On the Tractability of (k, i)-Coloring

Saurabh Joshi, Subrahmanyam Kalyanasundaram, Anjeneya Swami Kare, and Sriram Bhyravarapu*

Department of Computer Science and Engineering, IIT Hyderabad
{sbjoshi,subruk,cs14resch01002,cs16resch11001}@iith.ac.in

Abstract. In an undirected graph, a proper (k, i)-coloring is an assignment of a set of k colors to each vertex such that any two adjacent vertices have at most i common colors. The (k, i)-coloring problem is to compute the minimum number of colors required for a proper (k, i)-coloring. This is a generalization of the classic graph coloring problem. Majumdar et. al. [CALDAM 2017] studied this problem and showed that the decision version of the (k, i)-coloring problem is fixed parameter tractable (FPT) with tree-width as the parameter. They asked if there exists an FPT algorithm with the size of the feedback vertex set (FVS) as the parameter without using tree-width machinery. We answer this in positive by giving a parameterized algorithm with the size of the FVS as the parameter. We also give a faster and simpler exact algorithm for $(k, k-1)$-coloring, and make progress on the NP-completeness of specific cases of (k, i)-coloring.

1 Introduction

In an undirected graph $G = (V, E)$, $|V| = n$, a *proper vertex coloring* is to color the vertices of the graph such that adjacent vertices get different colors. The classic graph coloring problem asks to compute the minimum number of colors required to properly color the graph. The minimum number of colors required is called the *chromatic number* of the graph, denoted by $\chi(G)$. This is a well known NP-hard problem and has been studied in multiple directions.

Many variants and generalizations of the graph coloring problem have been studied in the past. In this paper we address a generalization of the graph coloring problem called (k, i)-coloring problem. For a proper (k, i)-coloring, we need to assign a set of k colors to each vertex such that the adjacent vertices share at most i colors. The (k, i)-coloring problem asks to compute the minimum number of colors required to properly (k, i)-color the graph. The minimum number of colors required is called the (k, i)-*chromatic number*, denoted by $\chi_k^i(G)$. Note that $(1, 0)$-coloring is the same as the classic graph coloring problem.

* The authors are in alphabetical order, with the fourth author's first name being considered.

(k, i)-Coloring Problem
Instance: An undirected graph $G = (V, E)$.
Output: The (k, i)-chromatic number of G, $\chi_k^i(G)$.

We also define below the (q, k, i)-coloring problem, the decision version of the (k, i)-coloring problem.

(q, k, i)-Coloring Problem
Instance: An undirected graph $G = (V, E)$.
Question: Does G have a proper (k, i)-coloring using at most q colors?

The (k, i)-coloring problem was first studied by Méndez-Díaz and Zabala in [1]. For arbitrary k and i, the (k, i)-coloring problem is NP-hard because $(1, 0)$-coloring is NP-hard. Apart from studying the basic properties, they also gave an integer linear programming formulation of the problem. Stahl [2] and independently Bollobás and Thomason [3] introduced the $(k, 0)$-coloring problem under the names of k-tuple coloring and k-set coloring respectively. The k-tuple coloring problem has been studied in detail [4,5], and Irving [6] showed that this problem is NP-hard as well. Some of the applications for the $(k, 0)$-coloring problem include construction of pseudorandom number generators, randomness extractors, secure password management schemes, aircraft scheduling, biprocessor tasks and frequency assignment to radio stations [7,8]. Brigham and Dutton [9] studied another variant of the problem, where k colors have to be assigned to each vertex such that the adjacent vertices share exactly i colors.

Bonomo et. al. [10] studied the connection between the (k, i)-coloring problem on cliques and the theory of error correcting codes. In coding theory, a (j, d, k)-constant weight code represents a set of codewords of length j with exactly k ones in each codeword, with Hamming distance at least d. Bonomo et. al. observed a direct connection between $A(j, d, k)$, the largest possible size of a (j, d, k)-constant weight code, and the (k, i)-colorability of cliques and used the existing results from coding theory (such as the Johnson bound [11]) to infer results on the (k, i)-colorability of cliques. Finding bounds on $A(j, d, k)$ is a well-studied problem in coding theory, and lots of questions on $A(j, d, k)$ are still open. This indicates the difficulty of the (k, i)-coloring problem even on graphs as simple as cliques.

Since the (k, i)-coloring problem is NP-hard in general, it is natural to study the tractability for special classes of graphs. Polynomial time algorithms are only known for a few of such classes namely bipartite graphs, cycles, cacti and graphs with bounded vertex cover or tree-width [10,12]. From the NP-hardness perspective, it is interesting to ask if the (k, i)-coloring problem is NP-hard for specific values of i. Except for the cases $i = k$, where the problem is trivial, and $i = 0$, where the problem is NP-hard [6], the NP-hardness remains open for all other values of i.

Recently, Majumdar et. al. [12] studied the (k, i)-coloring problem and gave exact and parameterized algorithms for the problem. They showed that the prob-

lem is fixed parameter tractable (FPT) when parameterized by tree-width. As the tree-width is at most $(|S|+1)$, where S is a feedback vertex set (FVS) of the graph, their algorithm also implies that (k,i)-coloring is FPT when parameterized by the size of FVS. As an open question, they asked to devise an FPT algorithm parameterized by the size of FVS, without going through tree-width. In this paper we answer this question.

Our results are:

- An $O(\binom{q}{k}^{|S|+2} n^{O(1)})$ time algorithm for the (q,k,i)-coloring problem that does not use tree-width machinery. Here S is an FVS of the graph.
- We make progress on the NP-hardness of the (k,i)-coloring problem. We show that $(k,1)$-coloring and $(k,k-1)$ coloring are NP-complete, in addition to other NP-completeness results. This partially answers questions posed in [1] and [12].
- We give a $2^n n^{O(1)}$ time exact algorithm for the $(k,k-1)$-coloring problem. This is a direct improvement to the algorithm given in [12] for the same problem.

2 Preliminaries

A *parameterized problem* is a language $B \subseteq \Sigma^* \times \mathbb{N}$ where Σ is a fixed, finite alphabet. For example $(x,\ell) \in \Sigma^* \times \mathbb{N}$, here ℓ is called the parameter. A parameterized problem $B \subseteq \Sigma^* \times \mathbb{N}$ is called *fixed-parameter tractable* (FPT) if there is an algorithm $\mathcal{A}$, a computable function $f : \mathbb{N} \to \mathbb{N}$, and a constant c such that, given $(x,\ell) \in \Sigma^* \times \mathbb{N}$, the algorithm $\mathcal{A}$ correctly decides whether $(x,\ell) \in B$ in time bounded by $f(\ell)|x|^c$.

We assume that the graph is simple and undirected. We use n to denote $|V|$, the number of vertices of the graph. We say that the vertices u and v are *adjacent* (*neighbors*) if $\{u,v\} \in E$. For $v \in V$, we let $N(v)$ denote the set of neighbors of v. For $S \subseteq V$, the sub graph induced by S is denoted by $G[S]$. We use $O^*(f(n))$ to denote $O(f(n)n^{O(1)})$. We use the set of natural numbers for coloring the graph. We use the standard notations $[q] = \{1,2,\ldots,q\}$ and $\binom{[q]}{k}$ to denote the set of all k-sized subsets of $[q]$. In the rest of the paper, we use the term *coloring* of a set $X \subseteq V$ to denote a mapping $h : X \to \binom{[q]}{k}$. We say that h is a *proper* (q,k,i)-coloring (or proper (k,i)-coloring) of X if any pair of adjacent vertices in X have no more than i colors in common.

3 (q,k,i)-Coloring Parameterized by Size of FVS

In this section, we assume that q,k,i are fixed values and focus on the decision problem of (q,k,i)-coloring. Let $G=(V,E)$ be an undirected graph. Let $G[V']$ denote a subgraph of G induced by $V' \subseteq V$. A *Feedback Vertex Set (FVS)* is a set of vertices $S \subseteq V$, removal of which from the graph G makes the remaining graph $(G[V \setminus S])$ acyclic. Many NP-hard problems have been shown to be tractable for graphs with bounded FVS [13].

In [12], Majumdar et.al., gave an algorithm for the (q,k,i)-coloring problem in $O(\binom{q}{k}^{tw+1}n^{O(1)})$ time[1], where tw denotes the tree-width of the graph. Let S be a smallest FVS of G. It is known that $tw \leq |S|+1$, see for instance [14]. In this section, we present an algorithm for (q,k,i)-coloring that runs in $O(\binom{q}{k}^{|S|+2}n^{O(1)})$ time, where $|S|$ is the size of the FVS of the graph. Our algorithm does not use the tree-width machinery. Note that, FVS has a 2-approximation algorithm [15], but there is no known polynomial time algorithm that approximates tree-width within a constant factor [16]. Computing the size of the smallest FVS is also known to be FPT parameterized by $|S|$, the size of the smallest FVS. There has been a series of results improving the running time, the fastest known algorithm [17] runs in $O(3.619^{|S|}n^{O(1)})$ time.

A brief description of our algorithm follows. Let S be an FVS of G. We start with a coloring of the vertices of S. Recall that $G[V\backslash S]$ is a forest. Each of the connected components of $G[V\backslash S]$ is a tree. For each of these components, we traverse the tree bottom-up and use a dynamic programming technique to compute the list of k-colorings that each vertex $w \in V\backslash S$ can take. For each $C \in \binom{[q]}{k}$, we include C in w's list if there is a coloring for the subtree rooted at w, consistent with the coloring of S, such that w receives color set C. We repeat this for all proper colorings of S.

Let $\Psi = \binom{[q]}{k}$ denote the family of all k-sized subsets of $[q]$. For any pair of sets $C, C' \in \Psi$, we say that (C,C') is *legal* if $|C \cap C'| \leq i$, and *illegal* if $|C \cap C'| > i$. Given two sets $C, C' \in \Psi$, it is easy to check if (C,C') is a legal pair. Formally, we have:

Proposition 1. *Given $C, C' \in \Psi$, it takes $O(k \log k)$ time to check if (C,C') is a legal pair.*

Definition 2. *Consider a partial coloring $h : S \to \Psi$ where only the vertices of the FVS S are colored. For a vertex $w \in V\backslash S$ and a set $C \in \Psi$, we say that (w,C) is* h-compatible *if for all $x \in S \cap N(w)$, the pair $(C, h(x))$ is legal.*

The set $\{C \in \Psi \mid (w,C)$ is h-compatible$\}$ is defined to be the set of h-compatible colorings of w.

Proposition 3. *Let $h : S \to \Psi$ be a coloring of the vertices in S. Let $w \in V\backslash S$ and $d_S(w) = |N(w) \cap S|$. Then the set of h-compatible colorings of w can be computed in time $O\left(\binom{q}{k} d_S(w) k \log k\right)$.*

Proof. For each $C \in \Psi$, we check if (w,C) is h-compatible. For this, we need to check for all neighbors x of w in S, whether $(C, h(x))$ is legal. The total running time is $\binom{q}{k} \cdot d_S(w) \cdot O(k \log k)$. □

Definition 4. *Given a graph $G = (V,E)$ and a coloring $h : X \to \Psi$ for some $X \subseteq V$, we say that the coloring $h' : V \to \Psi$ is an* extension of h, *or* extends h *if for all $v \in X$, we have $h(v) = h'(v)$.*

[1] Even though [12] claims a running time of $O(\binom{q}{k}^{tw}n^{O(1)})$ for their algorithm, there is an additional factor of $\binom{q}{k}$ that is omitted, presumably because $\binom{q}{k}$ is treated as a constant.

Lemma 5. *Given a proper (q, k, i)-coloring h of the vertices in a feedback vertex set S of the graph $G = (V, E)$, we can determine if h can be extended to a proper (q, k, i)-coloring of V in $O(\binom{q}{k}^2 n^{O(1)})$ time.*

Proof. The graph $G[V \setminus S]$ is a forest because S is a feedback vertex set. Therefore each connected component of $G[V \setminus S]$ is a tree. Below, we describe an algorithm that we can apply to each of these trees to yield a proper (q, k, i)-coloring extending h for the trees. Combining the colorings, we get a proper (q, k, i)-coloring of V, that is an extension of h.

Let T denote one of the trees in the forest. We will designate any one of the vertices (say r) of T as root. Let T_w denote the subtree rooted at a node $w \in T$.

Our plan is to maintain a table at each vertex w, indexed with the elements of Ψ. The entry at each color set C is denoted by $M_w(C)$. The entry $M_w(C)$ indicates whether there is a proper (q, k, i)-coloring of T_w, with w assigned the set C, consistent with the coloring h of S.

We will process T in a post order fashion as follows:

1. **When w is a leaf in T:** In this case, we set $M_w(C) = 1$ if (w, C) is h-compatible. Otherwise, we set $M_w(C) = 0$.
   For any leaf w, the values $M_w(C)$ corresponding to all $C \in \Psi$ can be computed in time $O(\binom{q}{k} d_S(w) k \log k)$ by Proposition 3. Here $d_S(w)$ denotes the number of neighbors of w in S.
2. **When w is an internal node in T:** Let $u_1, u_2, \ldots$ be the children of w in T. Recall that we process T in post order fashion. Before we process w, the M_{u_j} values for all the children of w would already have been computed. The value $M_w(C)$ is computed as follows:
   - If (w, C) is not h-compatible, we set $M_w(C) = 0$.
   - If (w, C) is h-compatible, we do the following:
     - If for each child u_j of w, there exists at least one coloring $C' \in \Psi$ such that $M_{u_j}(C') = 1$ and (C, C') is a legal pair, then set $M_w(C) = 1$.
     - Otherwise set $M_w(C) = 0$.

   For each w and C, the h-compatibility check takes $O(d_S(w) k \log k)$ time. If (w, C) is h-compatible, we need to check all the children u_j, and the table entries $M_{u_j}(C')$ for all $C' \in \Psi$. Together with the check for (C, C') being a legal pair, the computation takes $d_T(w) \cdot \binom{q}{k} \cdot O(k \log k)$ time, where $d_T(w)$ is the number of children of w in the tree T.
   Adding all up, the computation of the table entries for w takes time

$$O\left(\binom{q}{k} \cdot k \log k \cdot \left[d_S(w) + d_T(w)\binom{q}{k}\right]\right). \tag{1}$$

If for some $C \in \Psi$, $M_r(C) = 1$, then we know that there exists a proper (q, k, i)-coloring of T that is consistent with the coloring h of S.

The time complexity is obtained by adding the expression in (1) over all the vertices $w \in V \setminus S$. By using the bounds $d_S(w) \leq n$ and $\sum_{\text{Trees } T} \sum_{w \in V(T)} d_T(w) \leq \sum_{\text{Trees } T} |V(T)| \leq n$, we get that the time complexity is upper bounded by

$$O\left(\binom{q}{k} \cdot k \log k \cdot \left[n^2 + n\binom{q}{k}\right]\right),$$

which is at most $O\left(\binom{q}{k}^2 \cdot n^2\right)$, by noting that k is a constant. □

The correctness of the procedure explained in the above lemma can be proved using an induction on the vertices of T according to its post order traversal. The inductive claim says that $M_w(C) = 1$ if and only if there is a proper (q, k, i)-coloring of T_w, with w assigned the set C, consistent with the given coloring of S.

Lemma 6. *Given a proper (q, k, i)-coloring h of the vertices in a feedback vertex set S of the graph $G = (V, E)$, we can determine if h can be extended to a proper (q, k, i)-coloring of V with space complexity $O(\binom{q}{k}n)$.*

Proof. Recall the algorithm explained in Lemma 5. At each vertex w in $G[V \setminus S]$, we need $O(\binom{q}{k})$ space to store values $M_w(C)$ for all $C \in \Psi$. □

Theorem 7. *The (q, k, i)-coloring problem can be solved in time $O((\binom{q}{k})^{|S|+2}n^{O(1)})$ and $O(\binom{q}{k}n)$ space, where S is a feedback vertex set of G.*

Proof. For each coloring assignment h of S, we first determine if h is a proper (q, k, i)-coloring. This can be done in $O(|S|^2 k \log k)$ time. Then we determine whether there exists a proper (q, k, i)-coloring that extends h in $O((\binom{q}{k})^2.n^{O(1)})$ time by Lemma 5. Since there are at most $(\binom{q}{k})^{|S|}$ many colorings of S, we can determine whether there exists a proper (q, k, i)-coloring of G in $O((\binom{q}{k})^{|S|+2}n^{O(1)})$ time.

We need $O(|S|k \log q)$ space to store the coloring h of S. And by Lemma 6, we need $O(\binom{q}{k}n)$ space to determine if h can be extended to a proper coloring of G. The latter is the dominating term and determines the total space requirement of the algorithm. □

On generating a proper (q, k, i)-coloring. We observe that we can modify Theorem 7 to obtain an algorithm that generates a proper (q, k, i)-coloring of G, if one exists. After executing the steps of the algorithm corresponding to Theorem 7, we traverse the tree in top-down fashion from the root, and find colorings for each vertex $w \in T$, consistent with its parent, subtree T_w and coloring of S. The latter two are already encoded in $M_w(C)$ value. The asymptotic time and space complexity are the same as that in Theorem 7.

We would like to observe a difference in the space usage of our FPT algorithm to the FPT algorithm for (q, k, i)-coloring parameterized by tree-width in [12]. We note that the algorithm in [12] can also be modified similarly to obtain an algorithm that generates a proper coloring. However, such an algorithm would require to store all feasible colorings at each bag of the tree-decomposition, resulting in a $O((\binom{q}{k})^{tw+1})$ space usage at each bag. Since there are $O(n)$ bags, total space required by the algorithm is $O((\binom{q}{k})^{tw+1}n)$, which is significantly larger than the $O(\binom{q}{k}n)$ space required by our algorithm.

Decision vs. search problem. We note that we could run the algorithm for (q, k, i)-coloring for $q = 1, 2, 3, \ldots$ till we reach $\chi_k^i(G)$, the smallest q for which the graph has a proper (q, k, i)-coloring. The running time of this procedure would

be at most $O\left(\chi_k^i \binom{\chi_k^i}{k}^{|S|+2} n^{O(1)}\right)$. Thus an FPT algorithm parameterized by the size of the FVS for the (q,k,i)-coloring problem implies an FPT algorithm parameterized by combined parameters — the size of the FVS and the (k,i)-chromatic number.

3.1 Counting all proper (q,k,i)-colorings

Here we show that we can modify the algorithm described in Lemma 5 to count the number of proper (q,k,i)-colorings of G. Let a proper (q,k,i)-coloring h of FVS S be given. Instead of maintaining $M_w(C)$ for a vertex w in a rooted tree T, we maintain another value $M_w^{\#}(C)$.

$$M_w^{\#}(C) = \begin{cases} 0 & \text{if } (w,C) \text{ is not } h\text{-compatible.} \\ 1 & \begin{cases} \text{if } w \text{ is a leaf,} \\ \text{and } (w,C) \text{ is } h\text{-compatible.} \end{cases} \\ \displaystyle\prod_{\forall u_j \in \text{child}(w)} \sum_{\text{legal}(C,C')} M_{u_j}^{\#}(C') & \begin{cases} \text{if } w \text{ is a non-leaf vertex,} \\ \text{and } (w,C) \text{ is } h\text{-compatible.} \end{cases} \end{cases}$$

At each vertex w, $M_w^{\#}(C)$ maintains a count of the proper (q,k,i)-colorings of T_w, consistent with the coloring h of S, where w gets assigned the set C. The correctness can be verified by a straightforward induction on the tree vertices in post order traversal. If r is the root of T, $M_r^{\#}(C)$ gives the count of proper (q,k,i)-colorings of T, where r is colored C, consistent with the coloring h of S.

The total number of proper (q,k,i)-colorings of G is therefore computed by taking into account (i) all proper (q,k,i)-colorings h of S, (ii) all the trees T_j in $G[V \backslash S]$, and (iii) all color sets $C \in \Psi$ at the root of T_j. The full expression is as follows:

$$\text{No. of proper } (q,k,i)\text{-colorings } = \sum_{\substack{\text{proper } (q,k,i)\text{-} \\ \text{colorings of } S}} \left(\prod_{T_j \text{ in } G[V \backslash S]} \left(\sum_{C \in \Psi} M_{\text{root}(T_j)}^{\#}(C) \right) \right).$$

The above expression implies the following theorem. The asymptotic time complexity remains the same as Theorem 7, whereas the space complexity incurs a blowup of $nk \log q$, because of the maximum value $M_w^{\#}(C)$ can take.

Theorem 8. *There is an algorithm that computes the number of proper (q,k,i)-colorings of G, in $O(\binom{q}{k}^{|S|+2} n^{O(1)})$ time and $O(\binom{q}{k} n^2 \log q)$ space, where S is a feedback vertex set of G.*

4 Faster Exact Algorithm for $(k, k-1)$-coloring

In [12], Majumdar et. al. gave an $O^*(4^n)$ time exact algorithm for the $(k,k-1)$-coloring problem. Their algorithm was based on running an exact algorithm for a

set cover instance where the universe is the set of all the vertices V and the family of sets $\mathcal{F}$ is the set of all independent sets of vertices of G. To show correctness and running time, they used a claim (unnumbered) that relates $\chi_k^{k-1}(G)$ to the size of solution of the set cover instance, an $O(2^n \cdot n \cdot |\mathcal{F}|)$ time exact algorithm for the set cover problem [18] and an upper bound of 2^n on the size of the family of sets $\mathcal{F}$. Hence, the time complexity of their algorithm is $O(2^n \cdot n \cdot 2^n) = O(4^n \cdot n)$.

We first note that their algorithm also works when $\mathcal{F}$ is replaced by $\mathcal{F}'$, the set of all maximal independent sets of G. This is because any independent set $A \in \mathcal{F}$ is contained in a maximal independent set $A' \in \mathcal{F}'$. In any set covering of V using elements of $\mathcal{F}$, each set A can be replaced by an $A' \in \mathcal{F}'$, thus obtaining a set cover of V using elements of only $\mathcal{F}'$. By using the $3^{n/3}$ upper bound of Moon and Moser [19] on the number of maximal independent sets, the time complexity improves to $O(2^n \cdot n \cdot 3^{n/3}) = O(2.88^n \cdot n)$.

We now present a simpler and faster $O^*(2^n)$ algorithm to determine $\chi_k^{k-1}(G)$.

Lemma 9. *For any graph G, $\chi_k^{k-1}(G) = q$ where q is the smallest integer such that $\binom{q}{k} \geq \chi_1^0(G)$. Thus there is a polynomial time reduction from the $(k, k-1)$-coloring problem to the $(1, 0)$-coloring problem.*

Proof. The $(k, k-1)$-coloring problem asks to assign sets of k colors to each vertex, with the requirement that neighboring vertices must have distinct sets assigned to them. We may view each of the k-sized subsets as a color, and the $(1, 0)$-chromatic number $\chi_1^0(G)$ is the number of distinct k-sized subsets required.

Thus $\chi_k^{k-1}(G)$ is the smallest q that will provide $\chi_1^0(G)$ number of k-sized subsets. The polynomial time reduction is immediate. □

Combining the above lemma with the $O^*(2^n)$ time algorithm of Koivisto [20] to compute $\chi_1^0(G)$, we get the following theorem.

Theorem 10. *There is an algorithm with $O^*(2^n)$ time complexity that computes the $(k, k-1)$ chromatic number of a given graph.*

Further, we can infer from Lemma 9 that for those graphs G where we can compute χ_1^0(G) in polynomial time, $\chi_k^{k-1}(G)$ can also be found in polynomial time. For instance, $\chi_k^{k-1}(K_n)$ can be computed in polynomial time as $\chi_1^0(K_n) = n$.

5 NP-completeness results

Since the (k, i)-coloring problem is a generalization of the $(1, 0)$-coloring problem, it follows that (k, i)-coloring is NP-hard. Méndez-Díaz and Zabala [1] conjectured that the (k, i)-coloring problem remains NP-hard even for specific values of i. In this section, we show NP-completeness results for some specific cases of (k, i)-coloring. We will only be proving the NP-hardness aspect of NP-completeness. Given a coloring, we can easily verify that it is a proper (k, i)-coloring in polynomial time.

Trivially, we have $\chi_k^k(G) = k$ for all graphs G. For the $(k, 0)$-coloring problem, we have the following result by Irving.

Theorem 11 ($(k,0)$-coloring is NP-complete [6]). *The $(2k+1,k,0)$-coloring problem is NP-complete for all $k \geq 1$.*

The NP-completeness of the $(k, k-1)$-coloring problem is claimed by [1]. However, we are unable to follow and verify the proof. We provide an alternate NP-hardness proof as a consequence of the correspondence in Lemma 9.

Theorem 12. *The $(k, k-1)$-coloring problem is NP-complete for all $k \geq 1$.*

Proof. We use reductions from the $(1,0)$-coloring problem, for each value of $k \geq 2$. We show that the $(q, k, k-1)$-coloring problem is NP-complete for all values of $q > k \geq 2$. From the correspondence in Lemma 9, it follows that for any given $k \geq 1$, a graph G is $(q, k, k-1)$-colorable if and only if G is $(\binom{q}{k}, 1, 0)$-colorable. Since the $(r, 1, 0)$-coloring problems are NP-complete for all $r \geq 3$, it follows that $(\binom{q}{k}, 1, 0)$-coloring problems are NP-complete for all $q > k \geq 2$, and hence we get that the $(q, k, k-1)$-coloring problems are NP-complete for all $q > k \geq 2$. □

The following lemmas will help us in proving further NP-completeness results.

Lemma 13 (Complement trick). *For integers $k, i \geq 1$, any graph G is $(2k+i, k+i, i)$-colorable if and only if it is $(2k+i, k, 0)$-colorable.*

Proof. Let $f : V \to \binom{[2k+i]}{k}$ be a $(2k+i, k, 0)$-coloring of G. Consider the coloring f' where each vertex v is assigned the complement set $[2k+i] \setminus f(v)$. Notice that, every vertex is assigned $(k+i)$ colors, and any pair of adjacent vertices will share exactly i colors in the coloring f'. Thus we have a $(2k+i, k+i, i)$-coloring of G.

Similarly, if we start from a $(2k+i, k+i, i)$-coloring of G, we can get to a $(2k+i, k, 0)$-coloring by taking the complement coloring. □

Theorem 11 and the above lemma together imply the NP-completeness of $(k, 1)$-coloring for all $k \geq 2$.

Theorem 14 ($(k,1)$-coloring is NP-complete). *The $(2k+1, k+1, 1)$-coloring problem is NP-complete for all $k \geq 1$.*

Now we introduce another simple gadget, the universal vertex. Given a graph G, we can construct a graph G' by adding a new vertex v that is adjacent to all the vertices of G. It is straightforward to see that G has a $(q, k, 0)$-coloring if and only if the new graph G' has a $(q+k, k, 0)$-coloring. Thus we have the following:

Lemma 15 (Universal vertex). *There is a polynomial time reduction from the $(q, k, 0)$-coloring problem to the $(q+k, k, 0)$-coloring problem. Therefore, the $(q+k, k, 0)$-coloring problem is NP-complete when the $(q, k, 0)$-coloring problem is NP-complete.*

Combining Lemma 13 and Lemma 15 yields a collection of NP-completeness results.

Theorem 16. *For any integers $p \geq 2$ and $\ell \geq 1$, the $(p\ell+1, (p-1)\ell+1, (p-2)\ell+1)$-coloring problem is NP-complete.*

Proof. From Theorem 11, we have that the $(2\ell+1, \ell, 0)$-coloring problem is NP-complete. By applying the universal vertex gadget of Lemma 15 a total of $(p-2)$ times, we get that the $(p\ell+1, \ell, 0)$-coloring problem is NP-complete for all $p \geq 2$. Now we can use the complement trick of Lemma 13 to infer the NP-completeness of the $(p\ell+1, (p-1)\ell+1, (p-2)\ell+1)$-coloring problem. □

The above theorem gives us a collection of NP-completeness results. If we set $\ell = 2$, we get that problems $(5,3,1)$-coloring, $(7,5,3)$-coloring, $(9,7,5)$-coloring etc. are NP-complete. For other values of ℓ, we get similar sequence of NP-completeness results. But we cannot infer the NP-completeness of $(k,3)$-coloring, or $(k,5)$-coloring from this because all values of k are not covered. To show that $(k,3)$-coloring is NP-hard, we need to exhibit for all relevant k, a value q such that $(q,k,3)$-coloring is NP-hard. As conjectured in [1], we believe that the (k,i)-coloring problem is NP-complete for all values of i. As of now, the NP-completeness of (k,i)-coloring is still open for $2 \leq i \leq k-2$.

Acknowledgment: The authors would like to thank the anonymous reviewer for helpful comments, and pointing out a flaw in the proof of Theorem 12 in an earlier version of the paper.

References

1. Méndez-Díaz, I., Zabala, P.: A generalization of the graph coloring problem. Investigation Operation **8** (1999) 167–184
2. Stahl, S.: n-tuple colorings and associated graphs. Journal of Combinatorial Theory, Series B **20**(2) (1976) 185 – 203
3. Bollobás, B., Thomason, A.: Set colourings of graphs. Discrete Mathematics **25**(1) (1979) 21 – 26
4. Klostermeyer, W., Zhang, C.Q.: n-tuple coloring of planar graphs with large odd girth. Graphs and Combinatorics **18**(1) (2002) 119–132
5. Šparl, P., Žerovnik, J.: A note on n-tuple colourings and circular colourings of planar graphs with large odd girth. International Journal of Computer Mathematics **84**(12) (2007) 1743–1746
6. Irving, R.W.: NP-completeness of a family of graph-colouring problems. Discrete Applied Mathematics **5**(1) (1983) 111 – 117
7. Marx, D.: Graph colouring problems and their applications in scheduling. Periodica Polytechnica Electrical Engineering **48**(1-2) (2004) 11–16
8. Beideman, C., Blocki, J.: Set families with low pairwise intersection. arXiv preprint arXiv:1404.4622 (2014)
9. Brigham, R.C., Dutton, R.D.: Generalized k-tuple colorings of cycles and other graphs. Journal of Combinatorial Theory, Series B **32**(1) (1982) 90–94
10. Bonomo, F., Durán, G., Koch, I., Valencia-Pabon, M.: On the (k,i)-coloring of cacti and complete graphs. Ars Combinatoria (2014)
11. Johnson, S.: A new upper bound for error-correcting codes. IRE Transactions on Information Theory **8**(3) (1962) 203–207

12. Majumdar, D., Neogi, R., Raman, V., Tale, P.: Exact and parameterized algorithms for (k, i)-coloring. In: Algorithms and Discrete Applied Mathematics: Third International Conference, CALDAM 2017, India. (2017) 281–293
13. Kratsch, S., Schweitzer, P.: Isomorphism for graphs of bounded feedback vertex set number. Algorithm Theory-SWAT 2010 (2010) 81–92
14. Jansen, B.M., Raman, V., Vatshelle, M.: Parameter ecology for feedback vertex set. Tsinghua Science and Technology **19**(4) (2014) 387–409
15. Bafna, V., Berman, P., Fujito, T.: A 2-approximation algorithm for the undirected feedback vertex set problem. SIAM Journal on Discrete Mathematics **12**(3) (1999) 289–297
16. Wu, Y.L., Austrin, P., Pitassi, T., Liu, D.: Inapproximability of treewidth, one-shot pebbling, and related layout problems. J. Artif. Int. Res. **49**(1) (January 2014) 569–600
17. Kociumaka, T., Pilipczuk, M.: Faster deterministic feedback vertex set. Information Processing Letters **114**(10) (2014) 556 – 560
18. Fomin, F.V., Kratsch, D.: Exact Exponential Algorithms. Texts in Theoretical Computer Science, An EATCS Series. Springer (2010)
19. Moon, J.W., Moser, L.: On cliques in graphs. Israel Journal of Mathematics **3**(1) (Mar 1965) 23–28
20. Koivisto, M.: An $O^*(2^n)$ algorithm for graph coloring and other partitioning problems via inclusion–exclusion. In: Proceedings of the 47th Annual IEEE Symposium on Foundations of Computer Science. FOCS '06, Washington, DC, USA, IEEE Computer Society (2006) 583–590