Online Graph Coloring with Predictions

Antonios Antoniadis, Hajo Broersma, and Yang Meng(✉)

Faculty of Electrical Engineering Mathematics and Computer Science, University of Twente, P. O. Box 217, 7500 AE Enschede, The Netherlands
{a.antoniadis,h.j.broersma,y.meng}@utwente.nl

Abstract. We introduce learning augmented algorithms to the online graph coloring problem. Although the simple greedy algorithm FirstFit is known to perform poorly in the worst case, we are able to establish a relationship between the structure of any input graph G that is revealed online and the number of colors that FirstFit uses for G. Based on this relationship, we propose an online coloring algorithm FirstFitPredictions that extends FirstFit while making use of machine learned predictions. We show that FirstFitPredictions is both *consistent* and *smooth*. Moreover, we develop a novel framework for combining online algorithms at runtime specifically for the online graph coloring problem. Finally, we show how this framework can be used to robustify FirstFitPredictions by combining it with any classical online coloring algorithm (that disregards the predictions).

Keywords: learning augmented algorithms · online algorithms · online graph coloring · first fit

1 Introduction

Before we will properly define the concepts we use throughout this paper in Sect. 1.1, let us start with a less formal introduction to the subject, together with some background and motivation.

Graph coloring is a central topic within graph theory that finds its origin in the notorious Four Color Problem, dating back to 1852. Since then graph coloring has developed into a mature research field with numerous application areas, ranging from scheduling [20], and memory allocation [2] to robotics [7]. In the online version of the problem, the vertices of a (usually unknown) graph arrive online one by one, together with the adjacencies to the already present vertices. Upon the arrival of each vertex v, a color has to be irrevocably assigned to v. The goal is to obtain a proper coloring, that is a coloring in which no two adjacent vertices have the same color. The challenge is to design a coloring strategy which keeps the total number of assigned colors as small as possible.

In this paper, we introduce the online graph coloring problem to learning augmented algorithms. We assume that alongside the arrival of each vertex, the algorithm also obtains a prediction $P(v)$ (of unknown quality) on the color

A. Basu et al. (Eds.): ISCO 2024, LNCS 14594, pp. 289–302, 2024.
https://doi.org/10.1007/978-3-031-60924-4_22

that should be assigned to v. These predictions may be used by an algorithm to obtain a coloring which uses fewer colors, if the predictions turn out to be relatively accurate. At the same time, the algorithm should maintain a worst-case guarantee to safeguard against the case, in which the predictions are inaccurate.

Graph coloring is notoriously hard, already in the *offline setting*, where the whole input graph is known in advance. Let $\chi(G)$ denote the *chromatic number* of a graph G, that is the minimum number of distinct colors needed to obtain a proper coloring of G. A straightforward observation is that $\chi(G)$ is greater than or equal to $\omega(G)$, which is defined as the cardinality of a maximum clique in G. However, there even exist triangle-free graphs (so with no cliques of cardinality 3) with an arbitrarily large chromatic number. In case $\chi(H) = \omega(H)$ for every induced subgraph H of a graph G, then G is called a *perfect graph*. Perfect graphs are known to be $\chi(G)$-colorable in polynomial-time via semidefinite programming [8]. An interesting special case of perfect graphs is the class of bipartite graphs. These graphs admit a proper coloring using only 2 colors, and in the offline setting such a 2-coloring can be computed in linear time, for example via breadth first search. However, in general it is an NP-complete problem to decide whether a given graph admits a proper coloring using k colors, for any fixed $k \geq 3$ [15]. With respect to approximation algorithms, there is a polynomial-time algorithm using at most $O(n(\log\log n)^2/(\log n)^3) \cdot \chi(G)$ [11] colors for a graph on n vertices, and it is NP-hard to approximate the chromatic number within a factor $n^{1-\epsilon}$ for all $\epsilon > 0$ [24].

The problem becomes even more challenging in the *online setting* where, as mentioned, vertices arrive online one by one and an algorithm has to irrevocably assign a color to each vertex upon its arrival – while only having knowledge of the subgraph revealed so far. Any online algorithm may require at least $(2n/(\log n)^2)\chi(G)$ colors in the worst case [12], where n is the number of vertices of the input graph – and this is true even for bipartite graphs (so with chromatic number 2). Restricting the input graphs even further, for instance to P_6-free bipartite graphs (containing no path on six vertices as an induced subgraph), no online algorithm can guarantee a coloring with constantly many colors [10]. In such cases, bounding the number of colors used from above by a function of the chromatic number is not feasible. To this end, a line of research has focused on developing so-called *online-competitive algorithms*. Applied to any graph G, such online algorithms are guaranteed to produce a proper coloring, the number of colors of which is bounded from above by a function of the number of colors used by the *best possible online algorithm* for G. As an example, in the previously mentioned setting of P_6-free bipartite graphs, there exists an online algorithm that uses at most twice as many colors as the best possible online algorithm [5,21]. A good source for more information regarding online graph coloring is the following book chapter due to Kierstead [16]. We will come back to this in Sect. 3.3, where we review and utilize several known results, including more recent work.

Probably the conceptually simplest online algorithm for graph coloring is the greedy algorithm, which is known as FirstFit. Suppose we have a total order

over the set of available colors. Then upon arrival of each vertex, FirstFit assigns to it the smallest color according to that order, among the ones that maintain a proper coloring. This algorithm has been extensively analyzed in the literature, also for particular graph classes [10,13]. Although FirstFit performs well for many practically relevant inputs, for example for interval graphs and for the complement of bipartite graphs, it can be very sensitive to the order in which the vertices of G are revealed. For instance, in a popular example where G is a complete balanced bipartite graph $K_{n,n}$ minus a perfect matching, there is a specific permutation on the arrival of the vertices of G for which FirstFit requires n colors (whereas $\chi(G) = 2$) [14].

That FirstFit performs well in some practical scenarios, can be attributed to the fact that real-world graph coloring instances rarely resemble worst-case inputs. It is often the case that either the structure of the input graph or the permutation in which the vertices are revealed can be exploited by a heuristic or a machine-learning approach in order to yield reasonably good colorings, despite the inherent worst-case difficulty of the problem. However, and not too surprisingly, such approaches tend to come without a worst-case guarantee. In the (hopefully rare) cases where the input diverges substantially from the expected structure, the resulting coloring could be arbitrarily poor.

In this paper, we design an algorithm that incorporates predictions of unknown quality obtained by such a machine-learned approach. It produces a relatively good coloring in case the predictions turn out to be accurate, while at the same time providing a worst-case guarantee comparable to the best classical online algorithm that does not make use of the predictions. Our work falls within the context of *learning augmented algorithms*.

Learning augmented algorithms is a relatively new and very active field. The main goal is to develop algorithms combining the respective advantages of machine-learning approaches and classical worst case algorithm analysis. A plethora of online problems have already been investigated through the learning augmented algorithm lens, including for example, caching [19], facility location [6], ski-rental [23], or various scheduling problems [17,23] to name just a few. To the best of our knowledge, learning augmented algorithms have not been studied for graph coloring problems to date. For a more extensive discussion on learning augmented algorithms, we refer the interested reader to a recent survey [22].

A common approach to developing learning augmented algorithms is to design an algorithm that attempts to follow the predictions, in some sense. At the same time, this algorithm should be robustified by appropriately combining it with a classical algorithm that disregards the predictions. At a high level, this is our approach for graph coloring as well. However, the nature of the problem poses several novel challenges. First of all, already assigned colors may significantly restrict the choice of colors for the next and future assignments of the algorithm. This is in contrast to settings where an algorithm can, at some cost, move to any arbitrary configuration, for instance, in problems with an underlying metric. The fact that it is not possible for an algorithm to move to any

possible configuration also rules out a robustification approach by combining algorithms in an experts-like setting. See [1] for more information. Secondly, existing online algorithms for online graph coloring do not possess a particular monotonicity property that tends to be a crucial ingredient in robustifying algorithms for other problems. In particular, it is possible that running an algorithm only on the graph induced by a suffix of the input permutation requires significantly more colors than running the same algorithm on the graph of the complete input permutation. This further complicates the robustification, since one can not simply use a classical algorithm as a fall back option upon recognizing that the predictions are of insufficient quality.

1.1 Preliminaries

In this section, we formally define the problem setting and its associated prediction model. We start with the concepts related to (offline) graph coloring.

Definition 1 (Graph coloring [4]). *A* k-vertex coloring*, or simply a* k-coloring*, of a graph* G *is a mapping* $\phi : V(G) \to S$*, where* S *is a set of* k *colors. A* k*-coloring is proper if no two adjacent vertices are assigned the same color. A graph is* k-colorable *if it admits a proper* k*-coloring. The minimum* k *for which a graph* G *is* k*-colorable is called its* chromatic number*, denoted by* $\chi(G)$*. An* optimal coloring *of* G *is a proper* $\chi(G)$*-coloring.*

Online graph coloring describes the setting, in which the vertices of G arrive one by one in an online fashion. Upon arrival, the vertices have to be irrevocably and properly colored.

Definition 2 (Online graph coloring [21]). *An* online graph (G, π) *is a graph* G *together with a permutation* $\pi = v_1, v_2, \ldots, v_n$ *of* $V(G)$*. An* online coloring algorithm *takes an online graph* (G, π) *as input and produces a proper coloring of* $V(G)$*, where the color of a vertex* v_i *is chosen from a* universe $\mathcal{U}$ *of available colors and the choice depends only on the subgraph of* G *induced by* $\{v_1, v_2, \ldots, v_i\}$ *and the colors assigned to* $v_1, v_2, \ldots, v_{i-1}$*, for* $1 \leq i \leq n$*.*

Throughout the paper, and unless otherwise specified, algorithm refers to a *deterministic* algorithm. We next define the notions of *competitiveness, online competitiveness* and *competitive ratio*, which are used to evaluate the performance of algorithms for online graph coloring.

Definition 3 (Competitiveness [10], online competitiveness [9] and competitive ratio [18]). *Let* $AOL(G)$ *be the set of all online coloring algorithms for a graph* G *and let* $\Pi(G)$ *be the set of all permutations of* $V(G)$*. For an algorithm* $A \in AOL(G)$ *and a permutation* $\pi \in \Pi(G)$*, the number of colors used by* A *when* $V(G)$ *gets revealed according to* π *is denoted by* $\chi_A(G, \pi)$*. The* A*-chromatic number of* G *is the largest number of colors used by the online algorithm* A *for the graph* G*, denoted by* $\chi_A(G)$*. That is,*

$$\chi_A(G) = \max_{\pi \in \Pi(G)} \chi_A(G, \pi).$$

For a graph G, the online chromatic number $\chi_{OL}(G)$ is the minimum number of colors used for G, over all algorithms of $AOL(G)$. That is,

$$\chi_{OL}(G) = \min_{A \in AOL(G)} \chi_A(G).$$

Let $\mathcal{G}$ be a family of graphs and $AOL(\mathcal{G})$ be the set of online algorithms for $\mathcal{G}$. For some $A \in AOL(\mathcal{G})$, if there exists a function such that $\chi_A(G) \leq f(\chi(G))$, (resp. $\chi_A(G) \leq f(\chi_{OL}(G))$) holds for every $G \in \mathcal{G}$, then A is competitive *(resp.* online competitive*) on $\mathcal{G}$. Furthermore, the* competitive ratio *of an algorithm $A \in AOL(\mathcal{G})$ over a class of graphs $\mathcal{G}$ is the maximum of $\frac{\chi_A(G)}{\chi(G)}$ for all $G \in \mathcal{G}$.*

We note that the notion of competitiveness used here follows the literature on online graph coloring and contrasts the definition commonly used for other online problems, where an algorithm is said to be competitive if it attains a constant competitive ratio. We complete this section by presenting the basic definitions associated with the setting, involving predictions on the colors.

Predictions and Prediction Error. Here we assume that alongside the disclosure of each vertex v, the algorithm also obtains a prediction $P : V(G) \to \mathcal{U}$ on the color of v, where $\mathcal{U}$ is the set of available colors. These predictions are aimed at obtaining a reasonable coloring. They may stem from a machine-learning approach based on past inputs or training data, or from a simple heuristic known to perform well in practice. The quality of the obtained predictions is measured by means of a *prediction error*. This *prediction error* is defined naturally to be the (smallest) number of vertices that obtained wrong predictions.

Definition 4 (Prediction error). *Given an online graph (G, π), let $\mathcal{O}(G)$ be the set of all optimal colorings of G, where $O \in \mathcal{O}$ assigns color $O(v) \in \mathcal{U}$ to vertex v. Then the* prediction error *for online graph (G, π) is given by*

$$\eta(G) = \min_{O \in \mathcal{O}(G)} \Sigma_{v \in V(G)} |\{P(v)\} \setminus \{O(v)\}|.$$

In the following, we drop the dependence on G when the underlying online graph is clear from the context. We use the notation (G, π, P) to refer to an *online graph with predictions*, where G is the underlying graph, π the permutation in which V is revealed, and P the set of associated predictions.

Following the literature, we say that an algorithm is α-*consistent* if it attains a competitive ratio of α in the case that the predictions are perfect ($\eta = 0$), and *robust* if it independently of the prediction error obtains a competitive ratio within a constant factor of that of the best classical online algorithm. Furthermore, we say that an algorithm is *smooth* if its competitive ratio degrades at a rate that is at most linear in the prediction error. Note that the notion of robustness also extends to the case where the best known classical online algorithm A

is "only" online competitive. In this scenario, any algorithm that is guaranteed to use at most a constant number of colors more than what A uses is *robust*.

We note that one can trivially obtain an optimal coloring when the predictions are perfect (in other words when $\eta = 0$) by just coloring each vertex v with color $P(v)$ upon arrival. This already is a 1-consistent algorithm. However, when the obtained predictions are only slightly off, this algorithm may not even be a valid algorithm for online graph coloring. Indeed, consider the case where only one vertex v receives a wrong prediction $P(v)$ but is adjacent to a vertex u with $P(u) = P(v)$.

1.2 Our Contribution

Our first contribution lies in establishing a relationship between the structure of an online graph (G, π) and the amount of colors used by FirstFit for G. We emphasize that this result is independent of predictions and might be of broader interest. More specifically, in Sect. 2 we show that if FirstFit uses x colors for G, then there exists a set $V' \subseteq V$ of vertices of size $|V'| = x + q$, for some $0 \leq q \leq x-2$, such that V' can be partitioned into $q+1$ non-trivial subsets, each of which is a *clique* in G (that is, each subset consists of at least two vertices which are all pairwise adjacent in G). Our result is even constructive, i.e., we present such an algorithm for finding V' and a partitioning.

Theorem 1. *Let (G, π) be an online graph for which* FirstFit *uses x colors. Then, there exists a set $V' \subseteq V$ of size $|V'| = x+q$ with $0 \leq q \leq x-2$, such that V' can be partitioned into $q+1$ non-trivial subsets of vertices, each of which is a clique.*

Our second contribution is to develop a 1-consistent and smooth algorithm for online graph coloring with predictions, called FirstFitPredictions in Subsect. 3.1. Consider the setting where the algorithm, upon the reveal of a vertex v also obtains a prediction on the color that v should be colored with in an optimal coloring. We give an algorithm that employs FirstFit with a distinct color palette for each subgraph induced by the set of vertices that obtained the same prediction. By carefully utilizing the aforementioned structural result, we are able to associate the number of colors used by the algorithm with the number of wrong predictions obtained. More specifically, we are able to show that the number of colors used by the algorithm differs from that of an optimal coloring by at most the number of wrong predictions (implying that if the predictions are perfect, the algorithm actually recovers an optimal coloring, even though the quality of the predictions is not a priori known to the algorithm).

Theorem 2. *Assume that* FirstFitPredictions *uses $x(G)$ colors for some online graph with predictions (G, π, P) whose chromatic number is unknown to the algorithm, then*

$$x(G) \leq \eta(G) + \chi(G).$$

Our third contribution is a novel framework for combining different online graph coloring algorithms in Subsect. 3.2. Earlier frameworks developed for other online problems do not seem to carry over to the online graph coloring problem. Our framework allows us to robustify our algorithm by combining it with a classical algorithm that disregards the predictions. We show that the number of colors used by the combination of the two algorithms is within a factor of 2 from that of the best performing of the two on this input instance. This implies that this combination is a 2-consistent, smooth and robust algorithm.

Although in this paper we only use the framework to combine two algorithms, we prove the result for combining any number t of online algorithms (at a loss of t in the competitive ratio). Given an online graph (G, π) and an online coloring algorithm A, let $A(G)$ denote the number of colors A uses for G.

Theorem 3. *Let $A_1, A_2, \ldots, A_t$ be online graph coloring algorithms that may or may not make use of the predictions. Then, there exists a meta-algorithm A that combines $A_1, A_2, \ldots, A_t$, such that for any online graph with predictions (G, π, P) it holds that*

$$A(G) \leq t \cdot \min_{1 \leq i \leq t} A_i(G).$$

The generality of our result allows us to obtain learning augmented algorithms for online graph coloring for different graph classes in Subsect. 3.3.

2 A Structural Theorem About FirstFit

This section is devoted to proving Theorem 1, which establishes a relationship between the number of colors used by FirstFit for an online graph (G, π) and a partition of a subset of V into cliques. As mentioned, our proof is constructive and implies an efficient algorithm for finding such a partition. In the proof we assume that FirstFit uses $x \geq 2$ colors which are ordered as $c_0 < c_1 < \ldots < c_{x-1}$. We use $N(u)$ to denote the neighbors of a vertex u, i.e., the set of vertices that are adjacent to u.

Proof of Theorem 1. Let $t_{x-1} \in V$ be a vertex for which FirstFit uses color c_{x-1}. By the definition of FirstFit vertex t_{x-1} must be adjacent to vertices $t_0, t_1, \ldots, t_{x-2}$, such that t_i is colored with color $c(t_i) = c_i$ for all $i = 0, \ldots, x-2$. Let $S = \bigcup_{i=0}^{x-1} \{t_i\}$ and let $N^-(u) = \{w \in N(u) \cap S \mid c(w) < c(u)\}$ be its *neighborhood of smaller-colored vertices within* S, $\forall\ u \in S$.

Note that the set of vertices $V_0' = \{t_i \in S \mid N^-(t_i) = \{t_0, t_1, \ldots, t_{i-1}\}\}$ is a clique. Also note that $|V_0'| \geq 2$, since $\{t_0, t_{x-1}\} \subseteq V_0'$. If $V_0' = S$, then the theorem directly follows for $q = 0$. So, assume for the remainder of this proof that $V_0' \neq S$. Let $S' = S \setminus V_0' = \{u_1, u_2, \ldots, u_\ell\}$, in which the vertices of S' are ordered by increasing color. We describe an algorithm for partitioning S' into q subsets, with the property that each of these subsets of S' together with one distinct vertex from $V \setminus S$ is a clique of size at least two. Note that this implies the theorem since $1 \leq |S'| \leq x - 2$ and thus $1 \leq q \leq x - 2$.

For every $h \in \{1, ..., \ell\}$ let $\alpha(u_h) \in S'$ be a vertex of maximal color with $c(\alpha(u_h)) < c(u_h)$ such that $\alpha(u_h)$ is not adjacent to u_h, thus $\alpha(u_h) \notin N^-(u_h)$. And let $\beta(u_h) \in V \setminus S$ be a vertex with $c(\beta(u_h)) = c(\alpha(u_h))$ that is adjacent to u_h. Note that such vertices must exist: if $\alpha(u_h)$ did not exist, then u_h would be contained in V'_0; and if $\beta(u_h)$ did not exist, FirstFit would have assigned a smaller color, namely $c(\alpha(u_h))$, to u_h.

The algorithm proceeds iteratively over the vertices of S' in order of increasing color. For each vertex $u_h \in S'$, if $\beta(u_h) \notin V'_j$ for all j with $1 \leq j < h$, then a new clique $V'_h = \{u_h, \beta(u_h)\}$ is created. Else, there exists a j with $1 \leq j < h$ such that $\beta(u_h) \in V'_j$. In this case, set $V'_j := V'_j \cup \{u_h\}$, in other words, add u_h to V'_j. We will show that this creates a larger clique. But first note that by the definition of the algorithm each different $\beta(u_h)$ is added to exactly one such set V'_j, and each such set V'_j contains exactly one specific $\beta(u_h)$. Thus, the output is indeed a partition of $S' \bigcup \cup_h \{\beta(u_h)\}$.

It remains to argue that upon termination, each set V'_j is a clique. We will prove the stronger statement that throughout the execution of the algorithm each set V'_j is a clique, and consists of $\beta(u_j)$ and a subset of vertices of S' with a color strictly larger than $c(\beta(u_j)) = c(\alpha(u_j))$. This invariant clearly holds upon creation of such a set V'_j, since it is created as a clique $\{u_j, \beta(u_j)\}$ and $c(u_j) > c(\beta(u_j))$. Assume that the invariant holds up to some iteration $h-1$. Now consider iteration h during which u_h gets added to V'_j. By the definition of the algorithm $\beta(u_h) \in V'_j$, and thus $\beta(u_h) = \beta(u_j)$. And by the definition of $\beta(u_h)$, it is adjacent to u_h. It remains to show that u_h is adjacent to all the vertices in $V'_j \setminus \{\beta(u_j)\}$, and that $c(u_h) > c(\beta(u_j))$. The latter directly follows from our ordering and the fact that $c(u_h) > c(u_j) > c(\beta(u_j))$. For the former, recall that $\alpha(u_h)$ is defined as the vertex of S' of maximal color that is not adjacent to u_h. In other words, $N^-(u_h)$ contains a vertex of each color strictly between $c(\alpha(u_h))$ and $c(u_h)$, and therefore each vertex of S' with such a color is adjacent to u_h. By the induction hypothesis V'_j only contains such vertices (except for vertex $\beta(u_j) = \beta(u_h)$ whose adjacency to u_h has already been argued).

3 Algorithmic Results

In this section, we focus on deriving and analysing learning augmented algorithms for online graph coloring. We introduce a consistent and smooth algorithm in Subsect. 3.1, and show how it can be robustified in Subsect. 3.2. Finally, in Subsect. 3.3 we argue how it can be used to obtain learning augmented algorithms for online coloring of specific graph classes.

Due to space constraints most of the remaining proofs are deferred to the full version of the paper.

3.1 FirstFitPredictions (FFP)

Throughout this section we assume that the predicted colors are chosen from the set $\{c_0, c_1, c_2, \ldots\}$. Given an online graph with predictions (G, π, P), upon

revealing of a new vertex v with prediction $P(v) = c_i$, the algorithm FirstFit-Predictions (FFP for short) employs FirstFit with a distinct color palette associated with c_i. We use $C(i) = \{c_i^0 = c_i, c_i^1, c_i^2, \ldots\}$ to denote the *color palette* associated with color c_i, implying a natural ordering according to the superscripts. Keeping the colors of each such palette distinct enables us to associate the total number of colors used by FFP to the total prediction error.

> FirstFitPredictions: When a new vertex v is revealed with prediction $P(v) = c_i$, assign to v the smallest-superscript eligible color $c \in C(i)$.

FFP implies a partition of the vertices of G (and the subgraphs of G induced by the vertices that have been revealed so far) based on their color palettes.

Definition 5. *We say that a vertex v belongs to color palette $C(i)$, if it was assigned a color $c \in C(i)$ by* FFP *(or equivalently, it received the prediction c_i). We use $G_i = (V_i, E_i)$ to denote the subgraph of G induced by the set of vertices of color palette $C(i)$.*

Note that an alternative, equivalent description of FFP is that it colors each induced subgraph G_i of G using FirstFit with color palette $C(i)$. Also note that it is without loss of generality to assume that the color palettes are distinct: every time a specific color is predicted for the first time, one can "rename" it to a new, unused color (if required) and define the corresponding color palette accordingly. Finally, note that FFP does not require any information on the chromatic number $\chi(G)$ of the graph G. We can now relate the number of colors used by FFP in each color palette to the number of prediction errors within that color palette.

Lemma 1. *Fix an optimal coloring $O \in \mathcal{O}(G)$, let $\eta_i(G)$ be the number of vertices v of color palette $C(i)$ for which $O(v) \neq P(v)$, and let $x_i(G)$ be the number of distinct colors used by* FFP *for vertices of $C(i)$. Then*

$$x_i(G) \leq \eta_i(G) + 1.$$

Lemma 1 plays a central role in the proof of Theorem 2. Theorem 2 shows that FFP never uses more than $\eta(G) + \chi(G)$ colors for an online graph G with predictions. The next result shows that there exist graphs for which this amount of colors may indeed be used.

Lemma 2. *For every integer $k \geq 2$, there exists an online graph with predictions (G, π, P) and $\chi(G) = k$ for which* FFP *uses $x(G) = \eta(G) + k$ colors.*

Theorem 2 and Lemma 2 directly imply the following result.

Theorem 4. *The competitive ratio of* FFP *is $1 + \frac{\eta(G)}{\chi(G)}$.*

Theorem 4 directly implies the 1-consistency and smoothness of algorithm FFP: if $\eta(G) = 0$, then FFP produces a $\chi(G)$-coloring and is therefore optimal; furthermore the number of assigned colors by FFP grows linearly with the prediction error. Nevertheless, FFP is not a robust algorithm. Indeed, for example, suppose we are given a bipartite online graph of order n with predictions (G, π, P) such that $\eta(G) = \Theta(n)$. Then FFP would use $\Theta(n)$ colors. But there exist classical online algorithms (without predictions) [5,16,18] that can color any bipartite graph with $O(\log n)$ colors. In the next section, we present how FFP can be robustified by elegantly combining it with a classical algorithm.

3.2 RobustFirstFitPredictions(RFFP)

A common approach for robustifying a consistent algorithm is to appropriately combine it with a classical algorithm that disregards the predictions. A first such attempt for robustifying FFP would be to switch to some classical online coloring algorithm A, once the number of colors used by FFP becomes larger than some predetermined threshold T. Recall that $\chi_A(G)$ denotes the number of colors that algorithm A uses for an online graph with predictions (G, π, P), where $\pi = v_1, v_2, v_3, \ldots, v_n$. Furthermore, assume that FFP would for the first time use $T+1$ colors upon arrival of some vertex v_i. Then, by switching to a classical online coloring algorithm A (using the same set of colors) for the restriction of π to the *suffix-subgraph* G' induced by $\{v_i, v_{i+1}, \ldots, v_n\}$, the total number of colors used by the combined algorithm would be at most $T+\chi_A(G')$. Similarly to the deterministic combination result for problems with an underlying metric [1], this would already give a robust algorithm, if we can assume that A is weakly monotone in the following sense.

Definition 6. *Let $A(G, \pi)$ be the number of colors A uses for (G, π), where $\pi = v_1, v_2, \ldots, v_n$. Let $\pi(i)$ be the suffix $v_i, v_{i+1}, \ldots, v_n$ of π, and let $(G(i), \pi(i))$ be the online subgraph of (G, π) induced by the vertices in $\pi(i)$. We say that A is* weakly monotone *(resp.* monotone*) if for any i, $A(G(i), \pi(i)) \leq c \cdot A(G, \pi)$ for some constant c (resp. for $c = 1$).*

Unfortunately, and perhaps somewhat surprisingly, we are not aware of any weakly monotone online graph coloring algorithm with a non-trivial guarantee on the number of used colors. (In the full version we present instances showing that both FirstFit and BiColorMax (see [5]) are not weakly monotone, even on specific classes of bipartite graphs which are known to admit online competitive coloring algorithms.)

We are able to circumvent this issue related to the non-monotonicity by reserving a distinct color palette for each algorithm during the combination. This, however, has the side-effect that after switching to an algorithm A in some round r, it is possible that upon arrival of a vertex v the algorithm A itself does not increase its number of used colors (by using a color that was already employed before round r), but the combined algorithm does. This rules out an expert-setting approach for combining the algorithms (see [1,3] for more information),

but we are still able to bound the total number of colors used by the combined algorithm. More specifically, we are able to combine FFP with $t-1$ classical algorithms and obtain an algorithm RobustFirstFitPredictions *(*RFFP *for short)* which uses a number of colors bounded from above by the expression in Theorem 3.

Proof of Theorem 3. For $1 \leq i \leq n$, let $G(i)$ denote the (online) graph induced by $\{v_1, v_2, \ldots, v_i\}$. For any algorithm B, let $c(B, i)$ be the color that B assigns to vertex v_i.

In the following, we restrict each of the algorithms $A_1, A_2, \ldots, A_t$ to use its own distinct color palette, where we assume a total ordering of the colors within each palette. Meta-algorithm A is defined as the algorithm that upon arrival of vertex v_i (and the accompanying prediction $P(v_i)$) colors it with color $c(\mathrm{ALG}_i, i)$, where $\mathrm{ALG}_i \in \{A_1, A_2, \ldots, A_t\}$ is an algorithm realizing $\min_{1 \leq j \leq t} A_j(G(i))$.

Note that since each A_i produces a proper coloring and uses a distinct color palette, the resulting coloring after applying A is proper as well. It remains to argue about the number of colors it would use for G.

Let ALG be ALG_n, and for any algorithm A_i, let $d(i)$ be the maximal index such that $A_i(G(d(i))) \leq \mathrm{ALG}(G(d(i)))$. Note that by the definition of A, it will not use any color from A_i's color palette on vertices $v_{i+1}, v_{i+2}, \ldots, v_n$. Thus, A uses at most $A_i(G(d(i))$ colors from A_i's color palette. Overall, this gives

$$A(G) \leq \sum_{i=1}^{t} A_i(G(d(i))).$$

By the definition of $d(i)$, the above is at most

$$\sum_{i=1}^{t} \mathrm{ALG}(G(d(i))).$$

Since an online algorithm cannot alter any assigned color, $\mathrm{ALG}(G(j)) \leq \mathrm{ALG}(G(j+1))$ for all $j \in \{1, 2, \ldots, n-1\}$. This implies $\mathrm{ALG}(G(d(i))) \leq \mathrm{ALG}(G)$, which concludes the proof.

In the full version, we show that the result is tight for this meta-algorithm A.

In the previous result, we have been combining deterministic algorithms. However, Theorem 3 easily extends to randomized algorithms as well, assuming that one can simulate the execution of all algorithms simultaneously. The following directly follows from Theorem 3, Jensen's inequality and the concavity of the minimum function.

Corollary 1. *Let $A_1, A_2, \ldots, A_t$ be randomized algorithms for online graph coloring that may or may not make use of the predictions, and assume that one can simulate the execution of these algorithms simultaneously. Then, there exists a*

(randomized) meta-algorithm A that combines $A_1, A_2, \ldots, A_t$ and for any online graph (G, π)

$$\mathbb{E}(A(G)) \leq t \cdot \min_{1 \leq i \leq t} \mathbb{E}(A_i(G)).$$

Theorem 3 implies that we can combine FFP with a c-competitive classical algorithm (perhaps on a specific class of input graphs) and obtain a $2\min\{1 + \frac{\eta(G)}{\chi(G)}, c\}$-competitive algorithm. Assume that the (learning augmented) algorithm has knowledge that the input graph belongs to a specific class of graphs such that (i) all graphs of this class have chromatic number k and (ii) there exists a classical online algorithm that is online competitive on this class, with function $f(\cdot)$. In that specific case, a slight adaptation in the proof of Theorem 3 even gives us a 1-consistent algorithm that is at the same time robust.

Corollary 2. *For some fixed k, assume that the algorithm* FFP *is aware that the input graph belongs to a class $\mathcal{C}$ of graphs such that $\chi(G) = k$ for all $G \in \mathcal{C}$. Moreover, assume that a classical online algorithm A_1 is known for all graphs of class $\mathcal{C}$. Then, there exists a meta-algorithm A' that combines* FFP *with A_1, and for any online graph $(G, \pi, P) \in \mathcal{C}$,*

$$A'(G) \leq 3\min\{\mathrm{FFP}(G), A_1(G)\} \text{ if } \eta(G) > 0, \text{ and}$$
$$A'(G) = k \text{ otherwise.}$$

3.3 Results for Specific Classes of Graphs

Given that the input graph belongs to a specific graph class (and this is known to the algorithm a priori), we can obtain more refined results. In this section, we review some interesting cases for which classical (deterministic) online algorithms are known.

Theorem 5. *There exist (different) algorithms for online coloring bipartite graphs with predictions obtaining a competitive ratio of 1 if $\eta(G) = 0$, and $3\min\{\frac{\eta(G)}{2} + 1, X\}$ otherwise, where X is $\Theta(\log n)$ for general bipartite graphs, $\chi_{OL}(G) - \frac{1}{2}$ for bipartite P_6-free graphs, $2\chi_{OL}(G) - \frac{1}{2}$ for bipartite P_7-free graphs, $\frac{3(\chi_{OL}(G)+1)^2}{2}$ for bipartite P_8-free graphs, and $3(\chi_{OL}(G)+1)^2$ for bipartite P_9-free graphs.*

Theorem 6. *There exist (different) algorithms for online coloring chordal graphs, d-inductive graphs, graphs of treewidth d and disk graphs with predictions obtaining a competitive ratio of 1 if $\eta(G) = 0$, and $2\min\{\frac{\eta(G)}{\chi(G)} + 1, X\}$ otherwise, where $X = O(\log n)$ is the competitive ratio of the respective classical online algorithm.*

Theorem 7. *There exist (different) algorithms for online coloring I_s-free graphs and $K_{1,t}$-free graphs for $t \geq 3$ with predictions obtaining a competitive ratio of 1 if $\eta(G) = 0$, and $2\min\{\frac{\eta(G)}{\chi(G)} + 1, X\}$ otherwise, where X is $t - 1$ for $K_{1,t}$-free graphs, and $\frac{s}{2}$ for I_s-free graphs.*

4 Discussion

It would be interesting to investigate whether a learning augmented algorithm with a consistency better than 2 is possible, when the chromatic number of the input graph is not known to the algorithm. Furthermore, all presented algorithms are smooth and the number of used colors grows linearly with the prediction error. It is an open question whether there exist any learning augmented algorithms which achieves the same consistency, but with a better dependence on the prediction error.

References

1. Antoniadis, A., Coester, C., Eliás, M., Polak, A., Simon, B.: Online metric algorithms with untrusted predictions. In: ICML 2020, vol. 119, pp. 345–355 (2020)
2. Barenboim, L., Drucker, R., Zatulovsky, O., Levi, E.: Memory allocation for neural networks using graph coloring. In: ICDCN 2022, pp. 232–233 (2022)
3. Blum, A., Burch, C.: On-line learning and the metrical task system problem. Mach. Learn. **39**(1), 35–58 (2000)
4. Bondy, J.A., Murty, U.S.R.: Graph Theory with Applications. Macmillan Education, UK (1976)
5. Broersma, H., Capponi, A., Paulusma, D.: A new algorithm for on-line coloring bipartite graphs. SIAM J. Discret. Math. **22**(1), 72–91 (2008)
6. Cohen, I.R., Panigrahi, D.: A general framework for learning-augmented online allocation. CoRR, abs/2305.18861 (2023)
7. Demange, M., Ekim, T., de Werra, D.: A tutorial on the use of graph coloring for some problems in robotics. Eur. J. Oper. Res. **192**(1), 41–55 (2009)
8. Grötschel, M., Lovász, L., Schrijver, A.: Polynomial algorithms for perfect graphs. In: Topics on Perfect Graphs. North-Holland Mathematics Studies, vol. 88, pp. 325–356 (1984)
9. Gyárfás, A., Király, Z., Lehel, J.: Online competitive coloring algorithms. Technical report, TR-9703-1 (1997)
10. Gyárfás, A., Lehel, J.: On-line and first fit colorings of graphs. J. Graph Theory **12**(2), 217–227 (1988)
11. Halldórsson, M.M.: A still better performance guarantee for approximate graph coloring. Inf. Process. Lett. **45**(1), 19–23 (1993)
12. Halldórsson, M., Szegedy, M.: Lower bounds for on-line graph coloring. Theor. Comput. Sci. **130**(1), 163–174 (1994)
13. Irani, S.: Coloring inductive graphs on-line. Algorithmica **11**(1), 53–72 (1994)
14. Johnson, D.S.: Worst case behavior of graph coloring algorithms. In: Proceedings of SEICCGTC, pp. 513–527. Utilitas Mathematica (1974)
15. Karp, R.M.: Reducibility among combinatorial problems. In: Miller, R.E., Thatcher, J.W., Bohlinger, J.D. (eds.) Complexity of Computer Computations. The IBM Research Symposia Series, pp. 85–103. Springer, Cham (1972). https://doi.org/10.1007/978-1-4684-2001-2_9
16. Kierstead, H.A.: Coloring graphs on-line. In: Fiat, A., Woeginger, G.J. (eds.) Online Algorithms. LNCS, vol. 1442, pp. 281–305. Springer, Heidelberg (1996). https://doi.org/10.1007/BFb0029574
17. Lattanzi, S., Lavastida, T., Moseley, B., Vassilvitskii, S.: Online scheduling via learned weights. In: Proceedings of SODA 2020, pp. 1859–1877 (2020)

18. Lovász, L., Saks, M.E., Trotter, W.T.: An on-line graph coloring algorithm with sublinear performance ratio. Discret. Math. **75**(1–3), 319–325 (1989)
19. Lykouris, T., Vassilvitskii, S.: Competitive caching with machine learned advice. In: ICML 2018, vol. 80, pp. 3302–3311 (2018)
20. Marx, D.: Graph coloring problems and their application in scheduling. Periodica Polytech. Electr. Eng. **48**, 11–16 (2004)
21. Micek, P., Wiechert, V.: An on-line competitive algorithm for coloring bipartite graphs without long induced paths. Algorithmica **77**(4), 1060–1070 (2017)
22. Mitzenmacher, M., Vassilvitskii, S.: Algorithms with predictions. Commun. ACM **65**(7), 33–35 (2022)
23. Purohit, M., Svitkina, Z., Kumar, R.: Improving online algorithms via ML predictions. In: NeurIPS 2018, pp. 9684–9693 (2018)
24. Zuckerman, D.: Linear degree extractors and the inapproximability of max clique and chromatic number. Theory Comput. **3**(1), 103–128 (2007)