Graph Coloring with Rejection*

Leah Epstein† Asaf Levin‡ Gerhard J. Woeginger§

Abstract

We consider the following vertex coloring problem. We are given an undirected graph $G = (V, E)$, where each vertex v is associated with a penalty rejection cost r_v. We need to choose a subset of vertices, V', and to find a proper coloring of the induced subgraph of G over V'. We are interested in both the number of colors used to color the vertices of V', and in the total rejection cost of all other vertices. The goal is to minimize the sum of these two amounts. In this paper we consider both the online and the offline versions of this problem. In the online version, vertices arrive one at a time, revealing the rejection cost of the current vertex and the set of edges connecting it to previously revealed vertices. We also consider the classical online coloring problem on bounded degree graphs and on $(k+1)$-claw free graphs.

1 Introduction

Given an undirected graph $G = (V, E)$, a coloring of G is an assignment of colors to the vertices such that two adjacent vertices are assigned distinct colors. I.e., a coloring is a function $c : V \to Z^+$ such that if $(i, j) \in E$ then $c(i) \neq c(j)$. In the offline COLORING PROBLEM, the goal is to find a coloring of G where the number of used colors, $\max_{i \in V} c(i)$, is minimized. This problem is well-known to be NP-hard for general input graphs (see problem [GT4] in [4]), however it can be solved in polynomial time when the input graph is *perfect* (using the ellipsoid method, see [10]). The minimum number of colors that are necessary in order to color G, is called the *chromatic number of* G and it is denoted by $\chi(G)$. For a coloring c, denote by $n(c, G)$ the number of distinct colors that are used by c.

We define the COLORING WITH REJECTIONS problem, which is a new generalization of the standard coloring problem, as follows. We are given an undirected graph $G = (V, E)$, where each vertex $v \in V$ is associated with a non-negative penalty rejection cost r_v. The goal is to find a subset $V' \subseteq V$, and a coloring c of $G[V']$ that is the induced subgraph of G over V', so as to minimize the sum of the number of used colors (in the coloring of $G[V']$) and the total rejection cost of all the vertices in $V \setminus V'$. I.e., the goal function is $n(c, G[V']) + \sum_{v \in V \setminus V'} r_v$.

*An earlier extended abstract version of this paper appears in Proceedings of the 14th Annual European Symposium on Algorithms (ESA 2006).

†Department of Mathematics, University of Haifa, 31905 Haifa, Israel. lea@math.haifa.ac.il.

‡Department of Statistics, The Hebrew University, 91905 Jerusalem, Israel. levinas@mscc.huji.ac.il.

§Department of Mathematics and Computer Science, TU Eindhoven P.O. Box 513, 5600 MB Eindhoven, The Netherlands. gwoegi@win.tue.nl.

That is, one is penalized for adding a color and for rejecting a vertex; by normalizing we assume that the former penalty is 1.

If we can color (in polynomial time) every induced subgraph of G using a minimum number of colors, then our goal is to find a vertex set V' and a coloring of $G[V']$ that minimizes $\chi(G[V']) + \sum_{v \in V \setminus V'} r_v$.

The use of a penalty function for failing to serve some clients is a common practice in combinatorial optimization problems. For example, this is the motivation behind the definition of the prize-collecting Steiner tree and the prize-collecting traveling salesperson problems (in both problems we pay a penalty for not connecting a vertex, where in the first problem the goal is to construct a tree that spans some vertices, and in the second problem the goal is to construct a cycle over some vertex set). For an earlier work on these problems see for example [5].

Since there is no approximation algorithm for graph coloring of general graphs, unless P=NP, and moreover, there is no competitive online algorithm even for very limited graph classes (such as trees), and as the coloring problem with rejections generalizes the standard coloring problem, we consider both problems, both in the offline and online settings on special classes of graphs, where one can hope to achieve a finite approximation ratio and a finite competitive ratio.

In an online setting, vertices arrive one by one, and we need to deal with an arriving vertex before seeing any future vertices. In the ONLINE COLORING WITH REJECTIONS problem, when a vertex arrives the following information is revealed. Its rejection cost, and the edges that connect this new vertex to all the previously introduced vertices (but no information regarding edges to the future vertices is given at this time). We need to decide whether we would like to accept the new vertex (i.e., to add it to the vertex set V') or to reject it. If the algorithm decides to accept the new vertex, then it needs to color it using one of the existing colors or using a new color (and in this case we say that the algorithm *opens* a new color). In the ONLINE COLORING problem we simply do not reveal the rejection cost of a vertex (as such a notion is undefined for the standard coloring problem, or we can alternatively assume infinite rejection costs for all vertices in this case).

The MAXIMUM WEIGHT k-COLORABLE SUBGRAPH problem is the following related problem. The input to this problem consists of an integer number k and an undirected graph $G = (V, E)$, where each vertex v has a non-negative weight w_v. The goal is to pick a subset $V' \subseteq V$, such that there exists a coloring c of $G[V']$ with k colors, and among all such subsets, the value $\sum_{v \in V'} w_v$ is maximized. Yannakakis and Gavril [12] showed that this problem is NP-hard on split graphs when k is part of the input, and it is polynomially solvable on interval graphs. This was done by formulating the problem as an integer Linear Program on the vertices versus cliques constraint matrix, which is totally unimodular. If all weights are equal, we obtain the MAXIMUM SIZE k-COLORABLE SUBGRAPH problem, that was shown to be polynomial for comparability and co-comparability graphs by Frank [3].

The online coloring problem is well-studied (see Kierstead [8] for a survey). Gyárfás and Lehel [6] showed that for every positive integer k and every online algorithm $\mathcal{A}$, there exists a tree T_k on 2^{k-1} vertices, with maximum degree $k-1$, such that $\mathcal{A}$ must use at least k colors when coloring T_k. They also showed that for trees, the First-Fit algorithm denoted as FF

(the FF algorithm assigns for each vertex the minimum color that can be used to color the vertex) is best possible among all possible online algorithms (in terms of the competitive ratio). Bar-Noy, Motwani and Naor [1] showed that First-Fit is a 2-competitive online algorithm for coloring line graphs, and that any online algorithm has a competitive ratio of at least two (and hence First-Fit is the best possible online algorithm for coloring line graphs).

For an algorithm $\mathcal{A}$, we denote its cost by $\mathcal{A}$ as well. The cost of an optimal offline algorithm that knows the complete sequence of vertices is denoted by OPT. In this paper we consider the absolute competitive ratio and the absolute approximation ratio criteria. For an online algorithm we use the term competitive ratio whereas for an offline algorithm we use the term approximation ratio. The competitive ratio of $\mathcal{A}$ is the infimum $\mathcal{R}$ such that for any input, $\mathcal{A} \leq \mathcal{R} \cdot \mathrm{OPT}$. If the competitive ratio of an online algorithm is at most $\mathcal{C}$ we say that it is $\mathcal{C}$-competitive. The approximation ratio of a polynomial time offline algorithm is defined similarly to be the infimum $\mathcal{R}$ such that for any input, $\mathcal{A} \leq \mathcal{R} \cdot \mathrm{OPT}$. If the approximation ratio of a polynomial time offline algorithm is at most $\mathcal{R}$ we say that it is a $\mathcal{R}$-approximation.

A *perfect graph* is a graph G such that for every induced subgraph G' of G, the chromatic number of G' equals its maximum clique size. A *split graph* is a graph whose vertices can be partitioned into two subsets I and K such that I is an independent set and K induces a complete graph. Note that the complement of a split graph is a split graph, and every split graph is a perfect graph. A $(k+1)$-*claw free graph* is a graph that does not contain an induced subgraph that is a star with $k+1$ leaves, and a *claw-free graph* is a 3-claw free graph. By definition, every claw-free graph is also a $(k+1)$-claw free graph, for any $k > 2$. A *line graph* G can be modeled by the edges of a second graph $H = (V_H, E_H)$ such that the vertex set of G is E_H and there is an edge between two vertices of G if the corresponding edges of H share a common endpoint. Note that both classes of claw-free graphs and line graphs are not contained in the class of perfect graphs (a simple example of a line graph which is also a claw-free graph, but is not a perfect graph, is a cycle over five vertices).

Our results. In Section 2 we consider the offline coloring problem with rejections. Given a graph class $\mathcal{F}$ we show a close relation between the complexity of this problem on $\mathcal{F}$ to the complexity of the maximum weight k-colorable subgraph problem on $\mathcal{F}$. In fact we show that if the class $\mathcal{F}$ is closed under the operation of union with a disjoint finite clique, then one problem is polynomially solvable if and only if the other one is polynomially solvable. Among the sub-classes of the family of perfect graphs discussed in [7], the only graph class that is not closed under this operation is the class of split graphs. For this family it is known that the maximum weight k-colorable subgraph problem is NP-hard, and we show that the coloring problem with rejections on split graphs is NP-hard as well. Then, we turn our attention to approximation algorithms for this problem. We first consider split graphs, we show an approximation algorithm with an additive error guarantee of one from the optimum, and then we show how to derive a polynomial time approximation scheme for the coloring problem with rejections on split graphs. In addition, we present an $O(\log n)$-approximation algorithm for the coloring problem with rejections on perfect graphs that is based on the greedy algorithm. In Section 3 we focus on the classical online coloring problem, and analyze the competitive ratio of the First-Fit algorithm on two graph classes. We show that First-Fit is a $\left(\frac{\Delta+1}{2}\right)$-competitive algorithm for the class of graphs with a maximum vertex degree of at

most Δ, and we show that the First-Fit algorithm colors a $(k+1)$-claw free graph using at most $k\mathrm{OPT} - k + 1$ colors. We also show that any online algorithm cannot perform better on these graph classes. In Section 4 we turn to deal with the online coloring problem with rejections. We first show an online algorithm whose competitive ratio is $\Delta + 2$, for the class of graphs with maximum degree at most Δ. Thus we show that for bounded degree graphs, adding the notion of rejections to the problem makes it harder but still tractable. We then show that there is no online algorithm whose competitive ratio is smaller (in terms of Δ), even if the input graph is a collection of disjoint cliques. Since the class of $(k+1)$-claw free graphs contains all graphs which are collections of cliques, we get that unlike the bounded degree problem, adding rejections to this problem makes it much harder . Among the sub-classes of the family of perfect graphs discussed in [7], the only graph class that does not contain a disjoint union of cliques is the class of split graphs. Therefore, we conclude the paper by showing that even for split graphs there is no online algorithm with a finite competitive ratio.

2 Offline Coloring with Rejections

In this section we study the offline version of the coloring with rejection. We show a close connection between the tractability of this problem on graph classes $\mathcal{F}$ and the tractability of the problem of computing the maximum weight k-colorable subgraph of graphs that belong to $\mathcal{F}$. Note that we do not assume that checking if a given graph G belongs to $\mathcal{F}$ can be done in polynomial time.

Theorem 1 *Given a graph class $\mathcal{F}$ such that $\mathcal{F}$ is closed under the operation of union with a disjoint finite clique, the offline coloring problem with rejections on $\mathcal{F}$ is polynomially solvable if and only if the maximum weight k-colorable subgraph problem on graphs that belong to $\mathcal{F}$ is polynomially solvable.*

Proof. Assume that for $\mathcal{F}$ it is possible to solve the maximum weight k-colorable subgraph problem. Then, given an instance of the offline coloring with rejections $G = (V, E)$ where $G \in \mathcal{F}$ and rejection cost $r_v,\ v \in V$, we apply the following procedure. For each value of k we compute the maximum weight k-colorable subgraph of G where the weight w_v of a vertex v equals its rejection cost, and denote this subgraph by (V_k, E_k). For each value of k, we compute the total cost of k and the total rejection cost of the vertices from $V \setminus V_k$, and we pick the solution whose total cost is minimized. In order to analyze the performance of the resulting algorithm we fix an optimal solution OPT, and consider the iteration in which the algorithm uses the value of k that equals the correct number of colors that OPT uses. By the optimality of (V_k, E_k) to the maximum weight k-colorable subgraph problem, we conclude that the total rejection cost of the vertices in $V \setminus V_k$ is at most the total rejection cost that OPT pays. Therefore, the algorithm returns an optimal solution to the coloring with rejections problem.

Next assume that for the class $\mathcal{F}$, it is possible to compute in polynomial time an optimal solution to the coloring problem with rejection. Let $G = (V, E)$ and $w : V \to R^+$ be an input instance to the maximum weight k-colorable subgraph problem such that $G \in \mathcal{F}$. Let $G' = (V', E')$ be the graph resulting from G by augmenting it with a set of k new vertices

and a clique over them (so G' is a union of G and a clique of size k). Since $G \in \mathcal{F}$ and by the assumption on $\mathcal{F}$, we conclude that $G' \in \mathcal{F}$. Therefore, it is possible to compute in polynomial time the optimal solution to the coloring with rejections problem for every rejection cost function, r. Let $0 \leq \varepsilon \leq \frac{1}{1+\sum_{v \in V} w(v)}$ and define a rejection cost function r as follows: $r(v) = 1$ if $v \in V' \setminus V$ (i.e., if it belongs to the new clique), and otherwise $r(v) = \varepsilon \cdot w(v)$.

Consider an optimal solution OPT to the instance of the coloring problem with rejections. Without loss of generality, OPT does not reject a vertex $v \in V' \setminus V$ (this can be assumed as opening a new color and assigning v to the new color does not increase the total cost). Therefore, the number of colors that OPT uses is at least k. We next argue that OPT uses exactly k colors. To see this claim, assume otherwise that OPT uses at least $k+1$ colors. Then, there is a color in OPT such that each vertex that is assigned to this color is from V, and therefore by definition of ε, the total rejection cost of all the vertices that are assigned to this color is less than 1. Therefore, rejecting all these vertices rather than opening this color results in a solution of a strictly smaller cost, which contradicts the optimality of OPT.

Given that OPT uses exactly k colors, it assigns each one of the clique vertices, $V' \setminus V$, to one of these colors, and also assigns a maximum weight k-colorable subgraph of G to these colors (and the remaining vertices if there are such, are rejected). Since the rejection costs are simply the scaled original weights, OPT solves also the maximum weight k-colorable subgraph of G in polynomial time. ■

We next show that the offline coloring with rejection problem is NP-hard even when the input graph is restricted to be a split graph. Although computing the maximum weight k-colorable subgraph of a split graph is known to be NP-hard (when k is not fixed), the next result does not follow from Theorem 1 as split graphs are not closed under the operation of union with a disjoint clique.

Proposition 2 *The offline coloring problem with rejections is NP-hard in the strong sense even when the input graph is restricted to be a split graph.*

Proof. We use a reduction from the 3-set packing problem that is the special case of set packing where each set in the input has at most three elements (see problem [SP3] in [4]). The input of the 3-set packing problem consists of a collection $\mathcal{C}$ of sets where each set has exactly three elements of a ground set $\mathcal{X}$, and a positive number r. The goal is to test if there is a sub-collection of r mutually disjoint sets in $\mathcal{C}$. Given such an input we construct a split graph input to the coloring with rejections problem as follows. The input graph $G = (V, E)$ is $V = \mathcal{C} \cup \mathcal{X}$ (i.e., there is a vertex for each set in $\mathcal{C}$ and for each element of the ground set), the vertices in $\mathcal{C}$ induce a complete graph in G, and the vertex set $\mathcal{X}$ is an independent set in G, and for each $c \in \mathcal{C}$ and $x \in \mathcal{X}$ there is an edge $(c, x) \in E$ if and only if $x \notin c$. Note that G is clearly a split graph. We next define the rejection costs in the following way. For $c \in \mathcal{C}$ let $r_c = 1 - \frac{2}{|\mathcal{X}|+1}$, and for $x \in \mathcal{X}$ let $r_x = \frac{1}{|\mathcal{X}|+1}$. Note that the resulting instance has polynomial size even if all the numbers are encoded in unary.

Denote by OPT an optimal solution to the coloring with rejections instance. We first argue that without loss of generality, if OPT colors an independent set I of G using a common color, then I consists of exactly one vertex c from $\mathcal{C}$, and the three elements $x_1, x_2, x_3 \in \mathcal{X}$ of c. Note that I cannot contain two vertices of $\mathcal{C}$ since such a pair of vertices are adjacent (and

this contradicts the independence of I). If $I \cap \mathcal{C} = \emptyset$ then I contains at most $|\mathcal{X}|$ vertices with total penalty of at most $\frac{|\mathcal{X}|}{|\mathcal{X}|+1} < 1$. Therefore rejecting all the vertices of I instead of allocating them a color (that costs one) would result in a strictly smaller cost. Therefore, $I \cap \mathcal{C} = \{c\}$. Since c is adjacent to all the vertices beside the three element vertices x_1, x_2, x_3 such that $c = \{x_1, x_2, x_3\}$, then all these three vertices can be colored using the same color as c. However, if the total penalty of the vertices in I is at most 1, we can reject all these vertices instead of allocating them a color (that costs one), and by doing so we do not increase the total cost of the resulting solution. The only case that this cannot be done is the case where the total penalty of the vertices of I is strictly greater than one, but in this case all four vertices must participate in I, i.e., $I = \{c, x_1, x_2, x_3\}$. As a result, we may assume that if OPT colors an independent set I of G using a common color, then I consists of exactly one vertex c from $\mathcal{C}$, and the three elements $x_1, x_2, x_3 \in \mathcal{X}$ of c.

Note that for each such independent set I that OPT colors (with a common color), OPT saves a cost of $\frac{1}{|\mathcal{X}|+1}$ from the total rejection cost of the vertices in I. Therefore, the goal of the solution is to find a maximum size sub-collection of sets in $\mathcal{C}$ that are mutually disjoint. This will clearly solve the 3-set packing instance. ■

We next show that for a split graph there is an approximation algorithm with an additive approximation guarantee. We do not require an input split graph to be introduced together with its realization. However, such a realization is easy to find in polynomial time.

Theorem 3 *If the input to the offline coloring with rejections problem is a split graph, and the optimal solution* OPT *has cost* OPT*, then it is possible to compute in polynomial time a feasible solution* SOL *whose cost is at most* OPT $+ 1$.

Proof. The first step of the algorithm is to compute a decomposition of the input graph G into an independent set $\mathcal{I}$ and a vertex set $\mathcal{K}$ such that the induced subgraph of G over $\mathcal{K}$ is a complete graph. Clearly such a decomposition can be found in polynomial time since G is a split graph.

The algorithm guesses the number of colors that OPT uses. Denote this number by $\mathcal{N}$. The term "guessing" means performing an exhaustive enumeration of all possibilities from the polynomial size set $\{0, 1, 2, \ldots, |V|\}$, and picking the cheapest solution among the resulting solutions (one solution for each possibility). In the analysis it suffices to consider the solution that the algorithm returns in the iteration in which it uses the correct value of $\mathcal{N}$.

SOL uses $\mathcal{N} + 1$ colors. The first color is used to color the independent set $\mathcal{I}$ of G. The other $\mathcal{N}$ colors are used to color the $\mathcal{N}$ most expensive rejection cost vertices from $\mathcal{K}$ (one vertex per each of these $\mathcal{N}$ colors).

The total cost that SOL pays is the sum of $\mathcal{N} + 1$ and the total rejection cost. It suffices to show that the total rejection cost that OPT pays is at least the total rejection cost that SOL pays. However, this is clear as each color in OPT can be used to color at most one vertex from $\mathcal{K}$ and therefore OPT pays the rejection cost of at least $|\mathcal{K}| - \mathcal{N}$ vertices from $\mathcal{K}$ as SOL does. We conclude the proof using the fact that SOL pays the rejection cost for the cheapest vertices in $\mathcal{K}$. ■

We next show how to transform the algorithm of Theorem 3 into a polynomial time approximation scheme.

Corollary 4 *There is a PTAS for approximating the offline coloring problem with rejections on split graphs.*

Proof. Let $\varepsilon > 0$. The algorithm guesses the number of colors that OPT uses. Denote this number by $\mathcal{N}$ (again this is done via exhaustive enumeration). If $\mathcal{N} \geq \frac{1}{\varepsilon}$, then the algorithm applies the method of Theorem 3 to obtain a solution whose cost is OPT + 1. Since OPT uses at least $\frac{1}{\varepsilon}$ colors we conclude that $\text{OPT} + 1 \leq \text{OPT} \cdot (1 + \varepsilon)$. Otherwise, OPT uses a fixed number of colors (that is at most $\frac{1}{\varepsilon}$). In this case we compute the optimal solution to the maximum weight $\mathcal{N}$-colorable subgraph of G (where the weight of each vertex is its rejection cost). This procedure can be done in polynomial time (since split graphs are chordal, and by application of the algorithm of Yannakakis and Gavril [12]). Then, the rejection cost of the solution returned by the algorithm is at most the total rejection cost of OPT. Therefore, if $\mathcal{N} \leq \frac{1}{\varepsilon}$ the algorithm returns an optimal solution. ■

We conclude this section by considering approximation algorithms for perfect graphs. For the maximum weight k-colorable subgraph problem on perfect graphs there is an easy $(1 - \frac{1}{e})$-approximation algorithm that is based on the greedy algorithm for the maximum coverage problem. The MAXIMUM COVERAGE problem is defined as follows: We are given a ground set $\mathcal{E}$ where each element $e \in \mathcal{E}$ has a weight w_e, a collection $\mathcal{S}$ of subsets of $\mathcal{E}$ and an integer number k. The goal is to find a sub-collection $\mathcal{S}'$ of $\mathcal{S}$ of (exactly) k subsets that covers a maximum total weight of the elements of $\mathcal{E}$ (where covering an element means that at least one of the subsets in $\mathcal{S}'$ contains the element). It is known (see [9]) that the *greedy algorithm* is an $(1 - \frac{1}{e})$-approximation algorithm where the greedy algorithm at each iteration adds the set from $\mathcal{S}$ that covers the most total weight of elements that were not covered prior to this iteration, to the collection $\mathcal{S}'$ until there are k subsets in $\mathcal{S}'$. Noting that we can formulate the maximum weight k-colorable subgraph problem as an instance of the maximum coverage problem, by letting $\mathcal{E}$ be the set of vertices of the input graph, and $\mathcal{S}$ be the collection of independent sets of the graph. Note that the greedy algorithm does not use a list of the sets in $\mathcal{S}$, but it only needs to compute a maximum weight independent set in a residual graph (the graph induced by the non-covered vertices), and this can be done in polynomial time for perfect graphs. So we can apply the greedy algorithm for the maximum coverage problem where each iteration is implemented in polynomial time (where the time complexity is polynomial in the size of the input graph).

However, we are not aware of a constant approximation algorithm for the offline coloring with rejections problem when applied to perfect graphs. We next discuss an $O(\log n)$-approximation algorithm for this problem.

Theorem 5 *There is an $O(\log n)$-approximation algorithm for the offline coloring problem with rejections on perfect graphs.*

Proof. We denote the total rejection cost of vertices for which OPT does not pay the rejection cost (this is the sum of rejection costs of the vertices that OPT colors) by R and by $\overline{R}$ the total rejection cost that OPT pays. We also denote by C the number of colors that OPT uses. The PARTIAL COVER problem is defined as follows. We are given a ground set $\mathcal{E}$ where each element $e \in \mathcal{E}$ has a weight w_e, a collection $\mathcal{S}$ of subsets of $\mathcal{E}$ and a number W. The goal is to find a sub-collection $\mathcal{S}'$ of $\mathcal{S}$ of minimum number of subsets that covers

a subset of $\mathcal{E}$ whose total weight is at least W (where covering an element means that at least one of the subsets in $\mathcal{S}'$ contains the element). It is known (see [11]) that the *greedy algorithm* is an $O(\log n)$-approximation algorithm for the partial cover problem where the greedy algorithm at each iteration adds the set from $\mathcal{S}$ that covers the most total weight of elements that were not covered prior to this iteration, to the collection $\mathcal{S}'$ until the total weight of covered elements is at least W. Note that we can formulate the offline coloring with rejections problem as an instance of the partial cover problem by letting $\mathcal{E}$ be the set of vertices of the input graph, $\mathcal{S}$ be the collection of independent sets of the graph and $W = R$. Given this instance note that there is a solution to the partial cover instance that uses at most C subsets and therefore the greedy algorithm returns a solution with at most $O(C \log n)$ subsets, and the total cost of the solution of the offline coloring with rejections is at most $O(C \log n) + \overline{R} \leq O(\log n) \cdot (C + \overline{R}) = O(\log n) \cdot \mathrm{OPT}$. So if the exact value of R is known to us, then the greedy algorithm for the partial cover instance is an $O(\log n)$-approximation algorithm. To overcome the lack of this information, we consider the following algorithm.

Apply the greedy algorithm for the partial cover instance until all elements are covered, adding each of the intermediate solutions created at the end of each iteration of the greedy algorithm, to a solution set denoted by SOL. When the algorithm terminates, return the best solution from SOL. We note that SOL contains the approximated greedy solutions for the partial cover instances for all possible values of W. Therefore, picking the cheapest solution among SOL is superior to the solution created for the correct value of R, and the last solution is an $O(\log n)$-approximation algorithm. ■

3 Online Coloring (Without Rejections)

In this section, we consider the classical online coloring problem. We consider two classes of graphs, $(k+1)$-claw free graphs and bounded degree graphs. We show that the First-Fit algorithm is a best possible online algorithm for both cases.

3.1 $(k+1)$-claw free graphs

In this subsection we prove that for $(k+1)$-claw free graphs, the First-Fit algorithm is a best possible online algorithm. Note that the class of line graphs is a proper sub-class of the class of 3-claw free graphs [2]. Our result extends the result of Bar-Noy, Motwani and Naor [1] for line graphs.

Theorem 6 *The solution returned by the First-Fit algorithm uses at most* $k\mathrm{OPT} - k + 1$ *colors when applied to* $(k+1)$*-claw free graphs. Moreover, no online algorithm can guarantee a smaller number of colors.*

Proof. We first show that the solution returned by First-Fit uses at most $k\mathrm{OPT} - k + 1$ colors. Assume that First-Fit uses $t+1$ colors, and let v be a vertex that is colored by First-Fit using the $t+1$-th color. Then, clearly v has at least t neighbors. Since the input graph is a $(k+1)$-claw free graph, the maximum size independent set in the induced subgraph over the neighbors of v, has size at most k. Therefore, OPT must use at least $\lceil \frac{t}{k} \rceil$ colors to

color the neighbors of v and one additional color to color v. Therefore, if $FF = t+1$, then $\lceil \frac{t}{k} \rceil + 1 \leq \mathrm{OPT}$. Therefore,

$$FF = t+1 \leq k \cdot \left(\left\lceil \frac{t}{k} \right\rceil + 1 \right) - k + 1 \leq k \cdot \mathrm{OPT} - k + 1.$$

It remains to show the lower bound. We fix a value of $k = \ell$, and the cost of the optimal solution, which we denote by $\mathrm{OPT} = n+1$. Our lower bound makes use of the following recursive construction. We define structures of types $1, 2, \ldots$, where each such structure is a graph which has a subset of n designated vertices which are called the "top clique". All other vertices are also arranged in cliques and are associated with levels. A structure of type 1 is simply defined to be a clique over n vertices, all these vertices are defined to be the top clique. These vertices are also called the level 1 clique. We define a structure of type t recursively using structures of types $\ell = 1, 2, \ldots, t$. To obtain a structure of type t, we construct for every $1 \leq i \leq t-1$, n copies of a type i structure. All $n(t-1)$ structures are pairwise disjoint and constructed independently from each other. Denote the structures of type i by $S_{i,1}, \ldots, S_{i,n}$. To combine them all into a type t structure, we construct an additional clique on n vertices, denoted by $v_1, \ldots, v_n$. This will be the top clique of the structure. Vertex v_j is connected to all vertices of the top clique in $S_{i,j}$ for all $1 \leq i \leq t-1$. The level i cliques of the type t structure, for $1 \leq i \leq t-1$ are all level i cliques of all structures of types $1, \ldots, t-1$ (or actually of types $i, \ldots, t-1$, since structures of smaller types do not have level i cliques), which are used in the type t structure. The level t clique of a type t structure is defined to be its top clique.

Next, we define a "superior structure" of type t. This structure is constructed similarly to a regular type t structure, but instead of a top clique which consists of n vertices, we use a top clique with $n+1$ vertices. Furthermore, we use $n+1$ structures of each type $1, \ldots, t-1$ instead of n such structures. Note that structures used in a recursive construction are regular structures and a superior structure is never used as a building block for another structure.

We first discuss an optimal coloring of a superior type t structure, and then we show how to force an online algorithm to use a large number of colors.

We start the coloring from the top level, i.e., from level t of the superior structure. We color this top clique using all the colors $1, \ldots, n+1$. For every $1 \leq i \leq t-1$, each vertex of the top clique is connected to all vertices of the top clique of a (regular) type i structure. Given such a vertex v of the top clique of the type t structure, that is colored with color c, the vertices of the tops cliques that it is connected to all of them, are colored using the colors of $\{1, \ldots, n+1\} - \{c\}$. The other vertices of the structures of smaller types are colored in a similar way, recursively. Note that when coloring the vertices of a clique, the coloring is always proper with respect to the induced graph containing all vertices that were already colored. Thus, we achieve a proper coloring.

We next consider an arbitrary online algorithm. For this algorithm, we construct a graph which consists of disjoint copies of structures defined above, all of types $1, \ldots, \ell - 1$, and superior structures of type ℓ. We show that it is possible to construct such a graph in a way that is uses at least $n\ell + 1$ colors. Since $\mathrm{OPT} = n+1$ for every value of t, we get that the online algorithm uses $\ell(\mathrm{OPT} - 1) + 1 = \ell \mathrm{OPT} - \ell + 1$ colors.

Since our goal is to show that the online algorithm uses at least $n\ell+1$ colors, if at some time during the construction this amount of colors is used, we can stop the construction right away, without introducing any further structures. Therefore, we assume that at each intermediate step, the graph is colored by at most $\kappa = n\ell$ colors.

In the first step, a large amount M_1 of type 1 structures is introduced. Since we are given κ colors, there are at least $\frac{M_1}{\kappa^n}$ such structures having the same set of n colors. We call these colors $1, \ldots, n$. We call structures colored with this given set of colors "properly colored". The structures having a different set of colors are never used again, and remain disconnected from future parts of the graph. The rest of the construction is done inductively. To build structures of type t, we assume that we already built a sufficient amount of properly colored structures of types $1, \ldots, t-1$, where a structure of type i is called "properly colored" if it uses a set of colors whom we call $(i-1)n+1, \ldots, in$, for its top clique. A large amount of type t structures is constructed (as described above) using properly colored structures of smaller types. Note that due to the construction, and due to the fact that all structures of smaller types are colored properly, no new vertex can receive any color among $1, \ldots, (t-1)n$. Therefore, there are less than κ^n subsets of colored which the algorithm can use to color the top cliques of the type t structures. If we built M_t structures, then there is a subset of at least $\frac{M_t}{\kappa^n}$ top cliques which use the same subset of colors. We call these colors $(t-1)n+1, \ldots, tn$ and proceed to the next type. In the last type, the arguments are the same but it is enough to build a single type ℓ structure. Moreover, this structure needs to be superior. Due to the construction and usage of properly colored structures of types $1, \ldots, \ell-1$, the top clique of the superior structure receives $n+1$ new colors, which we call $(\ell-1)n+1, \ldots, n\ell, n\ell+1$ and thus the algorithm uses a total of $n\ell+1$ colors.

Finally, to complete the lower bound construction, we need to show that the resulting graph is indeed $(\ell+1)$-claw free. We say that a vertex is the center vertex in a $k+1$-claw if it is a vertex connected to $k+1$ vertices which form an independent set. Consider a vertex x which belongs to a structure of type t (regular or superior), and is in a level i clique. We would like to find the size of the largest claw in which x is the center vertex. This independent set cannot contain neighbors of the same clique, so we need to count the number of distinct cliques that x is connected to. If x does not belong to a top level structure, i.e., $i < \ell$ the level i is created when a type i structure is constructed. At that time, x is connected to $i-1$ cliques. Later, x will be connected to one additional vertex of the top clique of a type $i+1$ structure. Thus x participates as the center vertex is a $i+1$-claw (we can use one neighbor from each clique, including the level i clique to which x belongs). Since $i+1 \leq \ell$, this does not create a $(\ell+1)$-claw. For a vertex y of the top level, the proof is the same, only no new levels are added, so the vertex may belong to a ℓ-claw, but not to an $\ell+1$-claw. ■

3.2 Bounded degree graphs

In this section we restrict ourselves to input graphs with maximum degree at most Δ, and we prove that the First-Fit algorithm (FF) is a best possible online algorithm for this case.

Theorem 7 *The First-Fit algorithm is a $\left(\frac{\Delta+1}{2}\right)$-competitive algorithm for online coloring of graphs with maximum degree at most Δ, and no online algorithm can guarantee a better competitive ratio on this graph class.*

Proof. If the First-Fit algorithm uses at least two colors, then the input graph contains at least one edge, and therefore OPT uses at least two colors. We next note that when the input graph has maximum degree at most Δ, then the First-Fit algorithm uses at most $\Delta+1$ colors. Therefore, if the input graph has at least one edge, then the competitive ratio of First-Fit is at most $\frac{\Delta+1}{2}$, and otherwise both First-Fit and OPT use exactly one color and in this case the competitive ratio of First-Fit is also at most $\frac{\Delta+1}{2}$.

To see that there is no better online algorithm, we can use the above construction in the proof of Theorem 6, using the value $n = 1$. In this case, each vertex of level $i < \ell$ is connected to one vertex of every level in $1, \ldots, i-1$, and one vertex of level $i+1$. A vertex of level ℓ is connected to one vertex of every level, including ℓ (since the structure of that type is superior). Thus the degree of each vertex is at most ℓ. The number of colors used in the construction presented above is $\ell + 1$, and OPT $= 2$, thus we get a lower bound of $\frac{\Delta+1}{2}$. ■

Remark 8 *Another way to achieve the same lower bound (using a similar construction) for coloring bounded degree graphs using the First-Fit algorithm, is by adapting the lower bound construction in [6]. In that paper, a lower bound of* $\Omega(\log n)$ *on the competitive ratio of online coloring of trees is given. One can stop their construction when a new vertex has a degree of exactly* Δ*. At this step each vertex has a degree of at most* Δ*. Moreover,* OPT *uses only two colors whereas the online algorithm is forced to use at least* $\Delta+1$ *colors. Therefore, no online algorithm can guarantee a competitive ratio smaller than* $\frac{\Delta+1}{2}$*.*

4 Online Coloring with Rejections

The main result of this section is a $(\Delta + 2)$-competitive algorithm for online coloring with rejections, where the input graph has a maximum degree of at most Δ, which is best possible for such graphs.

Theorem 9 *There is a* $(\Delta + 2)$*-competitive algorithm for the online coloring problem with rejections on graphs with a maximum degree of at most* Δ*. Moreover, for any* $\delta > 0$*, there is no online algorithm whose competitive ratio is at most* $\Delta + 2 - \delta$ *for this problem.*

Proof. Our algorithm rejects all arriving vertices as long as the total rejection costs does not exceed one. Then, it opens $\Delta + 1$ colors and uses the First-Fit algorithm to color all future vertices (starting from the vertex in which the total rejection costs exceeds one, and that vertex is the first one to be accepted). If the algorithm reaches the point where it opens $\Delta + 1$ colors, then its total cost is at most $\Delta + 2$ (as the total rejection cost is at most one). In this case the cost of an optimal solution must be at least one, since if OPT uses at least one color, then its cost is at least one, and otherwise it rejects all vertices with total rejection cost at least one (as our algorithm decided to open the colors).

Otherwise, assume that the algorithm rejects all vertices with a total rejection cost of x, then $x \leq 1$, and an optimal solution also rejects all the vertices with total cost of x (if $x = 1$ then there may be a different optimal solution which uses a single color and has the same cost). In this case the competitive ratio is $1 < \Delta + 2$.

Assume that there is an online algorithm whose competitive ratio is at most $\Delta + 2 - \delta$. Let $\varepsilon > 0$ to be defined later. We define a sequence $\{\varepsilon_k\}_{k=1}^{\Delta+1}$ as the (unique) solution for the

following equations $\varepsilon_1 = \varepsilon$ and for all $k = 1, 2, \ldots, \Delta$, the following equation $\varepsilon \cdot \varepsilon_k = \sum_{j=k+1}^{\Delta+1} \varepsilon_j$. The implicit values of the solution are $\varepsilon_i = \frac{\varepsilon^i}{(1+\varepsilon)^{i-1}}$, for $i \leq \Delta$, and $\varepsilon_{\Delta+1} = \frac{\varepsilon^{\Delta+1}}{(1+\varepsilon)^{\Delta-1}}$. Our construction will have two phases. In both phases we will present vertices of disjoint cliques one clique after the other. In the first phase vertex i of the current clique has rejection cost $\varepsilon_{\Delta+2-i}$, and in the second phase vertex i of the current clique has rejection cost $\frac{\varepsilon_{\Delta+2-i}}{\varepsilon}$. We stop presenting vertices in the current clique (and move to the next clique) after the first time that the online algorithm has rejected a vertex from the clique. If the algorithm has opened (at least) $\Delta + 1$ colors we stop the instance immediately. This means that no clique has more than $\Delta + 1$ vertices, since introducing more than $\Delta + 1$ vertices in the clique means that the online algorithm did not reject any of the vertices. However, in that case, it must use $\Delta + 1$ colors to color the clique, and thus the construction terminates at this time. Given an upper bound of $\Delta + 1$ on the size of cliques, we get that the largest degree ever used is at most Δ.

The first phase lasts as long as the total rejection cost of the instance is at most one. Thus the last clique in this phase is presented until the total rejection cost is about to become at least one, and not until the algorithm rejects some vertex. Afterwards, in the next clique that we present, we move to the second phase. If the total rejection cost that the online algorithm pays ever exceeds $3(\Delta + 2)$ at some time, we stop the instance immediately, even if the online algorithm did not open at least $\Delta + 1$ colors. Note that at least one of the two events must happen. In each clique of the second phase, where less than $\Delta + 1$ colors are used by the algorithm, the algorithm increases its rejection cost by at least $\frac{\varepsilon_{\Delta+1}}{\varepsilon}$. Thus within a finite number of cliques either the rejection cost becomes large enough, or at least $\Delta + 1$ colors are used.

If we stop the instance already in the first phase, then the optimal solution rejects all vertices. Otherwise, we get to the second phase, and the solution that we consider as an upper bound on the optimal cost has one color, and it accepts only the last vertex from each clique. Therefore, if we denote the total rejection cost of the optimal solution by R_{opt} and the total rejection cost of the online algorithm by R_{onl}, then we argue that $R_{opt} \leq \varepsilon^2 + \varepsilon R_{onl} \leq \varepsilon + \varepsilon R_{onl}$ is satisfied. To show this, consider first the very last clique of the first phase. Excluding the last vertex of this clique, for which OPT does not need to pay for its rejection cost (since it accepts and colors it), the clique has a total rejection cost of at most ε. As for all preceding cliques, OPT pays the rejection cost only for all vertices except for the last one, whereas the online algorithm pays only for the last vertex, and $\varepsilon \cdot \varepsilon_k = \sum_{j=k+1}^{\Delta+1} \varepsilon_j$ holds.

Assume that the instance stops while we are in the first phase. Let x denote the total rejection cost of all the vertices. Then, the optimal solution of the instance is to reject all the vertices with a total cost of x. The total rejection cost of all the vertices in the last clique is less than 2ε, and therefore the online algorithm has a total rejection cost of at least $\frac{x-2\varepsilon}{1+\varepsilon}$. Since we stop the instance in the first phase the online algorithm has opened $\Delta + 1$ colors, and therefore its total cost is $\Delta + 1 + \frac{x-2\varepsilon}{1+\varepsilon} \geq \Delta + 1 + x - 3\varepsilon$, where the inequality holds as $x \leq 1$. We next note that for all $x \leq 1$ and $\varepsilon \leq \frac{1}{3}$ the following holds $\frac{\Delta+1-3\varepsilon+x}{x} \geq \Delta + 2 - 3\varepsilon$. Therefore, if $\varepsilon \leq \frac{\delta}{3}$, and by the assumption that our online algorithm has a competitive ratio of at most $\Delta + 2 - \delta$, we get that the algorithm does not open the $\Delta + 1$-th color in the first phase. Hence the total rejection cost that the online solution pays for the vertices in the first phase is at least $\frac{1-2\varepsilon}{1+\varepsilon}$.

Recall that R_{onl} is the total rejection cost that the online algorithm pays. If $R_{onl} \geq 3(\Delta+2)$, then $R_{onl} \leq 3(\Delta+2)+\varepsilon$. Assume that $\varepsilon \leq \frac{1}{3(\Delta+2)+1}$, and therefore the competitive ratio of the resulting algorithm is at least $\frac{R_{onl}}{1+R_{opt}} \geq \frac{R_{onl}}{1+\varepsilon+\varepsilon R_{onl}} \geq \frac{R_{onl}}{1+\varepsilon+1}$, where the last inequality holds as $\varepsilon \cdot R_{onl} \leq \varepsilon \cdot (3(\Delta+2)+1) \leq 1$. Therefore, the competitive ratio of the algorithm is at least $\frac{R_{onl}}{3} \geq \frac{3(\Delta+2)}{3} = \Delta+2$, and this contradicts the fact that the competitive ratio of the algorithm is at most $\Delta+2-\delta$. Therefore, the total rejection cost that the online algorithm pays satisfies $\frac{1-2\varepsilon}{1+\varepsilon} \leq R_{onl} \leq 3(\Delta+2)$, and the algorithm pays an additional $\Delta+1$ units of cost for opening at least $\Delta+1$ colors.

Therefore, the total cost paid by the online algorithm is at least $\Delta+1+\frac{1-2\varepsilon}{1+\varepsilon} \geq \Delta+2-3\varepsilon \geq \Delta+2-\frac{\delta}{2}$ (assuming $\varepsilon < \frac{\delta}{6}$), and the total cost paid by OPT is at most $1+R_{opt} \leq 1+\varepsilon+\varepsilon R_{onl} \leq 1+\varepsilon \cdot (7+3\Delta)$. The competitive ratio of the online algorithm is at least $\frac{\Delta+2-\frac{\delta}{2}}{1+\varepsilon \cdot (7+3\Delta)}$. Note that this last quantity should be at most $\Delta+2-\delta$. However, for $\varepsilon < \frac{\delta}{2(7+3\Delta)(\Delta+2-\delta)}$, the above constraint does not hold, and we reach a contradiction. ■

As a corollary, we get that there is no competitive online algorithm for graphs with arbitrarily large vertex degrees. Note that our lower bound of Theorem 9 applies to a disjoint union of cliques (with arbitrary size, so these are not bounded degree graphs).

Corollary 10 *There is no competitive online algorithm for the online coloring problem with rejections, even if the input graph is a disjoint union of cliques.*

Proof. We use the proof of Theorem 9, and note that where the maximum degree can be arbitrarily large the lower bound is arbitrarily large. ■

Since a disjoint union of cliques is a $(k+1)$-claw free graph, for any $k \geq 2$, we get that there is no constant competitive algorithm for the coloring problem with rejections of $(k+1)$-claw free graphs. Thus, despite the similar behavior of the two classes of graphs, $(k+1)$-claw free graphs and bounded degree graphs, with respect to the standard online coloring problem, the online coloring problem with rejections separates these two classes.

The conclusion of Corollary 10 does not apply to split graphs as a union of two (or more) disjoint cliques is not a split graph. We note that the offline problem on split graphs has a polynomial time approximation scheme (and thus is approximable very well), whereas the online problem cannot be tackled. We next present a lower bound for split graphs.

Theorem 11 *There is no competitive online algorithm for the online coloring problem with rejections on split graphs.*

Proof. Assume that there is an online algorithm with competitive ratio ρ (where ρ is a fixed constant). We present a sequence of vertices, each of them has a rejection cost of $\varepsilon = \frac{1}{2\rho}$, with the following properties.

The vertex set K, which is going to be the clique of the split graph, is the set of vertices that the online algorithm accepts. The set I, which will be the independent set of the split graph, is the complement set of vertices, i.e., the rejected vertices of the algorithm. Each new vertex v is adjacent to all the vertices from the clique set K that arrived previously (and therefore we can decide afterwards if v is added to K or to I, and this decision depends upon the action of the algorithm). If the algorithm decides to accept v, it must use a new color for

it (as all the previously accepted vertices are adjacent to v), and then we will add v to K. If the algorithm decides to reject v, then we add v to I (and therefore it will not be adjacent to any future vertex). Using the partition into I and K we conclude that the optimal solution can always color I using a single color, and it can reject all the vertices from K with a total cost $1+|K|\cdot\varepsilon$. The total cost of the online algorithm is $|I|\cdot\varepsilon+|K|$, this holds since all vertices of K are colored, K is a clique, and thus a total of $|K|$ colors are used for this purpose.

First, assume that the algorithm accepts at least $\frac{1}{\varepsilon}+1$ vertices. Its cost is at least $|K|$ and the competitive ratio is at least $\frac{|K|}{1+|K|\varepsilon} > \frac{|K|}{2|K|\varepsilon} = \frac{1}{2\varepsilon} = \rho$, and this contradicts the fact that the competitive ratio of the algorithm is ρ.

Otherwise, the algorithm never accepts more than $\frac{1}{\varepsilon}$ vertices. Therefore, at the time when at least $\frac{4}{\varepsilon^2}$ vertices were introduced to the algorithm, it rejects at least $\frac{3}{\varepsilon^2}$ vertices, and pays a total rejection cost of at least $\frac{3}{\varepsilon}$, whereas the optimal solution pays a total rejection cost of at most $\varepsilon\cdot\left(\frac{1}{\varepsilon}\right)=1$. Therefore, the total cost of the optimal solution is at most two, and the competitive ratio in this case is larger than $\frac{1}{\varepsilon}=2\rho>\rho$, and this contradicts the assumption that the competitive ratio of the algorithm is ρ. ■

References

[1] A. Bar-Noy, R. Motwani, and J. Naor. The greedy algorithm is optimal for on-line edge coloring. *Information Processing Letters*, 44(5):251–253, 1992.

[2] A. Brandstädt, V. B. Le, and J. P. Spinrad. *Graph classes: A survey.* SIAM Monographs on Discrete Mathematics and Applications. SIAM, the Society for Industrial and Applied Mathematics, 1999.

[3] A. Frank. On chain and antichain families of a partially ordered set. *Journal of Combinatorial Theory Series B*, 29:176–184, 1980.

[4] M. R. Garey and D. S. Johnson. *Computers and intractability.* W. H. Freeman and Company, New York, 1979.

[5] M. X. Goemans and D. P. Williamson. A general approximation technique for constrained forest problems. *SIAM J. Comput.*, 24(2):296–317, 1995.

[6] A. Gyárfás and J. Lehel. On-line and first-fit colorings of graphs. *Journal of Graph Theory*, 12:217–227, 1988.

[7] D. S. Johnson. The NP-completeness column: an ongoing guide. *Journal of Algorithms*, 6(3):434–451, 1985.

[8] H. A. Kierstead. Coloring graphs on-line. In A. Fiat and Gerhard J. Woeginger, editors, *Online Algorithms - The State of the Art*, chapter 13, pages 281–305. Springer-Verlag, Berlin, 1998.

[9] G. L. Nemhauser and L. Wolsey. Maximizing submodular set functions: formulations and analysis of algorithms. In *Studies of graphs and discrete programming*, pages 279–301. North-Holland, Amsterdam, 1972.

[10] A. Schrijver. *Combinatorial optimization polyhedra and efficiency.* Springer-Verlag, 2003.

[11] P. Slavík. Improved performance for the greedy algorithm for partial cover. *Information Processing Letters*, 64(5):251–254, 1997.

[12] M. Yannakakis and F. Gavril. The maximum k-colorable subgraph problem for chordal graphs. *Information Processing Letters*, 24(2):133–137, 1987.