B-Chromatic Number: Beyond NP-Hardness*

Fahad Panolan[1], Geevarghese Philip[2], and Saket Saurabh[1,3]

1 **Institute of Mathematical Sciences, Chennai, India**
{fahad|saket}@imsc.res.in
2 **Chennai Mathematical Institute, India**
gphilip@cmi.ac.in
3 **University of Bergen, Norway**

Abstract

The b-chromatic number of a graph G, $\chi_b(G)$, is the largest integer k such that G has a k-vertex coloring with the property that each color class has a vertex which is adjacent to at least one vertex in each of the other color classes. In the b-Chromatic Number problem, the objective is to decide whether $\chi_b(G) \geq k$. Testing whether $\chi_b(G) = \Delta(G) + 1$, where $\Delta(G)$ is the maximum degree of a graph, itself is NP-complete even for connected bipartite graphs (Kratochvíl, Tuza and Voigt, WG 2002). In this paper we study b-Chromatic Number in the realm of parameterized complexity and exact exponential time algorithms. We show that b-Chromatic Number is W[1]-hard when parameterized by k, resolving the open question posed by Havet and Sampaio (Algorithmica 2013). When $k = \Delta(G) + 1$, we design an algorithm for b-Chromatic Number running in time $2^{\mathcal{O}(k^2 \log k)} n^{\mathcal{O}(1)}$. Finally, we show that b-Chromatic Number for an n-vertex graph can be solved in time $\mathcal{O}(3^n n^4 \log n)$.

1998 ACM Subject Classification F.2 Analysis of Algorithms and Problem Complexity

Keywords and phrases b-chromatic number, exact algorithm, parameterized complexity

Digital Object Identifier 10.4230/LIPIcs.IPEC.2015.389

1 Introduction

Graph coloring (proper vertex coloring), is an assignment of colors to the vertices of a graph such that no edge connects two identically colored vertices. In other words graph coloring is a partition of vertex set into independent sets. A proper vertex coloring using k colors is called a *k-vertex coloring*. The least number of colors required for a proper vertex coloring of a graph G is called the *chromatic number* of G. The most common question about graph coloring is–"what is the chromatic number of a graph". This question has got lots of attention in graph theory and algorithms. The study of graph coloring leads to the four color theorem in planar graphs by Appel and Haken [1], study of chromatic polynomial introduced by Birkhoff, which was generalised to the Tutte polynomial by Tutte, etc. Graph coloring has been studied as an algorithmic problem since the early 1970s. The chromatic number problem is one of Karp's 21 NP-complete problems from 1972 [11]. An exact algorithm to compute the chromatic number of a graph dates back to 1976. Lawler [13] gave an algorithm for finding chromatic number running in time $2.4423^n n^{\mathcal{O}(1)}$. Finally, after 30 years, using the principle of inclusion-exclusion Björklund et al. [2] gave an algorithm for chromatic number

* Supported by the European Research Council (ERC) via grants Rigorous Theory of Preprocessing, reference 267959 and PARAPPROX, reference 306992; and by the Department of Science and Technology (DST), Government of India, the German Federal Ministry of Education and Research (BMBF), and the Max Planck Society (MPG), via the Indo-German Max Planck Center for Computer Science (IMPECS).

problem running in time $2^n n^{\mathcal{O}(1)}$. This is still the fastest known exact algorithm to compute the chromatic number of a graph.

Not only finding chromatic number but also different variations of graph coloring has been studied in the literature. A *complete coloring* of a graph G is a proper vertex coloring such that no two color classes together form an independent set. The parameter *achromatic number* of a graph G is the largest integer k such that there is a complete coloring of G using k colors. Irving and Manlove [10] introduced *b-chromatic number*, another parameter related to graph coloring. The *b-chromatic number* of a graph G, denoted by $\chi_b(G)$, is the largest integer k such that G has a k-vertex coloring with the property that each color class has a vertex which is adjacent to at least one vertex in each of the other color classes. Such a coloring is called a *b-coloring*. Irving and Manlove showed that determining b-chromatic number is NP-complete for general graphs, but polynomial time solvable for trees [10]. From the definition of b-chromatic number it is clear that $\chi_b(G) \leq \Delta(G) + 1$, where $\Delta(G)$ is the maximum degree of the graph G. Kratochvíl et al. [12] showed that determining whether $\chi_b(G) = \Delta(G)+1$ is NP-hard even for connected bipartite graphs. Havet et al. [8] showed that b-chromatic number can be computed in polynomial time for split graphs and it is NP-hard for connected chordal graphs. Regarding approximation algorithms for the problem, Galcík et al. [7] showed that b-chromatic number of an n-vertex graph can not be approximated within a factor $n^{1/4-\epsilon}$ for any constant $\epsilon > 0$, in polynomial time, unless P =NP.

In this work we address the algorithmic question of b-chromatic number in the realm of parameterized complexity and exact exponential time algorithms.

b-Chromatic Number	**Parameter:** k
Input: An n-vertex graph G and an integer k	
Question: Is the b-chromatic number of G is at least k	

For a detailed overview of parameterized complexity reader is referred to monographs [5, 4]. In the parameterized complexity framework, the b-Chromatic Number problem is studied with a dual parameter by Havet et al. [9]. In particular, they show that one can decide whether $\chi_b(G) \geq n - k$ in time $2^{\mathcal{O}(k \log k)} n^{\mathcal{O}(1)}$ and asked the question whether b-Chromatic Number is FPT when parameterized by k. Recently, Effantin et al. [6] studied a relaxed version of b-coloring and repeated the question about the parameterized complexity of b-Chromatic Number. In this work we answer this question negatively, by showing that b-Chromatic Number is W[1]-hard. But, when $k = \Delta(G)+1$, we design an algorithm for b-Chromatic Number running in time $2^{\mathcal{O}(k^2 \log k)} n^{\mathcal{O}(1)}$. Finally we show that b-Chromatic Number for an n-vertex graph can be solved in time $\mathcal{O}(3^n n^4 \log n)$. After the results in this article were presented at IPEC, Manfred Cochefert informed us that he had derived, in his PhD thesis [3] a $3^n n^{\mathcal{O}(1)}$ time algorithm for b-Chromatic Number using principle of inclusion-exclusion.

Our methods. To show b-Chromatic Number is W[1]-hard, when parameterized by k, we give an FPT-reduction from Multi Colored Independent Set, which is very well known to be W[1]-hard [4]. When $k = \Delta(G) + 1$, to get an FPT algorithm for b-Chromatic Number, we first show that it is enough to find $C \subseteq V(G)$ such that $\chi_b(G[C]) = k$ (we call such a subset C as b-chromatic core of order k). Then we give a polynomial kernel for the problem of finding b-chromatic core of order $\Delta(G) + 1$, which leads to an FPT algorithm for b-Chromatic Number when $k = \Delta(G) + 1$. For the exact exponential time algorithm for b-Chromatic Number, we reduce the problem to many instances of single variate polynomial multiplication of degree 2^n.

2 Preliminaries

We use "graph" to denote simple graphs without self-loops, directions, or labels. We use $V(G)$, $E(G)$ and $\Delta(G)$, respectively, to denote the vertex set, edge set and maximum degree of a graph G. We also use $G = (V, E)$ to denote a graph G on vertex set V and edge set E. For $v, u \in V(G)$ and $V' \subseteq V(G)$, we use $G[V']$ to denote the subgraph of G induced on V', $N[v] = \{u \ : \ (v, u) \in E(G)\} \cup \{v\}$ and $d(u, v)$ is the shortest distance between u and v. For a graph G and a b-coloring of G with color classes $C_1, \ldots, C_k$, we say a vertex $v \in C_i$ is a *dominating* vertex for the color class C_i if v is adjacent to a vertex in C_j for each $j \neq i$.

We use $[n]$ to denote the set $\{1, 2, \ldots, n\}$. We use $\uplus$ to denote the disjoint union of sets: for any two sets A, B, the set $A \uplus B$ is defined only if $(A \cap B) = \emptyset$, and in this case $(A \uplus B) = (A \cup B)$. We assume that $\uplus$ associates to the left; that is, we write $\biguplus_{1 \leq i \leq n} A_i = A_1 \uplus A_2 \uplus A_3 \cdots \uplus A_n$ to mean $(\cdots((A_1 \uplus A_2) \uplus A_3) \cdots \uplus A_n)$. Further, every use of $\uplus$ in an expression carries with it the implicit assertion that the two sets involved are disjoint.

A *binary vector* is a finite sequence of bits, and its *width* is the number of bits in the sequence. If A, B are binary vectors, then by $A + B$ we mean the *integer* $val(A) + val(B)$ where for a binary vector X the expression $val(X)$ denotes the integer of which X is a binary representation. Let $U = \{u_1, u_2, \ldots, u_n\}$ be a set of cardinality n, and let $S \subseteq U$. The *characteristic vector* $\chi(S)$ of S with respect to U is the binary vector with $|U| = n$ bits whose ℓ^{th} bit, for $1 \leq \ell \leq n$, is 1 if element u_ℓ belongs to set S, and 0 otherwise. We use $\mathbb{N}$ to denote the set of non-negative integers. The *Hamming weight* $\mathcal{H}(r)$ of a binary vector r is the number of 1s in r. For a finite set U, a subset $S \subseteq U$, and the characteristic vector $\chi(S)$ of S with respect to U, observe that $\mathcal{H}(\chi(S)) = |S|$. We define the Hamming weight of $n \in \mathbb{N}$ to be the number $\mathcal{H}(n)$ of the number of 1s in a binary representation of n. Note that an integer $n \in \mathbb{N}$ does not have a *unique* binary representation, since we can pad any such representation with zeroes on the left without changing its numerical value. We call the total number of bits in a binary representation r of n the *width* of r.

▶ **Lemma 1** ($\star$[1]). *Let S_1, S_2 be two disjoint subsets of a set $U = \{u_1, u_2, \ldots, u_n\}$, and let $S = S_1 \uplus S_2$. Then*

1. $\chi(S) = \chi(S_1) + \chi(S_2)$
2. $\chi(S_1) + \chi(S_2)$ *has a binary representation of width n*
3. $\mathcal{H}(\chi(S)) = \mathcal{H}(\chi(S_1)) + \mathcal{H}(\chi(S_2))$

We make extensive use of single-variate polynomials with integer coefficients. Let $P = \sum_{i=0}^{n} a_i x^i$ be such a polynomial. We say that polynomial P *contains* monomial x^i, or that monomial x^i is *present in* polynomial P, if the coefficient a_i of x_i is not zero. We say that the polynomial $P' = \sum_{i=0}^{n} b_i x^i$ is the *representative polynomial* of P if $b_i = 1$ whenever $a_i \neq 0$ and $b_i = 0$ otherwise. That is, the representative polynomial remembers just the *degrees* of the monomials which are present in P, and forgets their coefficients. For an $h \in \mathbb{N}$, we define the *Hamming projection* of polynomial P to h to be $\mathcal{H}_h(P) = \sum_{i=0}^{n} b_i x^i$ where $b_i = a_i$ if $\mathcal{H}(i) = h$ and 0 otherwise. That is, $\mathcal{H}_h(P)$ is the sum of all those monomials in P whose *degrees* have Hamming weight h. To obtain the stated running time for our exponential-time algorithm, we make use of the fast algorithm for multiplying polynomials which is based on the Fast Fourier Transform.

[1] Proofs of results marked with a $\star$ are deferred to the full version of the paper.

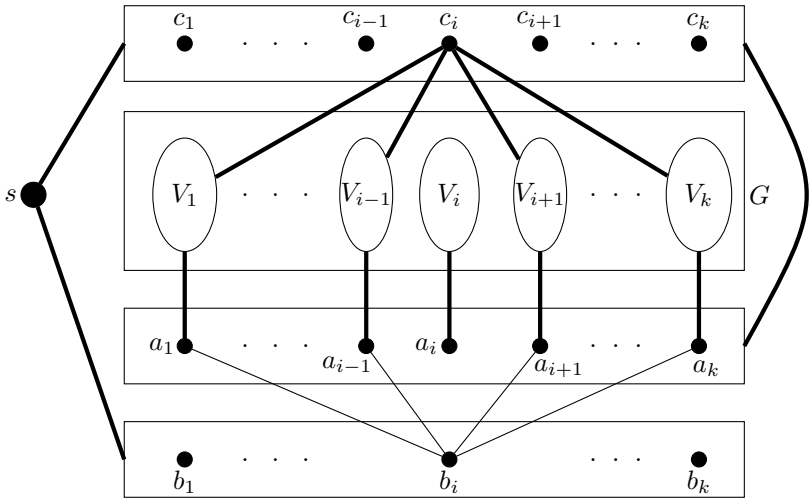

Figure 1 The graph G' constructed from the input instance $G = (V_1 \uplus \ldots \uplus V_k, E)$ of Multi-Colored Independent Set. The thick edges represent all possible edges between two corresponding sets of vertices.

▶ **Lemma 2** ([14]). *Two polynomials of degree at most n over any commutative ring $\mathcal{R}$ can be multiplied using $\mathcal{O}(n \log n \log\log n)$ additions and multiplications in $\mathcal{R}$.*

3 Hardness

In this section we show that b-Chromatic Number is W[1]-hard by giving an FPT-reduction from Multi-Colored Independent Set.

Multi-Colored Independent Set	**Parameter:** k
Input: A k-partite graph G with its k-partition $V_1 \uplus \ldots \uplus V_k$ of $V(G)$	
Question: Is there an independent set of size k containing one vertex from each V_i?	

▶ **Theorem 3.** *There is a polynomial time algorithm that given an instance $G = (V_1 \uplus \ldots \uplus V_k, E)$ of* Multi-Colored Independent Set*, constructs an instance $(G', 2k+1)$ of* b-Chromatic Number *such that G is a* Yes *instance of* Multi-Colored Independent Set *if and only if $(G', 2k+1)$ is a* Yes *instance of* b-Chromatic Number*.*

Proof. Let G be an instance of Multi-Colored Independent Set, with its k-partition $V_1 \uplus \ldots \uplus V_k$.

Construction. We construct a graph G' from G as follows. The vertex set of G', $V(G') = V(G) \cup A \cup B \cup C \cup \{s\}$, where $A = \{a_1, \ldots, a_k\}, B = \{b_1, \ldots, b_k\}$ and $C = \{c_1, \ldots, c_k\}$. The edge set $E(G')$ contains $E(G)$ and the following sets of new edges (see Figure 1).

- $\{(a_i, a_j) \mid i \neq j\} \cup \{(c_i, c_j) \mid i \neq j\}$ (i.e A and C forms cliques),
- $\{(a_i, c_j) \mid 1 \leq i, j \leq k\}$ (i.e $A \cup C$ forms a clique),
- $\{(a_i, b_j) \mid i \neq j\}$,
- $\{(a_i, v) \mid v \in V_i\} \cup \{(c_i, v) \mid v \in V(G) \setminus V_i\}$,
- $\{(s, b_i), (s, c_i) \mid 1 \leq i \leq k\}$.

Completeness. Suppose G is an YES instance of MULTI-COLORED INDEPENDENT SET. Let $I = \{v_1, \ldots, v_k\}$ be an independent set such that $v_i \in V_i$ for all $i \in [k]$. Now we give a b-coloring of G' using $2k+1$ colors as follows. We define $2k+1$ color classes – $C_0, C_1, \ldots, C_{2k}$. The color class $C_0 = I \cup \{s\}$. For all $1 \leq i \leq k$, $C_i = (V_i \cup \{c_i\}) \setminus \{v_i\}$. For all $k+1 \leq i \leq 2k$, $C_i = \{a_i, b_i\}$. Note that for all $0 \leq i \leq 2k$, C_i is an independent set in G' and $C_0 \uplus C_1 \uplus \ldots \uplus C_{2k} = V(G')$. Now we show that each color class has a dominating vertex. For color class C_0, the vertex s is a dominating vertex, because s is adjacent to all the vertices in the set $B \cup C$. For each $1 \leq i \leq k$, the vertex c_i is the dominating vertex of the color class C_i, because it is adjacent to the vertices $A \cup \{s\}$ and $C \setminus \{c_i\}$. For each $k+1 \leq i \leq 2k$, a_i is the dominating vertex for the color class C_i, because it is adjacent to all the vertices in $C \cup (A \setminus \{a_i\})$ and the vertex v_i in the color class C_0.

Soundness. Let $(G', 2k+1)$ is an YES instance of b-CHROMATIC NUMBER. Let ϕ be a b-coloring for the graph G' using at least $2k+1$ colors. Since $A \cup C$ is a clique, all the vertices in $A \cup C$ are colored differently by ϕ. For $1 \leq i \leq k$, let C_i be the the color class which contains the vertex c_i and for each $k+1 \leq i \leq 2k$, let C_i be the color class which contains the vertex a_i. Let C_0 be an arbitrary color class other that $C_1, \ldots, C_{2k}$ in the coloring ϕ.

First note that since the degree of any vertex in B is k, no vertex in B can be a dominating vertex for any color class. Now consider the following claim.

▶ **Claim 1.** *No vertex $u \in V(G)$ can be a dominating vertex for any of the color classes $C_0, C_{k+1}, C_{k+2}, \ldots, C_{2k}$.*

Proof. Suppose $u \in V(G)$ is a dominating vertex for the color class C_j for some $j \in \{0, k+1, k+2, \ldots, 2k\}$. Let $u \in V_i$ for some $1 \leq i \leq k$. Consider the color class C_i. Note that $c_i \in C_i$. This implies that $C_i \subseteq V_i \cup B$, because the non-neighborhood of c_i is $V_i \cup B$. But since $u \in V_i$, u is not adjacent to any vertex in $V_i \cup B$. This contradicts that u is a dominating vertex for C_j. ◀

Since C_0 is disjoint from $A \cup C$ and by Claim 1, we can conclude that s is the dominating vertex for C_0. Also note that for any $k+1 \leq i \leq 2k$, C_i is disjoint from $(A \cup C \cup \{s\}) \setminus \{a_i\}$. By Claim 1, this implies that a_i is the dominating vertex for the class C_i. Since a_i is a dominating vertex for the class C_i for each $k+1 \leq i \leq 2k$, there is a vertex v in C_0 such that $(a_i, v) \in E(G')$. Since $C_0 \cap (A \cup B \cup C) = \emptyset$ (because $s \in C_0$), we have that $v \in V_{i-k}$. This implies that for each $k+1 \leq i \leq 2k$, there is a vertex $v \in V_{i-k}$ such that $v \in C_0$. This implies that G has an independent set of size k containing one vertex from each V_j for $1 \leq j \leq k$. This completes the proof of the lemma. ◀

4 FPT Algorithm for deciding whether $\chi_b(G) = \Delta(G) + 1$

In this section we design a parameterized algorithm for b-CHROMATIC NUMBER to decide whether $\chi_b(G) = \Delta(G) + 1$. For this section we set $k = \Delta(G) + 1$. Towards this we define a notion of b-*chromatic-core*. Given a graph G, and a positive integer ℓ, a set $C \subseteq V(G)$ is called a b-*chromatic-core of order* ℓ if $\chi_b(G[C]) \geq \ell$. Observe that a minimal set such that $\chi_b(G[C]) \geq \ell$ has size upper bounded by ℓ^2. We start by showing that for the case, when we want to test whether $\chi_b(G) = \Delta(G) + 1 = k$, it is sufficient to find a b-chromatic-core of order k.

▶ **Lemma 4.** *Let G be a graph. Then $\chi_b(G) = k$ if and only if G has a b-chromatic-core of order k. Here $k = \Delta(G) + 1$.*

Proof. For the forward direction let $\chi_b(G) = k$. Then $V(G)$ is a b-chromatic-core of order k.

For the reverse direction assume that G has a b-chromatic-core C of order k. By the definition of b-chromatic-core we have that $\chi_b(G[C]) \geq k$. Since $\chi_b(G[C]) \leq \Delta(G) + 1$ we have that $\chi_b(G[C]) = k$. Let $C_1, \ldots, C_k$ be the partition of C witnessing the fact that $\chi_b(G[C]) = k$. Now we will show that we can extend this partition to the vertex set of G. Let $w_1, \ldots, w_q$ denote the vertices of $V(G) \setminus C$. We iteratively go through the vertices in $V(G) \setminus C$ and try to place it in the already existing partition. Suppose at some stage we have taken care of vertices until say w_i. For w_{i+1} we do as follows. Since the degree of every vertex is upper bounded by Δ we have that there is a partition C_j such that w_{i+1} does not have any neighbor in C_j. We place $w_{i+1} \in C_j$. That is, $C_j := C_j \cup \{w_{i+1}\}$. Observe that the placement preserves the fact that C_j after the addition of the vertex remains independent. This proves the lemma. ◀

Lemma 4 allows us to look for b-chromatic-core of order k for G. Towards this we define the following reduction rule.

▶ **Reduction Rule 1.** *Let (G, k) be an instance to* b-Chromatic Number *and v be a vertex such that every vertex $w \in N[v]$ has degree at most $k - 2$. Then $(G \setminus \{v\}, k)$.*

▶ **Lemma 5 ($\star$).** *Reduction Rule 1 is safe. That is, (G, k) is a* Yes *instance to* b-Chromatic Number *if and only if $(G' = G \setminus \{v\}, k)$ is a* Yes *instance to* b-Chromatic Number.

▶ **Theorem 6.** *Let (G, k) be an instance to* b-Chromatic Number *such that $k = \Delta(G) + 1$. Then there is an algorithm that decides whether $\chi_b(G) = k$, in time $2^{\mathcal{O}(k^2 \log k)} + n^{\mathcal{O}(1)}$.*

Proof. We first apply Reduction Rule 1 exhaustively. Then, we greedily find a maximal set S such that (a) degree of every vertex is equal to $k - 1$ and (b) for any pair of vertices $v, w \in S$, we have that $d(v, w) \geq 4$. We have two cases either $|S| \geq k$ or $|S| < k$. In the first case we can show that $\chi_b(G) = k$ in polynomial time and in the later case we will bound the number of vertices of G by a polynomial function of k.

Case I: $|S| \geq k$. In this case we will show that S and its neighbors form b-chromatic-core of order k for G. Let $C = \{v_1, \ldots, v_k\}$ be an arbitrary subset of size k of S. For every $v_i \in C$ let $W_i = N(v_i)$. Let the vertices of W_i be denoted by $\{w_1^i, \ldots, w_{i-1}^i, w_{i+1}^i, w_k^i\}$. Observe that since for any pair of vertices $v, w \in S$, $d(v, w) \geq 4$, we have that $W_i \cap W_j = \emptyset$ and there are no edges between W_i and W_j for $i \neq j$. Now we make color class I_i, $i \in \{1, \ldots, k\}$, as follows: $I_i = \{w_i^j \mid j \neq i\} \cup \{v_i\}$. Observe that by construction every vertex v_i has neighbor in every color class except I_i and thus $C \cup W$ forms a b-chromatic-core of order k for G. Given the partition of $C \cup W$ we can find a b-chromatic partition of G in polynomial time using the procedure described in Lemma 4.

Case II: $|S| < k$. In this case we claim that for every vertex $v \in V(G) \setminus S$, there exists a vertex $u \in S$ such that $d(u, v) \leq 4$. First of all notice that either the degree of v is $k - 1$ or has a neighbor w with degree $k - 1$. The last assertion follows from the fact that we can not apply Reduction Rule 1 on G. Suppose the degree of v is $k - 1$ then since the greedy algorithm did not pick $v \in S$, we have that there exists a vertex $u \in S$ such that $d(u, v) \leq 3$. In the case when w has degree $k - 1$, we have that there exists a vertex $u \in S$ such that $d(u, w) \leq 3$ and thus $d(u, v) \leq 4$. This implies that every vertex in $V(G)$ can be reached from a vertex in S by a path of length at most 4. Since the maximum degree of this graph is at most $k - 1$, we have that $|V(G)| = 5k^5 = \mathcal{O}(k^5)$. Thus, to test whether $\chi_b(G) = k$ we

guess the b-chromatic-core of order k. We know that there exists one of size at most k^2. Thus, this results in

$$\binom{5k^5}{k^2} \leq 2^{\mathcal{O}(k^2 \log k)}$$

guesses. Given C we can test whether $\chi_b(G[C]) = k$ in time $2^{\mathcal{O}(|C|)} = 2^{\mathcal{O}(k^2)}$ (see the exact algorithm for the mentioned running time). Even the brute force partition into k parts will led to an algorithm with running time $2^{\mathcal{O}(k^2 \log k)}$. This concludes the proof. ◀

5 Exact Algorithm

In this section we show that given a graph G on n vertices as input, we can find the b-chromatic number of G in running time which is single-exponential in n, modulo polynomial factors:

▶ **Theorem 7.** *There is an algorithm which, given a graph G on n vertices as input, finds the b-chromatic number of G in $\mathcal{O}(3^n n^4 \log n)$ time.*

Our algorithm works by checking, for $k = n, (n-1), \ldots, 1$ in this order, whether G has a b-coloring with k colors. It outputs the first (and so, the largest) value of k for which the check returns Yes.

▶ **Theorem 8.** *There is an algorithm which, given a graph G on n vertices and an integer $1 \leq k \leq n$ as input, checks if G has a b-coloring with k colors in $\mathcal{O}(\binom{n}{k} 2^{(n-k)} n^4 \log n)$ time.*

Given Theorem 8, the proof of Theorem 7 is immediate:

Proof of Theorem 7. Since our algorithm for Theorem 7 consists of invoking the algorithm of Theorem 8 once for each $k \in \{1, 2, \ldots, n\}$, we get that the former algorithm runs in time

$$\sum_{0 \leq k \leq n} \mathcal{O}(\binom{n}{k} 2^{(n-k)} n^4 \log n) = \mathcal{O}(n^4 \log n \sum_{0 \leq k \leq n} \binom{n}{k} 2^{(n-k)}) = \mathcal{O}(3^n n^4 \log n),$$

where we get the simplified form from the Binomial Theorem. ◀

Observe that our goal in Theorem 8 is to check whether we can partition the vertex set of G into exactly k non-empty parts $V_1, V_2, \ldots, V_k$ such that

- each set V_i is an independent set in G, and
- for each $1 \leq i \leq k$ there is a vertex $v_i \in V_i$ which has a neighbour in each of the other sets V_j ; $j \neq i$.

Such a partition is a b-coloring of G with k colors, and we call such a set of k vertices $\{v_1, v_2, \ldots, v_k\}$ a *dominator set* for the b-coloring $V_1, V_2, \ldots, V_k$. Our algorithm for Theorem 8 checks, for each vertex subset $\{v_1, v_2, \ldots, v_k\}$ of G of size k, whether there is a b-coloring of G with k colors for which $\{v_1, v_2, \ldots, v_k\}$ is a dominator set.

▶ **Lemma 9.** *Given a graph G on n vertices and a vertex subset $D = \{v_1, v_2, \ldots, v_k\}$ of G of size k, we can find whether graph G has a b-coloring with k colors for which D is a dominator set, in $\mathcal{O}(2^{(n-k)} n^4 \log n)$ time.*

Note that Theorem 8 follows directly from Lemma 9. In the next subsection we describe an algorithm of the kind specified in Lemma 9. We then prove its correctness (subsection 5.2) and show that it runs within the stated time bounds (subsection 5.3).

5.1 The Algorithm

We describe an algorithm which, given a graph G on n vertices and a vertex subset $D = \{v_1, v_2, \ldots, v_k\}$ of G of size k, finds whether graph G has a b-coloring with k colors for which D is a dominator set. Let $\mathcal{I}$ denote the set of all vertex subsets of G which are independent sets in G. Let $V' = V(G) \setminus D$ be the set of all vertices of graph G which are *not* part of the candidate dominator set D. We label the vertices of V' as $V' = \{u_1, u_2, \ldots, u_{n-k}\}$. Let x be an indeterminate.

For each pair of distinct indices $i, j\ ;\ 1 \leq i \neq j \leq k$, let $\mathcal{S}_{ij}$ denote the set of subsets X of V' such that $(X \cup \{v_i\})$ is an independent set in G, and $(X \cup \{v_i, v_j\})$ is *not* an independent set in G:

$$\mathcal{S}_{ij} = \{X \subseteq V'\ ;\ (X \cup \{v_i\}) \in \mathcal{I} \text{ and } \forall 1 \leq j \neq i \leq k : (X \cup \{v_i, v_j\}) \notin \mathcal{I}\} \tag{1}$$

For each index $i\ ;\ 1 \leq i \leq k$, let $\mathcal{T}_i$ denote the intersection, over all $j \neq i$, of the sets $\mathcal{S}_{ij}$:

$$\mathcal{T}_i = \bigcap_{1 \leq j \neq i \leq k} \mathcal{S}_{ij} \tag{2}$$

Note that $\mathcal{T}_i$ is the set of all independent sets in V' which could *potentially* form a color class together with vertex v_i in a b-coloring of interest to us. Indeed, *any* b-coloring of G of the specified kind will consist—apart from the vertices of D—of a pairwise disjoint collection of k independent sets, one from each set $\mathcal{T}_i\ ;\ 1 \leq i \leq k$, such that their union makes up all of V'. Our algorithm looks for such a collection of independent sets, one from each set $\mathcal{T}_i$.

For each vertex $v_i \in D$ we construct the set $\mathcal{T}_i$. We then use $\mathcal{T}_i$ to construct a polynomial $\mathcal{P}_i$ in x for each vertex $v_i \in D$, in the following manner: We initialize $\mathcal{P}_i$ to zero. For each set $X \in \mathcal{T}_i$ we compute the characteristic vector $\chi(X)$ of X with respect to the set V'. We then add the monomial $x^{\chi(X)}$ to the polynomial $\mathcal{P}_i$:

$$\mathcal{P}_i = \sum_{X \in \mathcal{T}_i} x^{\chi(X)} \tag{3}$$

We now compute a sequence of polynomials $\mathcal{Q}_1, \mathcal{Q}_2, \ldots, \mathcal{Q}_k$. We set $\mathcal{Q}_1 = \mathcal{P}_1$. Now for each $2 \leq \ell \leq k$ we compute the polynomial $\mathcal{Q}_\ell$ as follows. We initialize $\mathcal{Q}_\ell$ to zero. For each pair of integers $i, j \geq 0\ ;\ i + j \leq (n - k)$, we

- Compute the polynomials $Q_i = \mathcal{H}_i(\mathcal{Q}_{(\ell-1)})$ and $P_j = \mathcal{H}_j(\mathcal{P}_{(\ell)})$, and their product $R'_{ij} = Q_i \times P_j$;
- Compute the representative polynomial R''_{ij} of R'_{ij}.
- Compute the Hamming projection $R_{ij} = \mathcal{H}_{(i+j)}(R''_{ij})$;
- Set $\mathcal{Q}'_\ell := \mathcal{Q}_\ell + R_{ij}$.
- Set $\mathcal{Q}_\ell$ to be the representative polynomial of $\mathcal{Q}'_\ell$.

▶ Remark. Note that this construction ensures that every nonzero monomial in each polynomial $\mathcal{Q}_i\ ;\ 1 \leq i \leq k$ has the form x^d for some $d \in \mathbb{N}$. That is, no monomial has a coefficient greater than 1 in any of these polynomials.

Our algorithm returns YES if the polynomial $\mathcal{Q}_k$ contains at least one monomial whose degree has Hamming weight $(n - k)$, and NO otherwise. This completes the description of the algorithm.

5.2 Correctness

We now prove that the algorithm of the previous section indeed works exactly as specified in the statement of Lemma 9. We prove this in two parts; first we show that the algorithm is complete (Lemma 10), and then we show that it is sound (Lemma 12).

▶ **Lemma 10.** *If a graph G on n vertices has a b-coloring with k colors for which $D = \{v_1, v_2, \ldots, v_k\} \subseteq V(G)$ is a dominator set, then the algorithm of subsection 5.1 returns* YES *on input (G, D).*

Proof. Suppose graph G has a b-coloring $V_1, V_2, \ldots, V_k$ with k colors for which the set $D = \{v_1, v_2, \ldots, v_k\}$ is a dominator set. Let $V' = V(G) \setminus D$ and for $1 \leq i \leq k$, let $X_i = V_i \setminus \{v_i\}$. Let $\chi(X)$ be the characteristic vector of a subset $X \subseteq V'$ with respect to the set V'. For each $1 \leq i \leq k$ let $m_i = x^{\chi(X_i)}$. It is not difficult to see that for each $1 \leq i \leq k$, the set X_i contributes the monomial m_i to the polynomial $\mathcal{P}_i$ computed by the algorithm.

▶ **Claim 2 ($\star$).** *For each $1 \leq i \leq k$, the polynomial $\mathcal{P}_i$ constructed by the algorithm contains the monomial $m_i = x^{\chi(X_i)}$.*

Our method of computing the polynomials $\mathcal{Q}_1, \mathcal{Q}_2, \ldots, \mathcal{Q}_k$ has two desirable implications: (i) the final polynomial $\mathcal{Q}_k$ computed by the algorithm contains the product of all the m_is as a monomial, and (ii) the degree of this monomial has Hamming weight exactly $(n-k)$. To see this, consider first some properties satisfied by all the polynomials $\mathcal{Q}_\ell$:

▶ **Claim 3.** *For each $1 \leq \ell \leq k$, the following hold:*

1. *Let d be the degree of the monomial $m = \prod_{1 \leq i \leq \ell} m_i$. Then*
   a. *$d \leq 2^{(n-k)}$, and hence d has a binary representation r of width $(n-k)$.*
   b. *Let $S = \biguplus_{1 \leq i \leq \ell} X_i$. Then $\chi(S) = r$.*
2. *The polynomial $\mathcal{Q}_\ell$ contains the monomial m.*

Proof of Claim 3. We prove the claim by induction on ℓ.

Base case: $\ell = 1$.

1. Here $m = m_1 = x^{\chi(X_1)}$.
   (a) Since $X_1 \subseteq V'$, $|V'| = (n-k)$, and the degree d of m_1 is $\chi(X_1)$, we have that $d \leq 2^{(n-k)}$ and that d has—by definition–the binary expansion $r = \chi(X_1)$ of width $(n-k)$.
   (b) The set S is just the subset $X_1 \subseteq V'$, and hence we get directly that $\chi(S) = \chi(X_1) = r$.
2. We get from Claim 2 that the polynomial $\mathcal{Q}_{(\ell-1)} = \mathcal{Q}_1 = \mathcal{P}_1$ contains the monomial $m = m_1 = x^{\chi(X_1)}$.

Induction step: $2 \leq \ell \leq k$

1. Here $m = m_1 m_2 \cdots m_\ell$. Let $m' = m_1 m_2 \cdots m_{(\ell-1)}$.
   (a) Let d be the degree of monomial m, and let d' be the degree of monomial m'. Let $S' = \biguplus_{1 \leq i \leq (\ell-1)} X_i$. By the inductive hypothesis, d' has a binary expansion r' of width $(n-k)$, and $\chi(S') = r'$. Also, $m_\ell = x^{\chi(X_\ell)}$ by definition. Since $m = m' \cdot m_\ell$ we get that $m = x^{d'} \cdot x^{\chi(X_\ell)} = x^{r'} \cdot x^{\chi(X_\ell)} = x^{\chi(S')} \cdot x^{\chi(X_\ell)} = x^{\chi(S') + \chi(X_\ell)}$, where we got the second expression by rewriting d' in binary. Now by assumption the sets S' and X_ℓ are disjoint subsets of V', and hence we get—see Lemma 1—that $d = \chi(S') + \chi(X_\ell) \leq 2^{(n-k)}$, and hence that d has a binary representation r of width $(n-k)$.

Algorithm 1 Algorithm for computing the sets $\mathcal{T}_i$ as bit strings T_i .

```
1: function ComputeTs(G, V', k)
2:     Create bit strings T_1, T_2, ..., T_k of length 2^(n-k) each, with all bits set to zero.
3:     for X ⊆ V' do
4:         for 1 ≤ i ≤ k do
5:             if (X ∪ {v_i}) is an independent set in G then
6:                 inTi ← True
7:                 for 1 ≤ j ≤ k, j ≠ i do
8:                     if (X ∪ {v_i, v_j}) is an independent set in G then
9:                         inTi ← False
10:                if inTi == True then
11:                    T_i[χ(X)] ← 1
12:    return (T_1, T_2, ..., T_k)
```

(b) Here $S = S' \uplus X_\ell$, and so from the above argument and Lemma 1 we get that $r = \chi(S)$.

2. We reuse notation from part (a) above. Let i be the Hamming weight of $\chi(S')$. Note that $i = |S'|$. Let $j = |X_\ell|$; then j is the Hamming weight of $\chi(X_\ell)$. Since S' and X_ℓ are disjoint subsets of the set V', we get from Lemma 1 that $i + j \leq (n - k)$. Therefore i, j is among the pairs of integers over which we iterate during the computation of the polynomial $\mathcal{Q}_\ell$. Let us examine carefully that step in this computation where we consider the pair i, j. Recall that we compute the polynomials $Q_i, P_j, R'_{ij}, R''_{ij}$, and R_{ij} in this step.

From the inductive assumption we get that (i) the polynomial $\mathcal{Q}_{(\ell-1)}$ contains the monomial $m' = m_1 m_2 \cdots m_{(\ell-1)}$, and (ii) $\chi(S')$ is the binary representation of the degree of m', and hence that i is the Hamming weight of the degree of m'. From these we get that the polynomial $Q_i = \mathcal{H}_i(\mathcal{Q}_{(\ell-1)})$ contains the monomial m'. Further, we have shown that the polynomial $\mathcal{P}_\ell$ contains the monomial $m_\ell = x^{\chi(X_\ell)}$, and by definition, j is the Hamming weight of the degree $\chi(X_\ell)$ of m_ℓ. From this we get that the polynomial $P_j = \mathcal{H}_j(\mathcal{P}_\ell)$ contains the monomial m_ℓ. Hence the product $R'_{ij} = Q_i \times P_j$, and thence its representative polynomial R''_{ij} both contain the monomial $m'm_\ell$.

From the inductive assumption we get that $m' = x^{\chi(S')}$, and by definition we have that $m_\ell = x^{\chi(X_\ell)}$. Thus $m'm_\ell = x^{\chi(S')+\chi(X_\ell)}$. Now since S' and X_ℓ are *disjoint* subsets of the set V', we get from Lemma 1 that $\chi(S') + \chi(X_\ell)$ has Hamming weight *exactly* $i + j$. Hence the monomial $m = m'm_\ell$ survives in the Hamming projection $R_{ij} = \mathcal{H}_{(i+j)}(R''_{ij})$. Therefore m is present in the polynomial $\mathcal{Q}_\ell$. ◀

▶ **Corollary 11** (⋆). *Let $m = \prod_{1 \leq \ell \leq k} m_i$. Then the polynomial $\mathcal{Q}_k$ contains m as a monomial, and the Hamming weight of the degree d of m is exactly $(n - k)$.*

Since $\mathcal{Q}_k$ contains the monomial $\prod_{1 \leq \ell \leq k} m_i$ whose degree has Hamming weight $(n - k)$, our algorithm returns Yes on this input. ◀

▶ **Lemma 12** (⋆). *If the algorithm of subsection 5.1 returns* Yes *on input (G, D), then graph G has a b-coloring with k colors for which D is a dominator set.*

Algorithm 2 Algorithm for computing the polynomial $\mathcal{Q}_\ell$ as a bit string S_ℓ.

```
1:  function ComputeQs(S_(ℓ−1), T_ℓ)
                          ▷ S_(ℓ−1), T_ℓ represent the polynomials Q_(ℓ−1), P_ℓ; see the text.
2:      Create bit strings P_1, P_2, ..., P_(n−k) of length 2^(n−k) each, with all bits set to zero.
3:      Create bit strings Q_1, Q_2, ..., Q_(n−k) of length 2^(n−k) each, with all bits set to zero.
4:      Create bit string S_ℓ of length 2^(n−k), with all bits set to zero.
5:      for 1 ≤ i ≤ 2^(n−k) do        ▷ Compute all the projections P_i and Q_i in one pass.
6:          h ← the Hamming weight of i
7:          if T_ℓ[i] == 1 then        ▷ The polynomial P_ℓ contains the monomial x^i
8:              P_h[i] ← 1
9:          if Q_(ℓ−1)[i] == 1 then    ▷ The polynomial Q_ℓ contains the monomial x^i
10:             Q_h[i] ← 1
11:     for 0 ≤ i ≤ (n − k) do
12:         for 0 ≤ j ≤ (n − k − i) do
13:             R''_ij ← FFT-Multiply(Q_i, P_j)        ▷ See explanation in text.
14:             for 1 ≤ p ≤ 2^(n−k) do
15:                 if the Hamming weight of p is i + j then
16:                     if R''_ij[p] == 1 then
17:                         S_ℓ[p] ← 1
18:     return S_ℓ
```

5.3 Running Time Analysis

▶ **Lemma 13.** *The algorithm of subsection 5.1 runs in $\mathcal{O}(2^{(n-k)}n^4 \log n)$ time where n is the number of vertices of the input graph G, and k is the size of the dominator set D.*

Proof. We assume that we are given the adjacency matrix of graph G as input. If required, we relabel the vertices of graph G as $V(G) = \{v_1, v_2, \ldots, v_n\}$ in such a way that the k vertices of the set D appear *last* in this list. In other words, such that $V' = V(G) \setminus D = \{v_1, v_2, \ldots, v_{(n-k)}\}$. This can be done in $\mathcal{O}(n)$ time.

We compute the sets $\mathcal{T}_1, \mathcal{T}_2, \ldots, \mathcal{T}_k$ as *bit strings* $T_1, T_2, \ldots, T_k$, respectively, of length 2^{n-k} each, with the following semantics: the j^{th} bit $T_i[j]$ of bit string T_i is 1 if and only if the subset $X \subseteq V'$ whose characteristic vector $\chi(X)$ with respect to V' satisfies the condition $val(X) = j$ is in the set $\mathcal{T}_i$. We use Algorithm 1 to compute these bit strings.

Creating the empty bit strings (line 2) takes $\mathcal{O}(2^{(n-k)})$ time. The innermost **for** loop (starting at line 7) has $\mathcal{O}(k)$ iterations, each of which takes $\mathcal{O}(n^2)$ time: this is the time we need for a brute-force check for independence of the set $X \cup \{v_i, v_j\}$ using the adjacency matrix of G. The **for** loop at line 7 therefore takes $\mathcal{O}(kn^2)$ time. Assuming that we can access any one bit of a $2^{(n-k)}$-sized bit string in time $\mathcal{O}(\log(2^{(n-k)})) = \mathcal{O}(n)$, line 11 takes $\mathcal{O}(n)$ time. The independence test of line 5 takes, as above,$\mathcal{O}(n^2)$ time. Thus the contents of the **for** loop at line 4 take $\mathcal{O}(n^2) + \mathcal{O}(kn^2) + +\mathcal{O}(n) = \mathcal{O}(kn^2)$ time. From this we get that the loop at line 3 takes $\mathcal{O}(2^{(n-k)} \cdot k \cdot kn^2) = \mathcal{O}(k^2n^2 2^{(n-k)})$ time. The algorithm for creating the bit strings $T_1, T_2, \ldots, T_k$ therefore takes $\mathcal{O}(k^2n^2 2^{(n-k)})$ time.

Observe now that each bit-string T_i serves *also* as a representation of the polynomial $\mathcal{P}_i$. This follows directly from the way we defined the polynomials $\mathcal{P}_i$ from the *sets* $\mathcal{T}_i$. Specifically: monomial x^d is present in polynomial $\mathcal{P}_i$ if and only if the bit $T_i[d]$ is set to 1. Hence we do not need to explicitly compute the polynomials $\mathcal{P}_i$; we just use the bit-strings T_i instead.

Recall that $\mathcal{Q}_1 = \mathcal{P}_1$. For each $2 \leq \ell \leq k$ we compute the polynomial $\mathcal{Q}_\ell$, again as a bit string S_ℓ of length $2^{(n-k)}$, as in Algorithm 2. We first set $S_1 = T_1$. For each $2 \leq \ell \leq k$, in increasing order of ℓ, we invoke the function ComputeQs with the arguments $S_{(\ell-1)}, T_\ell$ to obtain the bit string S_ℓ. The bit string S_k which we obtain at the end of this process is the representation of the polynomial $\mathcal{Q}_\ell$.

The function ComputeQs is mostly self-explanatory, except perhaps for lines 13 to 17. On line 13 we invoke the function **FFT-Multiply** which takes the bit string representations of two polynomials Q_i, P_j, computes the bit string representation of their product R'_{ij} using the Fast Fourier Transform, and returns the (bit string representation of the) representative polynomial R''_{ij} of R'_{ij} (see Lemma 2). In lines 14 to 17 we then add the Hamming projection $R_{ij} = \mathcal{H}_{(i+j)}(R''_{ij})$ to S_ℓ without explicitly computing R_{ij} as a separate bit string.

We now analyze the time taken by one invocation of ComputeQs. Lines 2 to 4 together take time $\mathcal{O}((n-k)2^{(n-k)})$. Lines 6 to 10 can each be executed in time $\mathcal{O}(n-k)$, and so the **for** loop at line 5 takes $\mathcal{O}((n-k)2^{(n-k)})$ time. Line 13 can be performed in $\mathcal{O}(2^{(n-k)}(n-k)\log(n-k))$ time (by Lemma 2), and the **for** loop at line 14 can be executed in $\mathcal{O}((n-k)2^{(n-k)})$ time as well. It follows that the **for** loop at line 11 takes $\mathcal{O}((n-k)^3 2^{(n-k)})$ time. In total, therefore, one invocation of ComputeQs takes $\mathcal{O}(2^{(n-k)}(n-k)^3 \log(n-k))$ time.

Since we invoke ComputeQs $(k-1)$ times, it follows that the complete algorithm runs in $\mathcal{O}(2^{(n-k)}(n-k)^3 k \log(n-k)) = \mathcal{O}(2^{(n-k)} n^4 \log n)$ time. ◀

6 Conclusion

In this paper we studied b-Chromatic Number in the realm of parameterized and exact algorithm and resolved the parameterized complexity of the problem. We would like to conclude with two concrete open problems.

- Is there an algorithm for b-Chromatic Number running in time $2^n n^{\mathcal{O}(1)}$?
- Does there exist an XP algorithm for b-Chromatic Number?
- Is the problem of finding a b-chromatic core of order k FPT when parameterized by k?

References

1 Kenneth Appel and Wolfgang Haken. The solution of the four-color-map problem. *Sci Am*, 237(4):108–121, October 1977.

2 Andreas Björklund, Thore Husfeldt, and Mikko Koivisto. Set partitioning via inclusion-exclusion. *SIAM J. Computing*, 39(2):546–563, 2009.

3 Manfred Cochefert. *Algorithmes Exacts et Exponentiels pour les Problèmes NP-difficiles sur les Graphes et Hypergraphes*. PhD thesis, Université de Lorraine, December 2014.

4 Marek Cygan, Fedor V. Fomin, Łukasz Kowalik, Daniel Lokshtanov, Dániel Marx, Marcin Pilipczuk, Michal Pilipczuk, and Saket Saurabh. *Parameterized Algorithms*. Springer, 2015.

5 Rodney G. Downey and Michael R. Fellows. *Fundamentals of Parameterized Complexity*. Texts in Computer Science. Springer, 2013.

6 Brice Effantin, Nicolas Gastineau, and Olivier Togni. A characterization of b-chromatic and partial grundy numbers by induced subgraphs. *CoRR*, abs/1505.07780, 2015.

7 Frantisek Galcík and Ján Katrenic. A note on approximating the b-chromatic number. *Discrete Applied Mathematics*, 161(7-8):1137–1140, 2013.

8 Frédéric Havet, Cláudia Linhares Sales, and Leonardo Sampaio. b-coloring of tight graphs. *Discrete Applied Mathematics*, 160(18):2709–2715, 2012.

9 Frédéric Havet and Leonardo Sampaio. On the grundy and b-chromatic numbers of a graph. *Algorithmica*, 65(4):885–899, 2013.

10 Robert W. Irving and David Manlove. The b-chromatic number of a graph. *Discrete Applied Mathematics*, 91(1-3):127–141, 1999.

11 Richard M. Karp. Reducibility among combinatorial problems. In *Complexity of computer computations*, pages 85–103. Plenum Press, New York, 1972.

12 Jan Kratochvíl, Zsolt Tuza, and Margit Voigt. On the b-chromatic number of graphs. In Ludek Kucera, editor, *Graph-Theoretic Concepts in Computer Science, 28th International Workshop, WG 2002, Cesky Krumlov, Czech Republic, June 13-15, 2002, Revised Papers*, volume 2573 of *Lecture Notes in Computer Science*, pages 310–320. Springer, 2002.

13 E. L. Lawler. A note on the complexity of the chromatic number problem. *Information Processing Lett.*, 5(3):66–67, 1976.

14 Arnold Schönhage and Volker Strassen. Schnelle Multiplikation großer Zahlen. *Computing*, 7(3-4):281–292, 1971.