidth. *Algorithmica*, 64(1):19–37, 2012.

[25] M. Langberg, Y. Rabani, and C. Swamy. Approximation algorithms for graph homomorphism problems. In *Proceedings of the 9th International Conference on Approximation Algorithms for Combinatorial Optimization Problems, and 10th International Conference on Randomization and Computation (APPROX-RANDOM)*, volume 4110 of *Lecture Notes in Computer Science*, pages 176–187. Springer, 2006.

[26] P. F. Lazarsfeld and R. K. Merton. Friendship as a social process: A substantive and methodological analysis. *Freedom and Control in Modern Society*, 18(1):18–66, 1954.

[27] M. McPherson, L. Smith-Lovin, and J. M. Cook. Birds of a Feather: Homophily in Social Networks. *Annual Review of Sociology*, 27(1):415–444, 2001.

[28] N. Misra and I. V. Reddy. The parameterized complexity of happy colorings. In *Proceedings of the 28th International Workshop on Combinatorial Algorithms (IWOCA)*, pages 142–153, 2017.

[29] P. Zhang, T. Jiang, and A. Li. Improved Approximation Algorithms for the Maximum Happy Vertices and Edges Problems. In *Proceedings of the 21st Annual International Conference on Computing and Combinatorics (COCOON)*, volume 9198 of *Lecture Notes in Computer Science*, pages 159–170. Springer, 2015.

[30] P. Zhang and A. Li. Algorithmic aspects of homophyly of networks. *Theoretical Computer Science*, 593:117–131, 2015.