Exact and Parameterized Algorithms for (k, i)-coloring

Diptapriyo Majumdar[1], Rian Neogi[2], Venkatesh Raman[1], and Prafullkumar Tale[1]

[1] The Institute of Mathematical Sciences, Chennai, HBNI, India.
`{diptapriyom|vraman|pptale}@imsc.res.in`
[2] NIIT University, Neemrana, Rajasthan, India. `rianneogi@gmail.com`

Abstract. Graph coloring problem asks to assign a color to every vertex such that adjacent vertices get different color. There have been different ways to generalize classical graph coloring problem. Among them, we study (k, i)-coloring of a graph. In (k, i)-coloring, every vertex is assigned a set of k colors so that adjacent vertices share at most i colors between them. The (k, i)-chromatic number of a graph is the minimum number of total colors used to assign a proper (k, i)-coloring. It is clear that $(1, 0)$-coloring is equivalent to the classical graph coloring problem. We extend the study of exact and parameterized algorithms for classical graph coloring problem to (k, i)-coloring of graphs. Given a graph with n vertices and m edges, we design algorithms that take

- $\mathcal{O}(2^{kn} \cdot n^{\mathcal{O}(1)})$ time to determine the $(k, 0)$-chromatic number.
- $\mathcal{O}(4^{n} \cdot n^{\mathcal{O}(1)})$ time to determine the $(k, k\text{-}1)$-chromatic number.
- $\mathcal{O}(2^{kn} \cdot k^{im} \cdot n^{\mathcal{O}(1)})$ time to determine the (k, i)-chromatic number.

We prove that (k, i)-coloring is fixed parameter tractable when parameterized by the size of the vertex cover or the treewidth of the graph. We also provide some observations on (k, i)-colorings on perfect graphs.

1 Introduction

We investigate efficient parameterized and exact exponential algorithms for a variant of graph coloring called (k, i)-coloring. Given a positive integer k, and a non-negative integer $i \leq k$, the (k, i)-coloring of a graph is a an assignment of a set of k colors to every vertex such that every pair of adjacent vertices shares at most i colors [10]. Motivation to pursue this problem comes from coding theory. An (n, d, w) constant-weight binary code is a set of binary vectors of length n, such that each vector contains w ones and any two vectors differ in at most d positions. One of the most basic questions in coding theory is given n, w, d, what is the largest possible size of an (n, d, w) constant-weight binary code? This question has been studied for almost five decades and remains open for any value of n, w, d [1]. It has been proved that the largest possible size of an (n, d, w) constant-weight binary code is closely related to the (k, i)-coloring of a complete graph on n vertices [5]. If the total number of distinct colors used by a (k, i)-coloring is q, then we say that the graph G has a proper (k, i)-coloring using

q colors. Notice that setting $k = 1, i = 0$ gives the classical graph coloring problem. The minimum number of colors needed to assign (single) color to vertices so that every adjacent pair gets different colors is called *chromatic number* of a graph G and it is denoted by $\chi(G)$. We denote the minimum number of colors needed to assign a proper (k, i)-coloring of graph G as $\chi_k^i(G)$ and call it (k, i)-*chromatic number* of a graph. The precise definition of the problem is given below.

> (k, i)-Coloring
> **Input:** Graph G, integer q
> **Question:** Does there exist a (k, i)-coloring of G using at most q colors?

This problem was first introduced by Mendez-Diaz and Zabala [10]. In the same paper authors provided a heuristic approach and a linear programming model for this problem. There are other types of tuple coloring problems generalizing the classical graph coloring problem. For arbitrary k and for $i = 0$, it is called *k-tuple coloring*. This idea of tuple coloring was also independently introduced by Hilton et al. [2], Stahl [18], Bollobas and Thomason [4]. Tuple-Coloring was generalized in a different way by Brigham and Dutton [6] where every vertex is assigned k colors and adjacent vertices share *exactly* i colors.

Computing the chromatic number of a graph has been considered as one of the notoriously difficult problems. This generalization makes the problem even harder. There exists a polynomial time algorithm to compute the chromatic number of a perfect graph. In the case of (k, i)-chromatic number, polynomial time algorithms are known only in case of simple cycles, cactus [5] and bipartite graphs [10]. No polynomial time algorithm is known for finding (k, i)-chromatic number even on well structured graphs like complete graphs for all values of n, k, i. We prove some simple connections of this parameter to the (standard) chromatic number and initiate a study of exact and parameterized complexity of the problem under different parameterizations. A brute force algorithm for testing if $\chi(G) \leq q$ takes $q^n \cdot n^{\mathcal{O}(1)}$ time. A series of improvements led to the current best runtime of $\mathcal{O}(2^n \cdot n^{\mathcal{O}(1)})$ time [13] to compute $\chi(G)$. Similarly, a brute force exact algorithm to determine whether $\chi_k^i(G) \leq q$ will run through all possible q colorings which will assign an arbitrary set of k colors to a vertex. Then, a vertex can be assigned $\binom{q}{k}$ color-sets and there are n vertices in the graph. So, this brute force algorithm will take $\binom{q}{k}^n \cdot n^{\mathcal{O}(1)}$ time. For $(k, 0)$-coloring and $(k, 1)$-coloring, we provide an improved exact exponential algorithm (using efficient algorithm for the classical set cover problem) and then generalize it for any (k, i)-coloring. We provide the following exact algorithms in Section 4. Given a graph with n vertices and m edges, we design algorithms that take

- $\mathcal{O}(2^{kn} \cdot n^{\mathcal{O}(1)})$ time to determine the $(k, 0)$-chromatic number.
- $\mathcal{O}(4^n \cdot n^{\mathcal{O}(1)})$ time to determine the $(k, k\text{-}1)$-chromatic number.
- $\mathcal{O}(2^{kn} \cdot k^{im} \cdot n^{\mathcal{O}(1)})$ time to determine the (k, i)-chromatic number.

Concerning parameterized complexity results, we first observe that for standard parameterization (where the parameter is the number of colors), classical graph coloring is para-NP-hard (see Section 2 for definitions). So, it is clear that $(1,0)$-Coloring is also para-NP-hard when it is parameterized by the number of colors. But, it is not clear whether (k,i)-Coloring is para-NP-hard for any other values of k and i when the parameter is the number of colors. We follow the modern trend and resort to structural parameterizations where the problem is parameterized by some structure in the input. Specifically we consider the (k,i)-Coloring problem parameterized by the size of *vertex cover* of the graph. We also give efficient FPT algorithm for the problem on bounded treewidth graphs.

We organize this paper as follows. We state preliminaries and terminologies regarding (k,i)-coloring in Section 2. In Section 3, we provide some observations about (k,i)-coloring on perfect graphs and prove a conjecture stated in [10]. Sections 4 and 5 contain exact and fixed parameter tractable algorithms for (k,i)-coloring respectively.

2 Preliminaries

All graphs considered here are finite, undirected and simple. For a graph G, its vertex set is denoted by $V(G)$ and its edge set is denoted by $E(G)$. Vertex u is said to be adjacent to vertex v is $uv \in E(G)$. For a vertex $v \in V(G)$, its open neighborhood, $N_G(v)$, is the set of all vertices adjacent to it. The closed neighborhood $N_G[v] = \{v\} \cup N_G(v)$. We drop the subscript if it is clear from context. For a set $X \subseteq V(G)$, the subgraph of G induced by X is denoted by $G[X]$ and it is defined with vertex set X and edge set $\{uv \in E(G) : u, v \in X\}$. The subgraph obtained after deleting X is denoted by $G \setminus X$. Number of vertices in a maximum induced clique of a graph G is denoted by $\omega(G)$. K_n denotes a clique on n vertices. For a positive integer q, we denote the set $\{1, 2, \ldots, q\}$ by $[q]$. The family of all the k-sized subsets of $[q]$ is denoted by $[q]^k$. A function $f : V(G) \rightarrow [q]$ is called a coloring function of graph G. If for all edges uv, $f(u) \neq f(v)$, we say that f is proper coloring of graph G. The smallest integer q for which it is possible to properly color all vertices of graph G is called its *chromatic number* and it is denoted by $\chi(G)$. (k,i)-coloring is a generalization of proper coloring and defined as follows.

Definition 1. *A coloring function $f : V(G) \rightarrow [q]^k$ is called proper-(k,i)-coloring of a graph G if for any edge $uv \in E(G)$, $|f(u) \cap f(v)| \leq i$.*

In Coloring problem, input is a graph G, integer q and the question is whether G can be properly colored using at most q colors. Analogously, in (k,i)-Coloring problem, input is a graph G, integer q and the question is whether G can be (k,i)-colored using at most q colors. Notice that, we consider the case when k, i are fixed constants and not part of input. In Set Cover problem, input is a universe U and a family $\mathcal{F}$ of its subsets and the question is to find cardinality of minimum sized subset $\mathcal{F}'$ of $\mathcal{F}$ which *covers* U. $\mathcal{F}'$ covers U if every element of U is present in at least one set in $\mathcal{F}'$.

Parameterized Complexity: The goal of parameterized complexity is to find ways of solving NP-hard problems more efficiently than brute force by associating a small parameter to each instance. Parameterization of a problem is assigning a positive integer parameter p to each input instance. We say that a parameterized problem is Fixed-Parameter Tractable (FPT) if there is an algorithm that solves the problem in time $f(p) \cdot |I|^{\mathcal{O}(1)}$, where $|I|$ is the size of the input and f is an arbitrary computable function depending only on the parameter p. Such an algorithm is called an FPT algorithm, and the runtime of the algorithm is also sometimes called as FPT running time. A parameterized problem is said to be in the class para-NP if it has a nondeterministic algorithm with FPT running time. To show that a problem is para-NP-hard, we need to show that the problem is NP-hard even when the parameter takes a value from a finite set of positive integers. For example COLORING problem parameterized by solution size is para-NP-hard as determining 3-colorability of a graph NP-hard. We refer interesting reader to [9], [11] for further discussions on parameterized complexity.

Structural Parameterization: *Vertex cover* of a graph is set $X \subseteq V(G)$ such that for every edge uv at least one of u or v is contained in X. In other words, $G - X$ is an independent set. A *tree decomposition* of a graph G is a pair $\mathcal{T} = (T, \{X_t\}_{t \in V(T)})$, where T is a tree whose every node t is assigned a vertex subset $X_t \subseteq V(G)$, called a bag, such that the following three conditions hold : (i) $\bigcup_{t \in V(T)} X_t = V(G)$. (ii) For every $uv \in E(G)$, there exists a node t of T such that bag X_t contains both u and v. (iii) For every $u \in V(G)$, the set $T_u = \{t \in V(T) \mid u \in X_t\}$ induces a connected subtree of T. The width of tree decomposition $\mathcal{T} = (T, \{X_t\}_{t \in V(T)})$ equals $\max_{t \in V(T)}\{|X_t| - 1\}$. The treewidth of a graph G, denoted by $tw(G)$, is the minimum possible width of a tree decomposition of G.

3 Elementary Results

In this section we state some observations related to (k, i)-coloring of general graph and on perfect graphs.

3.1 (k, i)-coloring on general graph

We omit the simple proof of the following two observations.

Observation 1 *For a given graph G and its (k, i)-coloring function $f : V(G) \to [q]^k$, let $C \subseteq [q]$ be the set of any $i + 1$ colors. If $U := \{u \in V(G) | \ C \subseteq f(u)\}$ then U is an independent set in the graph.*

Observation 2 *For a given graph G and its induced subgraph H, $\chi_k^i(H) \leq \chi_k^i(G)$.*

Observation 3 ($\star$) [3] *For a given graph G, the following bounds hold –*

1. *$2k - i \leq \chi_k^i(G)$ when G has an edge.*
2. *$\chi_k^i(G) \leq \chi_{k-i}^0(G) + i$.*
3. *$\chi_k^0(G) \leq k \cdot \chi_1^0(G)$.*

[3] Results marked with a $\star$ have their proofs in the full version of this paper.

3.2 Perfect graphs and (k, i)-coloring

Perfect graphs were defined by Berge in 1960 [3] as follows:

Definition 2. *Graph G is a perfect graph if for each of its induced subgraphs H, $\chi(H) = \omega(H)$.*

A *hole* is an induced cycle of length at least four. An *antihole* is a graph whose complement is a hole. It is easy to see that if G is perfect graph then it does not contain an induced hole or an antihole of length greater than or equal to 5. Berge conjectured the following statement in 1961 which has been resolved in a celebrated result in 2002 [8].

Strong perfect graph conjecture: G is a perfect graph if and only if G does not have induced odd holes or odd anti-holes of length greater than or equal to 5. In [10], authors have introduced a concept of (k, i)-perfect graphs. We first define (k, i)-clique number which will be used in defining (k, i)-perfect graphs.

Definition 3. *The (k, i)-clique number of a graph G is the (k, i)-chromatic number of its largest induced clique and it is denoted by $\omega_k^i(G)$.*

In other words, $\omega_k^i(G) = \chi_k^i(K_{\omega(G)})$. (k, i)-perfect graphs are defined as follows.

Definition 4. *A graph G is a (k, i)-perfect graph if for each of its induced subgraphs H, $\chi_k^i(H) = \omega_k^i(H)$.*

For $(k, i) = (1, 0)$ this definition coincides with the Berge's definition. Authors of [10] conjectured the following statement and proved the *if* implication.

Conjecture 5.1: G is $(k, 0)$-perfect for $k \geq 1$ if and only if G does not have induced odd holes or antiholes of length greater than or equal to 5.

Proposition 1 (Lemma 5.1 of [10]). *The odd holes of length greater than or equal to 5 and their complements are not $(k, 0)$-perfect graphs.*

With the following Lemma and using the proof of strong perfect graph conjecture, we prove the reverse direction concluding that this conjecture is true.

Lemma 1 ($\star$). *If G is a perfect graph then $\chi_k^i(G) = \chi_k^i(K_{\omega(G)})$.*

Lemma 2 ($\star$). *If graph G does not have induced odd holes or odd antiholes of length greater than or equal to 5 then G is $(k, 0)$-perfect graph.*

4 Exact Algorithms

For a given graph G, integer q and a coloring function $f : V(G) \to [q]^k$, one can check whether or not f is a proper (k, i)-coloring of G in $\mathcal{O}(|E(G)| \cdot k)$ time. For a given integer q, there are $\binom{q}{k}$ many choices of k-tuples, which a function

f can assign to a vertex v in $V(G)$. Hence the number of different coloring functions is $\binom{q}{k}^n$. By Observation 3, $\chi_k^i(G) \leq k\chi(G)$ and we know that $\chi(G) \leq n$. Brute force algorithm exhaustively searches through the all possible coloring functions $f : V(G) \to [q]^k$ for values of $q \in [kn]$ and returns the minimum value of q for which it finds a valid (k,i)-coloring of graph G. This algorithm runs in time $\mathcal{O}(2^{nk\log(nk)} \cdot n^{\mathcal{O}(1)})$. We present an exact algorithm which runs in time $\mathcal{O}(2^{kn}n^{\mathcal{O}(1)})$ to find $(k,0)$-chromatic number of graph G. We generalize the idea to present an algorithm running in time $\mathcal{O}(2^{kn} \cdot k^{im}n^{\mathcal{O}(1)})$ to find (k,i)-chromatic number. This algorithm out performs the brute force algorithm mentioned above when the number of edges in graph are linearly bounded by the number of vertices. Finally, we present an algorithm running in time $\mathcal{O}(4^n)$ to find $(k,k\text{-}1)$-chromatic number of a graph (which is an NP-complete problem [10]). Note that running time of this algorithm is independent of k.

4.1 Computing $(k,0)$-Chromatic Number

In $(k,0)$-coloring, adjacent vertices should be assigned disjoint color-sets. Such coloring is also known as k-tuple coloring and it is proved to be NP-complete for any value of $k \geq 3$ [14].

For a given graph G, construct an auxiliary graph G' as follows: Graph G' contains k copies of graph G indexed by integers $\{1,2,\ldots,k\}$. Every vertex $u \in V(G)$ has its k copies $\{u_1,u_2,\ldots,u_k\}$ in graph G'. Construct clique, K_k^u, in G' on the vertices $\{u_1,u_2,\ldots,u_k\}$ for every vertex $u \in V(G)$. If there is an $uv \in E(G)$, make each vertex in K_k^u adjacent with every vertex in K_k^v. In other words, $K_k^u \cup K_k^v$ is a clique of size $2k$. This finishes the construction. Note that the graph G' has kn vertices.

Lemma 3. $\chi_k^0(G) = \chi(G')$

Proof. Let $f' : V(G') \to [\chi(G')]$ be an optimal proper coloring of graph G'. We define a function $f : V(G) \to [\chi(G')]^k$ as $f(u) = \{f'(u_1) \mid u_1 \in K_k^u\}$. We argue that f is a valid $(k,0)$-coloring of G. Since f' is a proper coloring of G', for $u_1,u_2 \in K_k^u$, $f'(u_1) \neq f'(u_2)$ and hence $|f(u)| = k$ for all $u \in V(G)$. Suppose f is not a valid coloring and there exists edge $uv \in E(G)$ such that $|f(u) \cap f(v)| > 0$. Hence there exists two vertices $u_1 \in K_k^u$ and $v_1 \in K_k^v$ such that $f'(u_1) = f'(v_1)$. Since $uv \in E(G)$, by construction $K_k^u \cup K_k^v$ is a clique. This is contradiction to the fact that f' is proper coloring of G'. This proves that $\chi_k^0(G) \leq \chi(G')$.

We now prove that $\chi(G') \leq \chi_k^0(G)$. Let $f : V(G) \to [q]^k$ be an optimal (k,i)-coloring for graph G where $q = \chi_k^0(G)$. We construct function $f' : V(G') \to [q]$ by constructing a bijective map between the vertices in K_k^u and $f(u)$. For any edge $u'v' \in E(G')$, either $u'v' \in K_k^u$ for some u or $u' \in K_k^u$ and $v' \in K_k^v$ and $uv \in E(G)$. In first case, $f'(u) \neq f'(v)$ as f' is bijection from K_k^u to $f(u)$. In second case, since $|f(u) \cap f(v)| = 0$, and $f'(u) \in f(u), f'(v) \in f(v)$, $f'(u) \neq f'(v)$. This implies that $\chi(G') \leq \chi_k^0(G)$ which concludes the proof. □

Proposition 2 ([16]). *For an n-vertex graph G, there exists an algorithm running in time $\mathcal{O}(2^n \cdot n^{\mathcal{O}(1)})$ which computes its chromatic number.*

Combining Lemma 3 with Proposition 2, we obtain the following result.

Theorem 1. *For an n-vertex graph G, there exists an algorithm running in time $\mathcal{O}(2^{kn} \cdot n^{\mathcal{O}(1)})$ which computes its $(k, 0)$-chromatic number.*

4.2 Computing (k, i)-Chromatic Number

We generalize the idea used in the above construction to obtain the (k, i)-chromatic number of the given graph G. Now instead of one, we construct $\mathcal{O}(k^{2im})$ many auxiliary graphs each of which is on kn vertices.

For every edge $e = uv \in E(G)$, select an index set $I_e \subseteq [k]$ of cardinality i. Let $(I_1, I_2, \ldots, I_m)$ be a m-tuple of indices selected. For every such m-tuple, we first construct an auxiliary graph G' as in Section 4.1. If $uv \in E(G)$ then delete an edge $u_l v_l$ from graph G' for all $l \in I_e$. Let $\mathcal{G}$ be the set of different graphs created using this operation. The number of such m-tuples are bounded by $\binom{k}{i}^m$ and hence $|\mathcal{G}| \leq \mathcal{O}(k^{im})$. Notice that if $uv \in E(G)$ then there are at most i-many vertices in K_k^u which are not adjacent to some vertex in K_k^v.

Lemma 4 ($\star$). $\chi_k^i(G) = \min_{G' \in \mathcal{G}}\{\chi(G')\}$

Combining Lemma 4 with Proposition 2 and the bound on $\mathcal{G}$, we obtain the following result.

Theorem 2. *For an n-vertex, m-edges graph G, there exists an algorithm running in time $\mathcal{O}(2^{kn} \cdot k^{im} \cdot n^{\mathcal{O}(1)})$ which computes its (k, i)-chromatic number.*

4.3 Computing $(k, k\text{-}1)$-Chromatic Number

For a given graph G, we construct an instance of Set Cover by setting $U = V(G)$ and $\mathcal{F}$ as the family of all independent sets of $V(G)$. Notice that $|\mathcal{F}| \leq 2^n$. Let r be the cardinality of a minimum solution of the Set Cover instance for $(U, \mathcal{F})$.

Claim. If $k - i = 1$ and q is the smallest integer such that $r \leq \binom{q}{i+1}$ then $\chi_k^i(G) = q$.

Proof. Suppose $\mathcal{F}' \subseteq \mathcal{F}$ is an optimum solution for Set Cover of cardinality r. Define an injective function $\psi : \mathcal{F}' \to [q]^{i+1}$ by assigning each set S in $\mathcal{F}'$ an $i+1$ sized set from $[q]$. Since $r \leq \binom{q}{i+1}$, such injective function is possible. We now define a function $f : V(G) \to [q]^k$ as $f(v) = \psi(S)$ where S is any set in $\mathcal{F}'$ such that $u \in S$. Since $\mathcal{F}'$ is a cover of $V(G)$, for every vertex u this function assigns k colors to each vertex. We now prove that this is indeed a proper (k, i)-coloring of graph. Suppose not, then there exists an edge uv such that $|f(u) \cap f(v)| \geq i + 1$. Since $|f(u)| = |f(v)| = k = i + 1$, this implies $f(u) = f(v)$. By construction it implies that u, v are contained in same set S. This is contradiction to the fact that S is an independent set. Hence f is proper (k, i)-coloring of graph G. This implies that $\chi_k^i(G) \leq q$.

Suppose there exists a (k, i)-coloring $f' : V(G) \to [q']$ of graph G using $q' < q$

colors. By Observation 1, for any set X of $[q']$ which is of cardinality $i+1$, set $U := \{u \mid X \subseteq f'(u)\}$ is an independent set of graph G. We say that color set X characterizes vertex set U. Construct $\mathcal{F}'' = \{U \mid \exists X \subseteq [q]$ of cardinality $i+1$ which characterizes $U\}$. Since there are at most $\binom{q}{i+1}$ such color set X, $|\mathcal{F}''| \leq \binom{q'}{r+1} < r$ as q is the smallest integer such that $r \leq \binom{q}{i+1}$. This contradicts the fact that r is cardinality of a minimum solution of Set Cover. Hence our assumption is wrong and $q \leq \chi_k^i(G)$ which completes the proof. □

Proposition 3. *[13] For any given instance $(U, \mathcal{F})$ of* Set Cover *problem, there exists an algorithm which solves it in time $\mathcal{O}(2^n n|\mathcal{F}|)$ where $|U| = n$.*

Combining the above claim, Proposition 3 and using the bound that $|\mathcal{F}| \leq 2^n$, we get following result.

Theorem 3. *For an n-vertex graph G, there exists an algorithm running in time $\mathcal{O}(4^n \cdot n^{\mathcal{O}(1)})$ which computes its $(k, k\text{-}1)$-chromatic number.*

5 Fixed-Parameter Algorithms

Parameterization of a problem is assigning a positive integer, called parameter, to each of its input instance. One of the most natural choice for a parameter is the solution size which in this case is the number of colors needed for (k, i)-coloring. For a given graph G, it is NP-hard to determine whether it can be colored with at most 3 colors. This implies that Coloring parameterized by number of colors in para-NP-hard. Hence we can not expect (k, i)-Coloring to be FPT when parameterized by the number of colors. But, Coloring is FPT when parameterized by several structural properties of the input graph. Notion of treewidth was introduced by Roberson and Seymour. It is known that Coloring parameterized by treewidth of graph and number of colors is FPT. Structural Parameterizations of classical graph coloring problem was studied by Jansen and Kratsch [15] (also studied in [7, 12]). They proved that when input is a graph G with its vertex cover $Y \subseteq V(G)$ and the parameter is $|Y|$, finding the chromatic number of G is FPT. In this section, We generalize these results of classical graph coloring problem to (k, i)-coloring problem.

5.1 (k, i)-Coloring Parameterized by Vertex Cover

In this sub-section, we present an FPT algorithm for finding $\chi_k^i(G)$ when parameterized by size of a vertex cover of the input graph.

In case of structural parameters, sometimes it is necessary to demand a witness of the required structure as part of the input. However, when the size of a vertex cover is the parameter, this is not a serious demand. If given only a input graph, one can find a 2-approximation of the minimum vertex cover of the input graph G(pp 11, [17]). Thus, we may assume that we are solving the following problem.

(k,i)-Coloring	**Parameter:** $\|Y\|$
Input: Graph $G, Y \subseteq V(G)$ such that Y is a vertex cover of G	
Output: $\chi_k^i(G)$	

Theorem 4. *For an n-vertex graph G and its vertex cover Y, there exists an algorithm running in time $\mathcal{O}(2^{k|Y|\log(k|Y|)} \cdot kn^2)$ which computes $\chi_k^i(G)$.*

Proof. For a given graph G on n vertices, the algorithm iterates over all possible (k,i)-colorings of $G[Y]$. Since $\chi_k^i(G[Y]) \le k \cdot |Y|$, there are $\mathcal{O}(2^{k|Y|\log(k|Y|)})$ many such possible colorings. For every valid (k,i)-coloring f of $G[Y]$, we extend this coloring function to the rest of the graph in the greedy fashion. For every vertex $u \in V(G) \setminus Y$, f assigns k smallest colors to u such that for any $v \in N(u)$, $|f(u) \cap f(v)| \le i$. Since u is in an independent set, all of its neighbors have been assigned colors before function assigns k colors to u. This extension of valid coloring can be computed in $\mathcal{O}(kn^2)$ time to obtain a (k,i)-coloring of graph G. The algorithm returns the minimum number of colors used over all the valid (k,i)-coloring of graph G. The running time of this algorithm is $\mathcal{O}(2^{k|Y|\log(k|Y|)} \cdot kn^2)$ which is FPT when parameterized by cardinality of vertex cover. We now argue the correctness of the algorithm.

If the algorithm returns q as the minimum number of colors used over all the valid (k,i)-colorings of graph G, by construction it is clear that $\chi_k^i(G) \le q$. We now prove that $q \le \chi_k^i(G)$ using contradiction. Suppose $\chi_k^i(G) < q$. This implies that for every (k,i)-coloring f of $V(G)$ which is obtained as extension of valid (k,i)-coloring of $G[Y]$, there exists a vertex u such that $q \in f(u)$. Let $f^* : V(G) \to [\chi_k^i(G)]^k$ is a optimum (k,i)-coloring of graph G. Since we are iterating over all possible coloring of $G[Y]$, one of them is $f^*|_Y$. Let f' is an greedy extension of $f^*|_Y$ to entire graph. Hence there exists a vertex v such that $q \in f'(v)$. By Observation 2, $\chi_k^i(G[N[v]]) \le \chi_k^i(G)$. Since f' is obtained greedily as extension of $f^*|_Y$, for every k-sized set X of $\{1, 2, \ldots, \chi_k^i(G)\}$, there exists $u \in N(v)$ such that $|X \cap f^*(u)| \ge i+1$ which forced algorithm to use a color in $\{\chi_k^i(G), \ldots, q\}$ while constructing extension of $f^*|_Y$. This contradicts the fact that $f^*|_{N[v]}$ is a valid (k,i)-coloring of $N[v]$ which uses at most $\chi_k^i(G)$ colors. □

5.2 q-(k,i)-Coloring Parameterized by Treewidth

In this sub-section, we present an FPT algorithm for finding whether $\chi_k^i(G)$ is at most q when parameterized by treewidth of input graph. Notice that, unlike previous section, we assume that q is fixed and it is not part of input. Formally, we study the following problem.

q-(k,i)-Coloring	**Parameter:** tw
Input: Graph G with its tree decomposition $\mathcal{T}$ of width tw	
Output: Is $\chi_k^i(G) \le q$?	

We know that given a tree decomposition $\mathcal{T}' = (T', \{Y_t\}_{t \in V(T')})$, it can be converted into a *nice tree decomposition* $\mathcal{T} = (T, \{X_t\}_{t \in V(T)})$ in polynomial time

(for definition and other details, see Chapter 7 of [9]) where every node is one of the following types and has at most 2 children. For a *nice tree decomposition*, we distinguish one vertex r of T which will be the root of T.
Root Node: r is the root node where $X_r = \emptyset$.
Leaf Node: If $t \in V(T)$ is a leaf node, then $X_t = \emptyset$.
Introduce Node: If $t \in V(T)$ is an introduce node then t' is the only child of t in T and $X_t = X_{t'} \cup \{u\}$ where $u \notin X_{t'}$.
Forget Node: If $t \in V(T)$ is a forget node then t' is the only child of t in T and $X_t = X_{t'} \setminus \{u\}$ where $u \in X_{t'}$.
Join Node: If $t \in V(T)$ is a join node then t_1 and t_2 are the children of t in T and $X_t = X_{t_1} = X_{t_2}$.

We compute and store two values for every node $t \in V(T)$. These are $\mathcal{C}(t)$ and $\mathcal{D}(t)$ and they are defined as follows.

$\mathcal{C}(t) = \{f : X_t \to [q]^k | f$ is a proper (k,i)-coloring of $G[X_t]\}$.

$\mathcal{D}(t) = \{f \in \mathcal{C}(t) | f$ is extendable to a proper (k,i)-coloring of $G_t\}$.

We can compute $\mathcal{C}(t)$ for every $t \in V(T)$ independent of their children. But, $\mathcal{D}(t)$ needs to be computed by using $\mathcal{D}(t_1), \mathcal{D}(t_2)$ where t_1, t_2 are children of t.
Leaf Node: When a node $t \in V(T)$ is a *leaf node*, then $X_t = \emptyset$. So, $\mathcal{C}(t) = \{\emptyset\}$. $\mathcal{D}(t) = \mathcal{C}(t)$.
Introduce Node: When a node $t \in V(T)$ is an *introduce node* with only child t', and let $X_t = X_{t'} \cup \{u\}$ for $u \notin X_{t'}$.
$\mathcal{D}(t) = \{f \in \mathcal{C}(t) | \exists g \in \mathcal{D}(t')$ such that $g \equiv f|_{X'_t}\}$. Correctness is clear from construction as only feasible colorings are stored and all of them extend to a feasible coloring of the induced subgraph.
Forget Node: When a node $t \in V(T)$ is a *forget node* with only child t' and let $X_t = X_{t'} \setminus \{u\}$ for $u \in X_{t'}$. We say that $\mathcal{D}(t)$ is the projection of all the members of $\mathcal{D}(t')$ at t. Formally $\mathcal{D}(t) = \{f \in \mathcal{C}(t) | f \equiv g|_{X_t}$ where $g \in \mathcal{D}(t')\}$. Correctness of this is clear because $G_t = G_{t'}$.
Join Node: When a node $t \in V(T)$ is a *join node* with children t_1 and t_2, then $X_t = X_{t_1} = X_{t_2}$. We say $\mathcal{D}(t) = \mathcal{D}(t_1) \cap \mathcal{D}(t_2)$.
It is clear that if $f \in \mathcal{D}(t)$, then $f \in \mathcal{D}(t_1)$ and $f \in \mathcal{D}(t_2)$. So, $\mathcal{D}(t) \subseteq \mathcal{D}(t_1) \cap \mathcal{D}(t_2)$ as (k,i)-coloring is feasible for induced subgraphs. G_{t_1} and G_{t_2} are induced subgraphs of G_t. We now justify that $f \in \mathcal{D}(t_1) \cap \mathcal{D}(t_2) \subseteq \mathcal{D}(t)$. Let $f \in \mathcal{D}(t_1) \cap \mathcal{D}(t_2)$. Then f is a proper (k,i)-coloring in G_{t_1} and also G_{t_2}. By connectivity property (Property $T3$ in Chapter 7 of [9]) of tree decomposition, we know that there is no edge between two vertices one of which is in $G_{t_1} \setminus X_{t_1}$ and the other is in $G_{t_2} \setminus X_{t_2}$. So, $f \in \mathcal{D}(t)$ as well.
Now, we describe how to compute $\mathcal{D}(t)$ from $\mathcal{D}(t_1)$ and $\mathcal{D}(t_2)$ where t_1, t_2 are the children of t. $\mathcal{C}(t)$ can be computed in $\binom{q}{k}^{tw+1}$ time for every $t \in V(T)$ and this is independent of its children in the tree decomposition. We have the following lemma.

Lemma 5. *For every $t \in V(T)$, $\mathcal{D}(t)$ and $\mathcal{C}(t)$ can be computed in $\mathcal{O}^*(\binom{q}{k}^{tw})$ time.*

Proof. We prove this statement for each type of nodes.
Leaf Node: $t \in V(T)$ is a *leaf node*. Then, $|X_t| = 0$ and hence it is trivial as $\mathcal{D}(t) = \mathcal{C}(t)$.
Introduce Node: Let $t \in V(T)$ be an *introduce node* where $X_t = X_{t'} \cup \{u\}$. u is the only new vertex in where a color of k tuple has to be assigned. For every $R \in \binom{[q]}{k}$, for every $f' \in \mathcal{D}(t')$, we check if f' can be extended to $f : X_t \to [q]^k$ by assigning R to t. This takes $\binom{q}{k} \cdot |\mathcal{D}(t')| = \binom{q}{k}^{tw+1}$ time.
Forget Node: Let $t \in V(T)$ be a *forget node* where $X_t = X_{t'} \setminus \{u\}$. u is the vertex which was in t' but not in t. Then, we copy all the colorings of $\mathcal{D}(t')$ to $\mathcal{D}(t)$ where color tuple of the vertex u is not mentioned and then remove the redundant copies. Removing redundant copies can also be done in $\mathcal{O}(|\mathcal{D}(t')| \log_2 |\mathcal{D}(t')|) = \mathcal{O}(\binom{q}{k}^{tw} \cdot poly(n, tw, k))$ time by sorting all the members of $\mathcal{D}(t)$ in lexicographic order and identifying repetitions.
Join Node: Let $t \in V(T)$ be a *join node* where $X_t = X_{t_1} = X_{t_2}$. If we compute the intersection of two sets in a very naive way, then we will spend $|\mathcal{D}(t_1)| \cdot |\mathcal{D}(t_2)|$ time. That's why we again sort both $\mathcal{D}(t_1)$ and $\mathcal{D}(t_2)$ separately and then compute the intersection in $\mathcal{O}(|\mathcal{D}(t_1)| + |\mathcal{D}(t_2)|)$ time. This procedure takes $\mathcal{O}(|\mathcal{D}(t)| \cdot \log_2 |\mathcal{D}(t)|) = \mathcal{O}(\binom{q}{k}^{tw} \cdot poly(n, tw, k))$ time as t is a join node. ◻

The following theorem follows from the above lemma.

Theorem 5. *Given an n-vertex graph G with its tree decomposition of width tw, q-(k,i)-*Coloring *can be solved in time $\mathcal{O}(q^{k \cdot tw} \cdot n^{\mathcal{O}(1)})$.*

Proof. From Lemma 5, $\mathcal{D}(t)$ and $\mathcal{C}(t)$ can be computed in time $\mathcal{O}(\binom{q}{k}^{tw} \cdot n^{\mathcal{O}(1)})$ time. Let the root node of the tree decomposition be r. We say that the instance is a Yes-Instance if and only if $\mathcal{D}(r) \neq \emptyset$. Clearly when $\mathcal{D}(r) \neq \emptyset$, there exists a proper (k,i)-coloring of G using at most q colors. But when $\mathcal{D}(r) = \emptyset$, then we see that no proper (k,i)-coloring of $G[X_r]$ is extendable to a proper coloring of G. Then it is a No-Instance. Therefore, the algorithm correctly decides in $\mathcal{O}(\binom{q}{k}^{tw} \cdot n^{\mathcal{O}(1)})$ time whether there exists (k,i)-coloring of G using q colors. ◻

6 Conclusions

We considered the (k,i)-coloring problem which is a generalization of proper coloring and is a well motivated problem from coding theory. Difficulty introduced by this generalization is evident by the fact that no polynomial time algorithm is known to optimally color a given clique. In this paper, we initiate a study of exact and parameterized algorithms for (k,i)-coloring. We provide exact algorithms running in time c^n for two cases viz $i = 0$ and $i = k - 1$. NP-hardness of (k,i)-Coloring for any $0 < i < k$ is still an open question. It is also interesting to find graph classes in which this problem can be solved in polynomial time. We prove that this problem is FPT when parameterized by *treewidth* with number of colors as a combined parameter. We also provide an FPT algorithm (without treewidth machinery) when parameterized by the size of *vertex cover* of the graph. It is an interesting open question whether we can get an FPT algorithm for (k, i)-coloring parameterized by the size of *feedback vertex set* of the graph that does not use treewidth machinery.

Acknowledgements: The second and third authors thank Debajyoti Ghosh for introducing the problem.

References

1. E. Agrell, A. Vardy, and K. Zeger. Upper bounds for constant-weight codes. *IEEE Transactions on Information Theory*, 46(7):2373–2395, 2000.
2. A.Hilton, R. Rado, and S. Scott. A (< 5) color theorem for planar graph. *Bulletin of London Mathematical Society*, 5:302–306, 1973.
3. C. Berge. Les problemes de coloration en théorie des graphes. *Publ. Inst. Stat. Univ. Paris*, 9:123–160, 1960.
4. B. Bollobas and A. Thomason. Set colourings of graphs. *Discrete Mathematics*, 25(1):2126, 1979.
5. F. Bonomo, G. Duran, I. Koch, and M. Valencia-Pobon. On the (k, i)-coloring of cacti and complete graphs. *Ars Combinatorica*, 2014.
6. R. Brigham and R. Dutton. Generalized k-tuple colorings of cycles and other graphs. *Journal of Combinatorial Theory. Series B*, 32:9094, 1982.
7. L. Cai. Parameterized complexity of vertex colouring. *Discrete Applied Mathematics*, 127(3):415–429, 2003.
8. M. Chudnovsky, N. Robertson, P. Seymour, and R. Thomas. The strong perfect graph theorem. *Annals of mathematics*, pages 51–229, 2006.
9. M. Cygan, F. V. Fomin, L. Kowalik, D. Lokshtanov, D. Marx, M. Pilipczuk, M. Pilipczuk, and S. Saurabh. *Parameterized Algorithms*, volume 4. Springer, 2015.
10. I. M. Díaz and P. Zabala. A generalization of the graph coloring problem. *Investigation Operativa*, 1999.
11. R. G. Downey and M. R. Fellows. *Parameterized complexity*. Springer Science & Business Media, 2012.
12. J. Fiala, P. A. Golovach, and J. Kratochvíl. Parameterized complexity of coloring problems: Treewidth versus vertex cover. *Theor. Comput. Sci.*, 412(23):2513–2523, 2011.
13. F. V. Fomin and D. Kratsch. *Exact Exponential Algorithms*. Texts in Theoretical Computer Science. An EATCS Series. Springer, 2010.
14. R. W. Irving. NP-completeness of a family of graph-colouring problems. *Discrete Applied Mathematics*, 5(1):111–117, 1983.
15. B. M. P. Jansen and S. Kratsch. Data reduction for graph coloring problems. *Inf. Comput.*, 231:70–88, 2013.
16. M. Koivisto. An $\mathcal{O}^*(2^n)$ algorithm for graph coloring and other partitioning problems via inclusion–exclusion. In *FOCS*, pages 583–590. IEEE, 2006.
17. C. H. Papadimitriou and K. Steiglitz. *Combinatorial optimization: algorithms and complexity*. Courier Corporation, 1982.
18. S. Stahl. n-tuple colorings and associated graphs. *Journal of Combinatorial Theory. Series B*, 20:185–203, 1976.