Online Multi-Coloring with Advice[☆]

Marie G. Christ[b], Lene M. Favrholdt[b], Kim S. Larsen[b,1]

[a]*Department of Mathematics and Computer Science, University of Southern Denmark, Campusvej 55, DK-5230 Odense M, Denmark*

Keywords: Advice complexity, Graph multi-coloring, Frequency allocation, Online algorithms, Competitive analysis

---

[☆]Supported in part by the Danish Council for Independent Research and the Villum Foundation. An extended abstract appeared in the Twelfth Workshop on Approximation and Online Algorithms (WAOA), Lecture Notes in Computer Science, volume 8952. Springer, 2014.

Email addresses: `christm@imada.sdu.dk` (Marie G. Christ), `lenem@imada.sdu.dk` (Lene M. Favrholdt), `kslarsen@imada.sdu.dk` (Kim S. Larsen)

[1]Corresponding author.

Online Multi-Coloring with Advice[☆]

Marie G. Christ[b], Lene M. Favrholdt[b], Kim S. Larsen[b,1]

[b] *Department of Mathematics and Computer Science, University of Southern Denmark, Campusvej 55, DK-5230 Odense M, Denmark*

Abstract

We consider the problem of online graph multi-coloring with advice. Multi-coloring is often used to model frequency allocation in cellular networks. We give several nearly tight upper and lower bounds for the most standard topologies of cellular networks, paths and hexagonal graphs. For the path, negative results trivially carry over to bipartite graphs, and our positive results are also valid for bipartite graphs. For hexagonal graphs, negative results trivially carry over to 3-colorable graphs, and most of our positive results do as well. The advice given represents information that is likely to be available, studying for instance the data from earlier similar periods of time.

1. Introduction

We consider the problem of graph *multi-coloring*, where each node may receive multiple requests. Whenever a node is requested, a color must be assigned to the node, and this color must be different from any color previously assigned to that node or to any of its neighbors. The goal is to use as few colors as possible. In the *online* version, the requests arrive one by one, and each request must be

[☆]Supported in part by the Danish Council for Independent Research and the Villum Foundation. An extended abstract appeared in the Twelfth Workshop on Approximation and Online Algorithms (WAOA), Lecture Notes in Computer Science, volume 8952. Springer, 2014.

Email addresses: `christm@imada.sdu.dk` (Marie G. Christ), `lenem@imada.sdu.dk` (Lene M. Favrholdt), `kslarsen@imada.sdu.dk` (Kim S. Larsen)

[1]Corresponding author.

colored without any information about possible future requests. The underlying graph is known to the online algorithm in advance.

The multi-coloring problem is motivated by *frequency allocation* in cellular networks. These networks are formed by a number of base transceiver stations, each of which covers what is referred to as a cell. Due to possible interference, neighboring cells cannot use the same frequencies. In this paper, we use classic terminology and refer to these cells as nodes in a graph where nodes are connected by an edge if they correspond to neighboring cells in the network. Frequencies can then be modeled as colors. Multiple requests for frequencies can occur in one cell and overall bandwidth is a critical resource.

Two basic models dominate in the discussion of cellular networks, the highway and the city model. The former is modeled by linear cellular networks, corresponding to paths, and the latter by hexagonal graphs. We consider the problem of multi-coloring such graphs.

For practical applications, the assumption that absolutely nothing is known about the future is often unrealistic and hence, many problems have been studied in various semi-online settings. The notion of *advice* offers a quantitative and problem independent way of relaxing the online constraint. In this model, the online algorithm is provided with partial knowledge about the future by an oracle writing bits of advice to an advice tape. The *advice complexity* of the algorithm is the maximum total number of advice bits read from the tape, as a function of the length of the input sequence.

The advice supplied to the algorithms studied in this paper is essentially (an approximation of) the maximum number of requests given to any clique in the graph. For the application of frequency allocation, it does not seem unrealistic that this information could be derived from previous data.

2. Preliminaries

2.1. The problem

In this section we define the problem of online multi-coloring formally. A graph, $G = (V, E)$, is given from the beginning, and the input sequence consists of requests to nodes in G. Each time a node, v, is requested, the algorithm must assign a color to v. The color must be different from all colors previously given to v and its neighbors, and it must be chosen without any knowledge about possible future requests. The colors are positive integers, and the goal is to minimize the largest color used.

Multi-coloring has been studied in various settings. Some papers study a variation of the problem where requests may be *cancelled*, freeing their colors for future requests. In this paper, we only very briefly consider cancellations. Another variation is to allow color changes (reassignment of frequencies). This is called *recoloring*. An algorithm is *d-recoloring* if, in the process of treating a request, it may recolor up to a distance d away from the node of the request.

2.2. Notation

Throughout, we let n denote the number of requests in a given input sequence. We let log denote $\log_2$, the logarithm with base 2.

If A is a multi-coloring algorithm, we let $\mathrm{A}(I)$ denote the number of colors used by A on the input sequence I. When I is clear from the context, we simply write A instead of $\mathrm{A}(I)$.

For any input sequence, I, we let $\textsc{Opt}(I)$ (or simply $\textsc{Opt}$) denote the number of colors used by an optimal offline algorithm when given the requests in I.

2.3. Competitive ratio

The quality of an online algorithm is often given in terms of the competitive ratio, defined by Sleator and Tarjan [35] and named by Karlin et al. [27]. An online multi-coloring algorithm is c-*competitive* if there exists a constant α such that for all input sequences I, $\mathrm{A}(I) \leq c\,\textsc{Opt}(I) + \alpha$. The (asymptotic) *competitive ratio* of A is the infimum over all such c. Results that can be established using $\alpha = 0$ are referred to as *strict* (or absolute).

We use the term *strictly* 1-*competitive* to denote that an algorithm is as good as an optimal offline algorithm, and *optimal* to mean that no better online algorithm exists under the given conditions.

2.4. Advice complexity

We use the advice model by Hromkovič, Královič, and Královič [24], where the online algorithm has access to an infinite advice tape, written by an offline oracle with infinite computation power. In other words, the online algorithm can ask for the answer to any question and read the answer from the tape. Competitiveness is defined and measured as usual, and the advice complexity is simply the number of bits read from the tape. More precisely, for each given sequence length, n, the *advice complexity* of an algorithm is the maximum number of advice bits read by the algorithm, over all sequences of length at most n.

As the advice tape is infinite, we need to specify how many bits of advice the algorithm should read and if this knowledge is not implicitly available, it has to be given explicitly in the advice string. For instance, if we want $\textsc{Opt}$ as advice, then we cannot merely read $\lceil \log(\textsc{Opt}+1) \rceil$ bits, since this would require knowing something about the value of $\textsc{Opt}$.

One can use a *self-delimiting encoding* as introduced by Elias [21]. We use the variant by Boyar et al. [11], defined as follows: The value of a non-negative integer X is encoded by a bit sequence, partitioned into three consecutive parts. The last part is X written in binary. The middle part gives the number of bits in the last part, written in binary. The first part gives the number of bits in the middle part, written in unary and terminated with a zero. These three parts require $\lceil \log(\lceil \log(X+1) \rceil + 1) \rceil + 1$, $\lceil \log(\lceil \log(X+1) \rceil + 1) \rceil$, and $\lceil \log(X+1) \rceil$

bits, respectively, adding a low order term to the number of bits of information required by an algorithm.

We define $enc(x)$ to be the minimum number of bits necessary to encode a number x, and note that the encoding above is a (good) upper bound on $enc(x)$.

3. Related Work

3.1. Relaxing the online constraint

Considering advice, formalized under the notion of *advice complexity*, is a recent trend in the analysis of online algorithms. In the first model proposed by Dobrev, Královič, and Pardubská [19], the algorithm asks a question for each request and gets answers of possibly varying lengths. One problem with this approach is that the algorithm may get information from receiving answers of length 0. Thus, the total number of bits received may not be a correct measure of the amount of information passed from the oracle to the algorithm. This problem was addressed by Emek et al. [22], introducing a model where the algorithm receives a fixed number of advice bits per request. However, in this model it is not possible to receive a sublinear number of advice bits. Thus, in the present paper, we use the "advice-on-tape" model by Hromkovič, Královič, and Královič [24]: Before the first request arrives, the oracle writes the advice to an advice tape, and these advice bits can be read by the algorithm at any point during its computations.

The possibly most famous online problem of paging, where no deterministic online algorithm is better than k-competitive on a cache size of k, can be solved optimally with one bit of advice per request, saying whether to keep the requested page in cache until its next request; see Dobrev, Královič, Pardubská [19] and Böckenhauer et al. [6].

Many other problems have been studied in an advice setting, including disjoint path allocation by Barhum et al. [2], and job shop scheduling by Böckenhauer et al. [6], as well as k-server by Böckenhauer et al. [5], knapsack by Böckenhauer et al. [7], set cover by Komm, Královič, and Mömke [29], metrical task systems by Emek et al. [22], and buffer management by Dorrigiv, He, and Zeh [20].

Also graph coloring has been considered, but in a very different online setting, where the graph itself is not available from the beginning. Instead, the nodes are revealed one by one. Results have been obtained for paths by Forisek, Keller, and Steinová [23], bipartite graphs by Bianchi et al. [3], and 3-colorable graphs by Seibert, Sprock, and Unger [34]. Bianchi et al. [4] consider a coloring problem with restrictions going beyond the immediate neighbors.

Theoretically, results on advice complexity give some information in the direction of the hardness stemming from the problem being online, relaying information concerning how much we need to know about the future to perform better. For practical applications, though many problems must be addressed without knowing in which order requests arrive, quite often something is known about the sequence of requests as a whole.

This realization that input is not arbitrary (uniformly random, for instance) is not new, and work focusing on locality of reference in input data has tried to capture this. Early work includes access graph results, initiated by Borodin et al. [8]. A recent account for related work in continuation of the initial access graph results was given by Boyar, Gupta, and Larsen [10]. More distributional models have also been developed by Albers, Favrholdt, and Giel [1].

Boyar and Larsen [12] introduced an entirely different concept of accommodating sequences, further developed by Boyar, Larsen, and Nielsen [13] and Boyar et al. [9]. The idea is that for many problems requiring resources, there is a close connection between the resources available and the resources required for an optimal offline algorithm, as when capacity of transportation systems are matched with expected demand. This leans itself closely up against many of the results that we report here, where the advice needed to do better is often some information regarding the resources required by an optimal offline algorithm.

There are interesting connections between advice and randomization and sometimes results on advice complexity can be used to obtain efficient randomized algorithms; see Böckenhauer et al. [6], Komm and Královič [28], and Böckenhauer et al. [7].

3.2. Multi-coloring

For multi-coloring a *path*, the algorithm 4-Bucket is proven $\frac{4}{3}$-competitive by Chrobak and Sgall [18], and this is established as optimal by Chan et al. [15]. Even with 0-recoloring allowed (that is, colors at the requested node may be changed), 4-Bucket has been proved optimal by Christ, Favrholdt, and Larsen [16]. Furthermore, if 1-recoloring is allowed, the algorithm GreedyOpt is strictly 1-competitive [16], even if cancellations may occur. If recoloring is not allowed and cancellations may occur, there is a lower bound of $\frac{11}{7}$ by Chrobak and Sgall [18] and a $\frac{5}{3}$-competitive algorithm, Borrow, by Chan et al. [15].

For multi-coloring *bipartite graphs*, the optimal asymptotic competitive ratio has been determined to lie between $\frac{10}{7} \approx 1.428$ and $\frac{18-\sqrt{5}}{11} \approx 1.433$ by Chrobak, Jez, and Sgall [17].

The following results are all valid for the setting where cancellations may occur. Chan et al. [14] have shown that, for *hexagonal graphs*, no online algorithm can be better than $\frac{3}{2}$-competitive or have a better strict competitive ratio than 2. They also gave an algorithm, Hybrid, with an asymptotic competitive ratio of approximately 1.9 on hexagonal graphs. On k-colorable graphs, it is strictly $\frac{k+1}{2}$-competitive, and hence, it has an optimal strict competitive ratio on hexagonal graphs. Recoloring was studied by Janssen et al. [26]: No d-recoloring algorithm for hexagonal graphs has an asymptotic competitive ratio better than $1 + \frac{1}{4(d+1)}$. For $d = 0$, the lower bound was improved to $\frac{9}{7}$. The best known 1-recoloring algorithm for hexagonal graphs is $\frac{33}{24}$-competitive, which was established by Witkowski and Zerovnik [37]. Sparl and Zerovnik [36] gave a $\frac{4}{3}$-competitive 2-recoloring algorithm.

	Ratio	Lower	Type	Thm	Upper	Type	Thm
Paths	1	$\log n - 2$	s	1	$\log n + O(\log\log n)$	s	3
	$1 + \frac{1}{2^b}$	$b - 2$	a	2	$b + 1 + O(\log\log n)$	s	4
Hexagonal	1				$(n+1)\lceil \log n \rceil$	s	7
	$< \frac{5}{4}$	$\Omega(n)$	a	6			
	$\frac{4}{3}$				$n + 2\lvert V \rvert$	a	9
	$< \frac{3}{2}$	$\left\lfloor \frac{n-1}{3} \right\rfloor$	s	5			
	$\frac{3}{2}$				$\log n + O(\log\log n)$	a	8

Table 1: Overview of our results. We mark the ratios that are strict by "s" and the ones that are asymptotic by "a". For each bound, we indicate the number of the theorem proving the result. For readability, many of the bounds stated are weaker than those proven in the paper.

For the *offline* problem of multi-coloring hexagonal graphs, no polynomial time algorithm can obtain an absolute approximation ratio better than $\frac{4}{3}$ unless P = NP; see McDiarmid and Reed [30] and the overview by Narayanan and Shende [32, 33]. A $\frac{3}{2}$-approximation algorithm called the Fixed Preference Allocation algorithm was given by Janssen, Kilakos, and Marcotte [25]. Subsequently, Narayanan [31] simplified the strategy and noted that the algorithm can be converted to a 1-recoloring online algorithm. McDiarmid and Reed [30] introduced a slightly more involved algorithm with an approximation ratio of $\frac{4}{3}$.

4. Our Contribution

An overview of our results is given in Table 1.

For multi-coloring paths, we adapt a strictly 1-competitive 1-recoloring algorithm by Christ, Favrholdt, and Larsen [16] to use $\log n + O(\log\log n)$ bits of advice instead of recoloring. The advice given is the value of Opt, which is simply the maximum number of requests to any pair of neighboring nodes. We then show how a higher competitive ratio can be traded for fewer advice bits. Both results are complemented by lower bounds that are tight up to low order terms, with the latter holding even for the asymptotic competitive ratio. All results for paths are valid for bipartite graphs as well.

Turning to hexagonal graphs, we note that an optimal solution (for any kind of graph) can be described to the online algorithm using $(n+1)\lceil \log n \rceil$ bits of advice. We adapt two known approximation algorithms to obtain a $\frac{4}{3}$-competitive algorithm reading at most $n+2|V|$ bits of advice and a $\frac{3}{2}$-competitive algorithm reading at most $\log n + O(\log \log n)$ bits of advice. Finally, we prove the following lower bounds. To obtain an asymptotic competitive ratio strictly smaller than $\frac{5}{4}$, a linear number of bits are needed. For a strict competitive ratio strictly smaller than $\frac{4}{3}$, we also give a linear lower bound, namely $\left\lfloor \frac{n-1}{3} \right\rfloor$, on the number of advice bits needed. All results for hexagonal graph, except the $\frac{4}{3}$-competitive algorithm, hold for 3-colorable graphs in general.

The advice given to most of the algorithms is essentially the maximum number of requests given to any clique in the graph. For the underlying problem of frequency allocation, guessing these values based on previous data may not be unrealistic. Thus, the results in this paper could have practical applications. The results establish which type of information is useful, how algorithms should be designed to exploit this information, and what the limits are for what can be obtained.

Finally, our results lead to some more informal insight regarding locality and advice. When considering advice complexity of multi-coloring on a path, we can achieve 1-competitiveness with a small amount of advice. A recoloring algorithm needs to be 1-recoloring to achieve the same. The advice is basically the maximum number of requests to any two neighboring nodes. Thus, whether one has that global information once and for all, or can obtain and adjust according to the local variant of this information gives the same result.

For multi-coloring of hexagonal graphs, there is a similar connection between recoloring distance and advice. The 1-recoloring online version of the Fixed Preference Allocation algorithm discussed under related work has an advice variant and again, this advice represents information about the maximum number of requests to neighboring nodes. With additional global information about the bipartite induced subgraph, we can overcome the limitations of 1-recoloring algorithms and be as good as any known polynomial-time approximation algorithm.

5. The Path

As explained earlier, we establish all lower bounds for paths, and since a path is bipartite, all these negative results carry over to bipartite graphs. Similarly, all our (constructive) upper bounds are given for bipartite graphs and therefore also apply to paths. We start with two lower bound results.

5.1. Lower bounds

Theorem 1. Any strictly 1-competitive online algorithm for multi-coloring paths of at least 10 nodes has advice complexity at least $\lceil \log(\lfloor \frac{n}{4} \rfloor + 1) \rceil$.

Proof We let $m = \lfloor \frac{n}{4} \rfloor$ and define a set S of $m+1$ sequences, all having the same prefix of length $2m$. The set S will have the following property: for no two sequences in S can their prefixes be colored in the same way while ending up using the optimal number of colors on the complete sequence. Starting from one end of the path, we denote the nodes $v_1, v_2, \ldots$.

We define the set S to consist of the sequences $I_0, I_1, \ldots, I_m$, where I_i is defined in the following way. First m requests are given to each of the nodes v_1 and v_4. Then i requests to each of v_2 and v_3. To give all sequences the same length, the sequence ends with $n - 2m - 2i$ requests distributed as evenly as possible among v_6, v_8, and v_{10}. Since $\lceil (n - 2m - 2i)/3 \rceil \leq m$, the optimal number of colors will not be influenced by this part of the sequence.

We now explain that $\text{OPT}(I_i) = m+i$. At least $m+i$ colors are needed, since the neighboring vertices v_1 and v_2 receive $m+i$ requests in total. On the other hand, the requests of I_i can be colored using $m+i$ colors in total: colors $1, 2, \ldots, m$ at vertex v_1, colors $m+1, m+2, \ldots, m+i$ at vertex v_2, colors $1, 2, \ldots, i$ at vertex v_3, and colors $i+1, i+2, \ldots, m+i$ at vertex v_4. This is a legal coloring also with respect to v_2 and v_3, since $i \leq m$.

For any coloring of I_i, let x_i be the number of colors used at v_4 and not at v_1. Since the two vertices receive the same number of colors, this is also the number of colors used at v_1 and not at v_4. In the optimal coloring described above, $x_i = i$. Since the total number of colors used at v_1 and v_4 is $m+x_i$, any optimal coloring must have $x_i \leq i$. On the other hand, at most x_i of the colors used at v_1 can be used at v_3. Hence, the total number of colors used at v_1, v_2, and v_3 is at least $m + i + (i - x_i) = m + 2i - x_i$, so in an optimal coloring, $x_i \geq i$. This shows that to produce an optimal coloring, an algorithm must ensure that $x_i = i$.

The prefixes of length $2m$ in S are identical, so all information to distinguish between the different sequences must be given as advice. The cardinality of S is $m+1$. To specify one out of $m+1$ possible actions, $\lceil \log(m+1) \rceil$ bits are necessary. □

For algorithms that are $\frac{9}{8}$-competitive or better, we give the following lower bound.

Theorem 2. Consider multi-coloring paths of at least 10 nodes. For any $b \geq 3$ and any $(1 + \frac{1}{2^b})$-competitive algorithm, A, there exists an $N \in \mathbb{N}$ such that A has advice complexity at least $b - 2$ on sequences of length at least N.

Proof For any $(1 + \frac{1}{2^b})$-competitive algorithm, A, there exists an $\alpha \geq 1$ such that $\text{A}(I) \leq (1+\frac{1}{2^b})\,\text{OPT}(I)+\alpha$, for any input sequence I. We consider sequences of length $n \geq 2^{2b+2}\alpha + 3$.

Let $m = \left\lfloor \frac{n}{4} \right\rfloor$ and consider the same set of sequences as in the proof of Theorem 1. For the sequence I_i, let x_i denote the number of colors that A uses on

v_4, but not on v_1. As explained in the proof of Theorem 1, $\textsc{Opt}(I_i) = m + i$ and $\mathrm{A}(I_i) \geq \max\{m + x_i, m + 2i - x_i\}$.

We will prove that there are $p \geq 2^{b-2}$ sequences $I_{i_1}, I_{i_2}, \ldots, I_{i_p}$ such that, for any pair $i_j \neq i_k$, we have $x_{i_j} \neq x_{j_k}$, or otherwise A would not be $(1 + \frac{1}{2^b})$-competitive. This will immediately imply that A must use at least $b - 2$ advice bits.

Let $\varepsilon = \frac{1}{2^b} + \frac{1}{2^{2b}}$. From $\mathrm{A}(I_i) \leq (1 + \frac{1}{2^b})\,\textsc{Opt}(I_i) + \alpha$ and $m \geq 2^{2b}\alpha$, we obtain the inequalities

$$m + x_i \leq (1 + \varepsilon)(m + i)$$

and

$$m + 2i - x_i \leq (1 + \varepsilon)(m + i)$$

which reduce to

$$x_i \leq \varepsilon m + (1 + \varepsilon)i \tag{1}$$

and

$$i \leq \frac{x_i + \varepsilon m}{1 - \varepsilon} \tag{2}$$

By (1), $x_0 \leq \varepsilon m$. Hence, by (2), we can have $x_i = x_0$, only if $i \leq \frac{2\varepsilon m}{1-\varepsilon}$. Therefore, we let $i_1 = 0$ and $i_2 = \lfloor \frac{2\varepsilon m}{1-\varepsilon} + 1 \rfloor$. In general, we ensure $x_{i_j} \neq x_{i_{j+1}}$ by letting $i_{j+1} = \lfloor \frac{x_{i_j} + \varepsilon m}{1-\varepsilon} + 1 \rfloor$. Thus,

$$\begin{aligned} i_{j+1} &\leq \frac{x_{i_j} + \varepsilon m}{1 - \varepsilon} + 1 \\ &\leq \frac{\varepsilon m + (1 + \varepsilon) i_j + \varepsilon m}{1 - \varepsilon} + 1, \text{ by (1)} \\ &= \frac{1 + \varepsilon}{1 - \varepsilon} \cdot i_j + \frac{2\varepsilon m}{1 - \varepsilon} + 1 \end{aligned}$$

Solving this recurrence relation, we get

$$\begin{aligned}
i_{j+1} &\leq \left(\frac{1+\varepsilon}{1-\varepsilon}\right)^j \cdot i_1 + \sum_{k=0}^{j-1} \left(\frac{1+\varepsilon}{1-\varepsilon}\right)^k \left(\frac{2\varepsilon m}{1-\varepsilon} + 1\right) \\
&= \left(\frac{1+\varepsilon}{1-\varepsilon}\right)^j \cdot 0 + \frac{\left(\frac{1+\varepsilon}{1-\varepsilon}\right)^j - 1}{\frac{1+\varepsilon}{1-\varepsilon} - 1} \left(\frac{2\varepsilon m}{1-\varepsilon} + 1\right) \\
&= \frac{\left(\frac{1+\varepsilon}{1-\varepsilon}\right)^j - 1}{1+\varepsilon-1+\varepsilon} (2\varepsilon m + 1 - \varepsilon) \\
&= \frac{\left(\frac{1+\varepsilon}{1-\varepsilon}\right)^j - 1}{2\varepsilon} (2\varepsilon m + 1 - \varepsilon)
\end{aligned}$$

We let p equal the largest j for which $i_j \leq m$:

$$\begin{aligned}
& m < i_{p+1} \leq \frac{\left(\frac{1+\varepsilon}{1-\varepsilon}\right)^p - 1}{2\varepsilon} (2\varepsilon m + 1 - \varepsilon) \\
\Rightarrow\ & 2\varepsilon m < \left(\frac{1+\varepsilon}{1-\varepsilon}\right)^p (2\varepsilon m + 1 - \varepsilon) - (2\varepsilon m + 1 - \varepsilon) \\
\Leftrightarrow\ & \frac{4m\varepsilon + 1 - \varepsilon}{2m\varepsilon + 1 - \varepsilon} < \left(\frac{1+\varepsilon}{1-\varepsilon}\right)^p \\
\Leftrightarrow\ & \ln\left(2 - \frac{1-\varepsilon}{2m\varepsilon + 1 - \varepsilon}\right) < p \cdot \ln\left(1 + \frac{2\varepsilon}{1-\varepsilon}\right) \\
\Rightarrow\ & \ln\left(2 - \frac{1-\varepsilon}{2m\varepsilon + 1 - \varepsilon}\right) < p \cdot \frac{2\varepsilon}{1-\varepsilon}, \text{ since } \ln(1+x) \leq x, \text{ for } x > -1 \\
\Rightarrow\ & \ln\left(2 - \frac{1}{16}\right) < p \cdot \frac{2\varepsilon}{1-\varepsilon}, \text{ since } m\varepsilon > 2^b \geq 8 \\
\Rightarrow\ & \ln\left(\sqrt{e}\right) < p \cdot \frac{2\varepsilon}{1-\varepsilon} \\
\Leftrightarrow\ & \frac{1}{2} < p \cdot \frac{2\varepsilon}{1-\varepsilon} \\
\Leftrightarrow\ & p > \frac{1-\varepsilon}{4\varepsilon} \\
\Leftrightarrow\ & p > \frac{1 - \frac{1}{2^b} - \frac{1}{2^{2b}}}{\frac{4}{2^b} + \frac{4}{2^{2b}}} = \frac{2^{2b} - 2^b - 1}{2^{b+2} + 4} > 2^{b-2} - 1 \\
\Rightarrow\ & p \geq 2^{b-2}, \text{ since } p \text{ is an integer}
\end{aligned}$$

This completes the proof. □

5.2. Upper bounds

In this section we use a result by Christ, Favrholdt, and Larsen [16] establishing a strictly 1-competitive 1-recoloring algorithm, GreedyOpt. First, we adapt the algorithm to use advice instead of recoloring, yielding Theorem 3. Then, in Theorem 4, we establish a trade-off between the number of advice bits and the competitive ratio.

GreedyOpt divides the nodes into two sets, *upper* and *lower*, such that every second node belongs to *upper* and the remaining nodes belong to *lower*. The following invariant is maintained: After each request, each node in *lower* uses consecutive colors starting with the color 1 and each node in *upper* uses consecutive colors ending with a color no larger than the optimal number of colors for the sequence of requests seen so far.

The algorithm for paths from [16] can be generalized to work on bipartite graphs by letting the nodes of one partition, L, belong to *lower* and the nodes of the other partition, U, belong to *upper*. Recoloring is only needed if the number of colors used by an optimal offline algorithm is not known. Hence, using $enc(\text{Opt})$ advice bits, an online algorithm can be strictly 1-competitive, even if recoloring is not allowed. We call the resulting algorithm GreedyOptAdvice.

To describe the algorithm GreedyOptAdvice in detail, we need some notation: Let $f_i(v)$ denote the set of colors assigned to node v after the first i requests, starting with request 1. Also, for notational convenience, we define $f_0(v) = \emptyset$ for all v. This notation will be used throughout. GreedyOptAdvice is listed as Algorithm 1.

Theorem 3. Algorithm GreedyOptAdvice is correct, strictly 1-competitive, and has advice complexity $enc(\text{Opt})$.

Proof We consider correctness first. Clearly, at time i, the maximum color assigned to a node $v \in L$ is $|f_i(v)|$ and the minimum color assigned to a node $v \in U$ is $\text{Opt} + 1 - |f_i(v)|$ (assuming v has received at least one request).

Algorithm 1 The multi-coloring algorithm GreedyOptAdvice.

Assume that a bipartite graph is given by the partition into L and U.
Advice: $m = \text{Opt}$
for $i = 1$ **to** n **do**
  Assume that the ith request, r, is to node v
  if $v \in U$ **then**
    /* using the upper colors top-down */
    give r color $m - |f_{i-1}(v)|$
  else
    /* $v \in L$; using the lower colors bottom-up */
    give r color $|f_{i-1}(v)| + 1$

Assume for the sake of contradiction that, at some time i, a request to a node $l \in L$ gets assigned the same color c as a request to a neighboring node u, which must belong to U. This means that $c = |f_i(l)|$ and $c \geq \text{Opt} + 1 - |f_i(u)|$. Since l and u are neighbors, $\text{Opt} \geq |f_i(l)| + |f_i(u)| \geq c + \text{Opt} + 1 - c = \text{Opt} + 1$. This is a contradiction, so GreedyOptAdvice is correct.

It follows directly that the maximum color that GreedyOptAdvice assigns is Opt, implying that GreedyOptAdvice is strictly 1-competitive. □

We now turn to nonoptimal variants of GreedyOptAdvice using fewer than $enc(\text{Opt})$ advice bits. We show how to obtain a particular competitive ratio of $1 + \frac{1}{2^b}$, using $b + 1 + O(\log \log \text{Opt})$ bits of advice. Thus, essentially, we are approaching optimality exponentially fast in the number of bits of advice.

Theorem 4. For any integer $b \geq 1$, there exists a strictly $(1 + \frac{1}{2^{b-1}})$-competitive online algorithm for multi-coloring bipartite graphs with advice complexity $b + enc(a)$, where $a + b$ is the total number of bits in the value Opt.

Proof As advice, the algorithm asks for the b high order bits of the value Opt, as well as the number $a = \lceil \log(\text{Opt} + 1) \rceil - b$ of low order bits, but not the value of these bits. The algorithm knows b and can therefore just read the first b bits, while a needs to be encoded. Thus, $b + enc(a)$ bits are sufficient to encode the advice.

First, if $\textsc{Opt}$ contains at most b bits, this is detected by a being zero. In this case, some of the b bits may be leading zeros. By Theorem 3, we can then be strictly 1-competitive.

Now assume that $\textsc{Opt}$ contains more than b bits. Let $\textsc{Opt}_b = \left\lfloor \frac{\textsc{Opt}}{2^a} \right\rfloor$ denote the value represented by the b high order bits. Then the algorithm computes $m = 2^a \textsc{Opt}_b + 2^a - 1$ and runs GreedyOptAdvice with this m. Since $\textsc{Opt} \leq m \leq \textsc{Opt} + 2^a - 1$, the algorithm is correct and uses at most $\textsc{Opt} + 2^a - 1$ colors.

For any number $x \geq 1$, consisting of c bits, with the most significant bit being one, $2^c \leq 2x$. Thus, $2^{b+a} \leq 2\,\textsc{Opt}$, so $2^a \leq \frac{2\,\textsc{Opt}}{2^b}$. This means that the number of colors used by GreedyOptAdvice is less than $\textsc{Opt} + \frac{2\,\textsc{Opt}}{2^b} = (1 + \frac{1}{2^{b-1}})\,\textsc{Opt}$, so the algorithm is strictly $(1 + \frac{1}{2^{b-1}})$-competitive. □

Corollary 1. For any $\varepsilon > 0$, there exists a strictly $(1+\varepsilon)$-competitive deterministic online algorithm for multi-coloring bipartite graphs with advice complexity $O(\log \log \textsc{Opt})$.

Proof Except for the term b, the advice stated in Theorem 4 is $O(\log \log \textsc{Opt})$. Thus, we just need to bound the term b. For a given ε, choose b large enough such that $\frac{1}{2^{b-1}} \leq \varepsilon$. Using this value for b in Theorem 4, we obtain an algorithm with a strict competitive ratio of at most $1 + \frac{1}{2^{b-1}} \leq 1 + \varepsilon$. Since, for any given ε, b is a constant, the total amount of advice is $O(\log \log \textsc{Opt})$. □

5.3. Cancellations

The Multi-Coloring problem is sometimes considered in the context of request cancellations, i.e., a color already given to a node disappears again. We observe that even using the weakest form of recoloring, namely 0-recoloring, where only requests at the node where the cancellation takes place may be recolored, we can extend the algorithm GreedyOptAdvice, using the same advice, to a strictly 1-competitive algorithm. This is simply done by recoloring at most one request per cancellation to ensure that the invariants regarding lower and upper nodes are maintained, i.e., ensuring that the colors used at any node form a consecutive sequence starting from one and increasing and starting from Opt and decreasing for lower and upper nodes, respectively. This algorithm, GreedyOptAdviceCancel, is listed as Algorithm 2. Note that the difference to Algorithm 1 is the check in line 5 as to whether the current request is a color request and the addition of lines 12–19 handling cancellations.

6. Hexagonal Graphs

A hexagonal graph is a graph that can be obtained by placing (at most) one node in each cell of a hexagonal grid (such as the one sketched in Figure 1) and adding an edge between any pair of nodes placed in neighboring cells. Note that any hexagonal graph can be 3-colored by using the three colors cyclically on the cells of each row of the underlying hexagonal grid. In fact, as mentioned earlier, all lower bounds in this section carry over to 3-colorable graphs, and all upper bound proofs, except for the final theorem, also hold for 3-colorable graphs.

6.1. Lower bounds

Theorem 5. Any online algorithm for multi-coloring hexagonal graphs with a strict competitive ratio strictly smaller than $\frac{3}{2}$ has advice complexity at least $\left\lfloor \frac{n-1}{3} \right\rfloor$.

Proof First, we explain a small part of the construction that we will use in many copies. We consider two sequences with the same prefix of length 2. Both sequences can be colored with two colors, but this requires coloring the two prefixes of length two differently. Consider the left-most part of Figure 1 (surrounded by thick lines) consisting of the "double" nodes D_1 and D_2, the "outer" nodes O_0 and O_1 and the "single" nodes S_1, and S_2. These nodes form the same type of configuration as the nodes D_3, D_4, O_1, O_2, S_3, and S_4. If a pair of outer nodes are given some requests, they can later be "connected" by

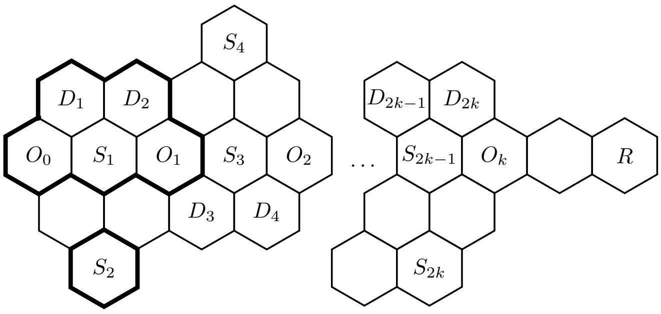

Figure 1: Hexagonal lower bound construction.

follow-up requests to either the two double nodes or the single node between them.

First the nodes O_0 and O_1 get one request each. Then, either D_1 and D_2 or S_1 and S_2 receive one request each. The node S_2 is used to get up to the same sequence length in all cases. In order not to use more than two colors, the outer nodes have to use different colors if we later give requests to the two D-nodes. Similarly, the O-nodes should have the same color if we later give a request to the S-node in between them. Since the prefix of length two is $\langle O_0, O_1 \rangle$ for both sequences, all information for an algorithm to distinguish between the two sequences must be given as advice.

We can repeat this graph pattern $\lfloor \frac{n-1}{3} \rfloor$ times, as illustrated in Figure 1 with $k = \lfloor \frac{n-1}{3} \rfloor$, giving the requests to all O-nodes first.

We now define the set of sequences S of cardinality $2^{\lfloor \frac{n-1}{3} \rfloor}$ formally, i.e., we define a sequence for each possible combination of requests to either D_{2j-1} and D_{2j} or S_{2j-1} and S_{2j}, for $j = 1, 2, \ldots, \lfloor \frac{n-1}{3} \rfloor$. A sequence is defined for any chosen combination of the following i-values, i.e., by choosing a tuple $(i_1, i_2, \ldots, i_{\lfloor \frac{n-1}{3} \rfloor}) \in \{0,1\}^{\lfloor \frac{n-1}{3} \rfloor}$. For any such choice, we define the sequence as a concatenation of the subsequences given below. In the description of the subsequences, we use the notation $\mathrm{Req}(v, m)$ to denote a sequence of m requests to a node v, and also use this notation for $m = 0$, denoting the empty request sequence, and $m = 1$, denoting one request.

- $\mathrm{Req}(O_j, 1)$, for $j = 0, 1, \ldots, \left\lfloor \frac{n-1}{3} \right\rfloor$
- $\mathrm{Req}(D_{2j-1}, i_j)$, $\mathrm{Req}(D_{2j}, i_j)$, for $j = 1, 2, \ldots, \left\lfloor \frac{n-1}{3} \right\rfloor$
- $\mathrm{Req}(S_{2j-1}, 1 - i_j)$, $\mathrm{Req}(S_{2j}, 1 - i_j)$, for $j = 1, 2, \ldots, \left\lfloor \frac{n-1}{3} \right\rfloor$
- $\mathrm{Req}(R, n - (3 \left\lfloor \frac{n-1}{3} \right\rfloor + 1))$

Note that for any i_j, $\{i_j, 1 - i_j\} = \{0, 1\}$ and we either give requests to the D-nodes or the S-nodes. The possible requests to R simply brings all sequences up to a length of n.

The node O_0 is given some color. After that, we have $\left\lfloor \frac{n-1}{3} \right\rfloor$ independent choices of coloring each node O_i in the prefix of any sequence identically to O_{i-1} or not. Since the prefixes are the same, all information for an algorithm to distinguish between the different sequences must be given as advice. To specify one out of $2^{\left\lfloor \frac{n-1}{3} \right\rfloor}$ possible actions, $\left\lceil \log 2^{\left\lfloor \frac{n-1}{3} \right\rfloor} \right\rceil = \left\lfloor \frac{n-1}{3} \right\rfloor$ bits are necessary. □

Theorem 6. Any online algorithm for multi-coloring hexagonal graphs with competitive ratio strictly smaller than $\frac{5}{4}$ has advice complexity $\Omega(n)$.

Proof We use the basic construction from Theorem 5. Assume p requests are given to one of the components like this:

First, we give $\frac{p}{4}$ requests to each of O_0 and O_1. Let q, $0 \leq q \leq \frac{p}{4}$, denote the number of colors used at both nodes. Then following up by giving $\frac{p}{4}$ requests to each S-node results in a minimum of $\frac{3p}{4} - q$ colors used, while giving the requests to the D-nodes instead results in a minimum of $\frac{p}{2} + q$ colors.

Note that $\textsc{Opt} = \frac{p}{2}$, independent of in which of the two ways the sequence is continued. Thus, for any $\varepsilon > 0$, any $(\frac{5}{4} - \varepsilon)$-competitive algorithm must choose q such that, for some constant α, $\frac{3p}{4} - q \leq \left(\frac{5}{4} - \varepsilon\right) \frac{p}{2} + \alpha$ and $\frac{p}{2} + q \leq \left(\frac{5}{4} - \varepsilon\right) \frac{p}{2} + \alpha$. Adding these two inequalities, we obtain $\frac{5p}{4} \leq (\frac{5}{4} - \varepsilon)p + 2\alpha$ which is equivalent to $\varepsilon p \leq 2\alpha$. Thus, if p is non-constant, no $(\frac{5}{4} - \varepsilon)$-competitive algorithm can use the same value of q for both sequences.

Now assume for the sake of contradiction that for some advice of $g(n) \in o(n)$ bits, we can obtain a ratio of $\frac{5}{4} - \varepsilon$. Let $f(n) = \frac{1}{2}\frac{n}{g(n)}$. Since $g(n) \in o(n)$, $f(n) \in \omega(1)$. The idea is now to repeat the construction as in the proof of Theorem 5 and give $f(n)$ requests to each construction ($f(n)$ has the role of p in the above). Since a pair of neighboring constructions share $f(n)/4$ requests, this results in $\frac{n-f(n)/4}{3f(n)/4} = \frac{4n-f(n)}{3f(n)} \geq \frac{n}{f(n)}$ constructions. We assume without loss of generality that all our divisions result in integers.

In order to be $(\frac{5}{4} - \varepsilon)$-competitive, an online algorithm must, for each two neighboring O-nodes, choose between at least two different values of q. These are independent decisions, and the ratio only ends up strictly better than $\frac{5}{4}$ if the algorithm decides correctly in every subconstruction. Thus, it needs at least $\frac{n}{f(n)}$ bits of advice. However, $\frac{n}{f(n)} = \frac{n}{\frac{1}{2}\frac{n}{g(n)}} = 2g(n) > g(n)$, which is a contradiction. □

6.2. Upper bounds

We start with a fairly trivial upper bound on the advice necessary to be optimal, independent of the graph topology. This is a weak type of result, basically asking for advice stating exactly what OPT would do with each request.

Theorem 7. There exists a strictly 1-competitive online multi-coloring algorithm with advice complexity $(n+1)\lceil \log \text{OPT} \rceil$.

Proof Start by asking for the number of bits necessary to represent values up to OPT. Then for each request, read $\lceil \log(\text{OPT}+1) \rceil$ bits, telling which color to use. This gives $enc(\lceil \log \text{OPT} \rceil) + n\lceil \log \text{OPT} \rceil < (n+1)\lceil \log \text{OPT} \rceil$. □

In the following, we will show how two known approximation algorithms can be converted to online algorithms with advice. In the description of the algorithms, we let the *weight* of a clique denote the total number of requests to the nodes of the clique. Note that the only maximal cliques in a hexagonal graph are isolated nodes, edges, or triangles. We let ω denote the maximum weight of any clique in the graph.[2]

A $\frac{3}{2}$-competitive algorithm called the Fixed Preference Allocation algorithm, FPA, was proposed by Janssen, Kilakos, and Marcotte [25]. Narayanan [31] simplified the strategy and it was noted that the algorithm can be converted to a 1-recoloring online algorithm. We describe the simplified offline algorithm below.

The algorithm uses three color classes, R, G, and B. The color classes represent a partitioning of the nodes in the graph so that no two neighbors are in the same partition. Each of the three color classes has its own set of $\lceil \frac{\omega}{2} \rceil$ colors, and each node in a given color class uses the colors of its color class, starting with the smallest. This set of colors is also referred to as the node's *private* colors. If more than $\lceil \frac{\omega}{2} \rceil$ requests are given to a node, then it borrows colors from the private colors of one of its neighbors, taking the highest available color. R nodes can borrow colors from G nodes, G from B, and B from R.

For completeness, we give the arguments that FPA is correct and obtains an approximation ratio of $\frac{3}{2}$. Assume for the purpose of contradiction that the coloring produced by the algorithm causes a conflict between an R node and a G node. This means that their combined number of requests must be greater than ω, which is a contradiction. The same argument holds for the other color combinations. Thus, the coloring is legal. Any optimal algorithm needs at least ω colors, so $\textsc{Opt} \geq \omega$ and the algorithm is a $\frac{3}{2}$-approximation algorithm.

Since $\lceil \frac{\omega}{2} \rceil \leq \lceil \frac{\textsc{Opt}}{2} \rceil$, we can give $\lceil \frac{\omega}{2} \rceil$ as advice, resulting in Algorithm 3. Note that the f-notation used in the pseudo-code was defined in connection with Algorithm 1.

Theorem 8. There exists a $\frac{3}{2}$-competitive online algorithm for multi-coloring

[2]The Greek letter ω is traditionally used here, so we will also do that. Since there is no argument, this should not give rise to confusion with the $\omega(f)$, stemming from asymptotic notation.

hexagonal graphs with advice complexity $enc(\lceil \frac{\textsc{Opt}}{2} \rceil)$.

Proof Given $\lceil \frac{\omega}{2} \rceil \leq \lceil \frac{\textsc{Opt}}{2} \rceil$ as advice, FPA can be used as an online algorithm (Algorithm 3). □

McDiarmid and Reed [30] introduced an algorithm with an approximation ratio of $\frac{4}{3}$. We now describe this algorithm. For completeness, we also give the arguments that the algorithm is correct and is a $\frac{4}{3}$-approximation algorithm:

The algorithm uses color classes in the same way as FPA, except that the private color sets contain only $\lfloor \frac{\omega+1}{3} \rfloor$ colors each. We use the following notation. For any node v, we let n_v denote the number of requests to v. Furthermore, b_v denotes the maximum number of colors that v can borrow, i.e., $b_v = \max\{0, \lfloor \frac{\omega+1}{3} \rfloor - n'_v\}$, where n'_v is the maximum number of requests to any of the neighboring nodes in the color class that v can borrow from.

The algorithm can be seen as working in up to three phases:

In the *first phase*, the algorithm colors $\min\{n_v, \lfloor \frac{\omega+1}{3} \rfloor\}$ requests to each node, v, using the node's private colors. Let G_1 be the graph induced by the nodes that still have uncolored requests after Phase 1.

For any node, v, in G_1, $\lfloor \frac{\omega+1}{3} \rfloor$ requests to v are colored with v's private colors in Phase 1. By the definition of ω, this immediately implies that any pair of neighboring nodes have a total of at most $\omega - 2\lfloor \frac{\omega+1}{3} \rfloor$ uncolored requests after Phase 1. Furthermore, G_1 cannot contain triangles. Each node in such a triangle would have received at least $\lfloor \frac{\omega+1}{3} \rfloor + 1$ requests, contradicting the definition of ω.

In the *second phase*, each node v of G_1 borrows $\min\{n_v - \lfloor \frac{\omega+1}{3} \rfloor, b_v\}$ colors. Let G_2 be the graph induced by nodes that still have uncolored requests after Phase 2.

Using the fact that G_1 (and hence G_2) does not contain triangles, it can be proven that G_2 is bipartite. Since any pair of neighbors in G_1 have a total of at most $\omega - 2\lfloor \frac{\omega+1}{3} \rfloor$ uncolored requests already after Phase 1, this means that, in

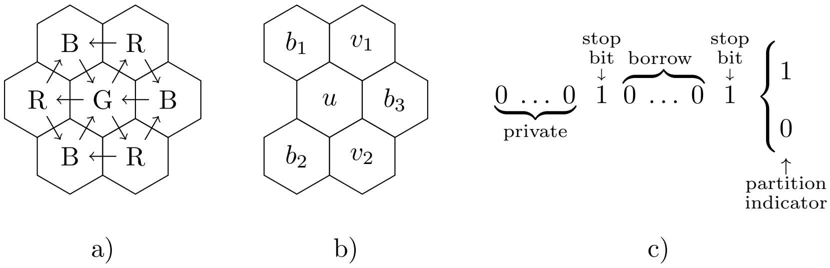

Figure 2: Illustration of the $\frac{4}{3}$-approximation algorithm. a) The borrow pattern. Arrows show the direction of the flow of colors in Phase 2. b) Part of a graph induced by nodes still having unprocessed requests after Phase 2. c) The subsequence of advice bits connected to one node. The sequence of advice bits is a merge of such sequences.

the third phase, the remaining requests can be colored with GreedyOpt (see Section 5.2) using $\omega - 2\left\lfloor\frac{\omega+1}{3}\right\rfloor$ additional colors.

We now argue that G_2 is acyclic and hence bipartite. Assume to the contrary that G_2 does contain a cycle, C. Assume without loss of generality that the R, G, B coloring of the underlying hexagonal grid is as shown in Figure 2a and let u be a leftmost node of C. We assume the borrowing pattern shown in Figure 2a, i.e., R nodes borrow from G nodes and so on. (Alternatively, one could use the pattern R nodes borrowing from B nodes and so on, and define u to be a rightmost node.)

Referring to Figure 2b, two of the nodes v_1, v_2, and b_3 must also be part of C. However, b_3 cannot be part of C, since then there would be a triangle after Phase 1. Thus, u, v_1, and v_2 are part of the cycle and hence receive at least $\left\lfloor\frac{\omega+1}{3}\right\rfloor + 1$ requests each.

Since u could not borrow enough colors from the nodes in the color class it is allowed to borrow from, one of the b-nodes, say b_j, together with u must have a total of at least $2\left\lfloor\frac{\omega+1}{3}\right\rfloor + 1$ requests. So, b_j and u must form a triangle together with either v_1 or v_2 so that the three nodes together have received a total of at least $(2\left\lfloor\frac{\omega+1}{3}\right\rfloor + 1) + (\left\lfloor\frac{\omega+1}{3}\right\rfloor + 1)$ requests. This quantity is strictly larger than ω, contradicting the definition of ω.

This ends the argument that the algorithm is correct.

Since the total number of colors used is at most $3\left\lfloor\frac{\omega+1}{3}\right\rfloor + (\omega - 2\left\lfloor\frac{\omega+1}{3}\right\rfloor) = \left\lfloor\frac{4\omega+1}{3}\right\rfloor$, the algorithm is a $\frac{4}{3}$-approximation algorithm.

We now show how an online algorithm, given the right advice, can behave as the offline $\frac{4}{3}$-approximation algorithm. Note that the three phases of the offline $\frac{4}{3}$-approximation algorithm are characterized by the coloring strategy (using the node's own private colors, borrowing private colors from neighbors, or coloring a bipartite graph). However, when requests arrive online, the nodes may not go from one phase to the next simultaneously.

Theorem 9. There exists a $\frac{4}{3}$-competitive online algorithm for multi-coloring hexagonal graphs with advice complexity at most $n + 2|V|$.

Proof We describe the algorithm and advice resulting in a coloring with at most $\frac{4}{3}\,\textsc{Opt} + \frac{1}{3}$ colors (see Algorithm 4, where we use the f-notation defined in connection with Algorithm 1).

Initially, each node is in Phase 1. On a request, the algorithm reads an advice bit and if it is zero, the next color from its private colors is used. If, instead, a one is read, this is treated as a stop bit for Phase 1, and this particular node enters Phase 2.

The algorithm starts with empty private color sets, and adds one color to each set whenever necessary, i.e., whenever a Phase 1 node that has already used all its private colors receives an additional request (this includes the first request to the node). As soon as a node leaves Phase 1, the algorithm knows that this node received $\left\lfloor\frac{\omega+1}{3}\right\rfloor$ requests, which is then the final size of each private color set. Knowing the size of the private color sets, the algorithm can calculate the maximum color for the complete coloring of the graph as $m = 4\left\lfloor\frac{\omega+1}{3}\right\rfloor$.

In Phase 2, every zero indicates that the algorithm should borrow a color. When another stop bit is received (which could be after no zeros at all if the borrowing phase is empty), it moves to Phase 3. In Phase 3, it reads one bit to decide which partition, upper or lower, of the bipartite graph it is in, and does not need more information after that, since it simply uses the colors $3\left\lfloor\frac{\omega+1}{3}\right\rfloor+1,\ldots,m$, either top-down or bottom-up.

If we allow the algorithm one bit per request, it needs at most two more bits per node, since the stop bits are the only bits that do not immediately tell the algorithm which action to take. Thus, $n+2|V|$ bits of advice suffice. $\square$

This algorithm can be used in many different ways, as long as the algorithm gets the information it needs. One other simple encoding would be to give the algorithm the value $\left\lfloor\frac{\omega+1}{3}\right\rfloor$ from the beginning and only give bit-wise advice after a node has used all its private colors. Since at least one color is private, this will save a total of at least $|V|$ bits, and result in at most $enc(\left\lfloor\frac{\omega+1}{3}\right\rfloor)+n+|V|$ bits of advice. This variant, and others, that are incomparable to each other, depending on the values of n, ω, and $|V|$, could all be used at the same time by first asking for a few bits to decide how to proceed. Thus, one could formulate a less readable but more accurate theorem basically taking the minimum of all the expressions. We have chosen clarity over precision, since the other expressions are mostly better in less interesting cases, where n is small compared to $|V|$, for instance.

Acknowledgment

The authors would like to thank the anonymous referees for improving the presentation and pointing out that our results for hexagonal graphs hold for 3-colorable graphs as well.

References

[1] S. Albers, L.M. Favrholdt, and O. Giel. On paging with locality of reference. *Journal of Computer and System Sciences*, 70(2):145–175, 2005.

[2] K. Barhum, H.-J. Böckenhauer, M. Forisek, H. Gebauer, J. Hromkovič, S. Krug, J. Smula, and B. Steffen. On the power of advice and randomization for the disjoint path allocation problem. In *SOFSEM*, volume 8327 of *LNCS*, pages 89–101. Springer, 2014.

[3] M. P. Bianchi, H.-J. Böckenhauer, J. Hromkovic, and L. Keller. Online coloring of bipartite graphs with and without advice. In *COCOON*, volume 7434 of *LNCS*, pages 519–530, 2012.

[4] M. P. Bianchi, H.-J. Böckenhauer, J. Hromkovič, S. Krug, and B. Steffen. On the advice complexity of the online $l(2,1)$-coloring problem on paths and cycles. *Theoretical Computer Science*, 554:22–39, 2014.

[5] H.-J. Böckenhauer, D. Komm, R. Královič, and R. Královič. On the advice complexity of the k-server problem. In *ICALP*, volume 6755 of *LNCS*, pages 207–218, 2011.

[6] H.-J. Böckenhauer, D. Komm, R. Královič, R. Královič, and T. Mömke. On the advice complexity of online problems. In *ISAAC*, volume 5878 of *LNCS*, pages 331–340, 2009.

[7] H.-J. Böckenhauer, D. Komm, R. Královič, and P. Rossmanith. The online knapsack problem: Advice and randomization. *Theoretical Computer Science*, 527:61–72, 2014.

[8] A. Borodin, S. Irani, P. Raghavan, and B. Schieber. Competitive paging with locality of reference. *Journal of Computer and System Sciences*, 50(2):244–258, 1995.

[9] J. Boyar, L.M. Favrholdt, K.S. Larsen, and M.N. Nielsen. Extending the Accommodating Function. *Acta Informatica*, 40(1):3–35, 2003.

[10] J. Boyar, S. Gupta, and K.S. Larsen. Access graphs results for LRU versus FIFO under relative worst order analysis. In *SWAT*, volume 7357 of *LNCS*, pages 328–339. Springer, 2012.

[11] J. Boyar, S. Kamali, K.S. Larsen, and A. López-Ortiz. Online bin packing with advice. In *STACS*, volume 25 of *LIPIcs*, pages 174–186. Schloss Dagstuhl – Leibniz-Zentrum für Informatik GmbH, 2014.

[12] J. Boyar and K.S. Larsen. The Seat Reservation Problem. *Algorithmica*, 25(4):403–417, 1999.

[13] J. Boyar, K.S. Larsen, and M.N. Nielsen. The Accommodating Function: a generalization of the competitive ratio. *SIAM Journal on Computing*, 31(1):233–258, 2001.

[14] J.W.-T Chan, F.Y.L. Chin, D. Ye, and Y. Zhang. Absolute and asymptotic bounds for online frequency allocation in cellular networks. *Algorithmica*, 58(2):498–515, 2010.

[15] J.W.-T. Chan, F.Y.L. Chin, D. Ye, Y. Zhang, and H. Zhu. Frequency allocation problems for linear cellular networks. In *ISAAC*, volume 4288 of *LNCS*, pages 61–70. Springer, 2006.

[16] M.G. Christ, L.M. Favrholdt, and K.S. Larsen. Online multi-coloring on the path revisited. *Acta Informatica*, 50(5–6):343–357, 2013.

[17] M. Chrobak, L. Jez, and J. Sgall. Better bounds for incremental frequency allocation in bipartite graphs. *Theoretical Computer Science*, 514:75–83, 2013.

[18] M. Chrobak and J. Sgall. Three results on frequency assignment in linear cellular networks. *Theoretical Computer Science*, 411(1):131–137, 2010.

[19] S. Dobrev, R. Královič, and D. Pardubská. Measuring the problem-relevant information in input. *RAIRO Theoretical Informatics and Applications*, 43(3):585–613, 2009.

[20] R. Dorrigiv, M. He, and N. Zeh. On the advice complexity of buffer management. In *ISAAC*, volume 7676 of *LNCS*, pages 136–145, 2012.

[21] P. Elias. Universal codeword sets and representations of the integers. *IEEE Transactions on Information Theory*, 21(2):194–203, 1975.

[22] Y. Emek, P. Fraigniaud, A. Korman, and A. Rosén. Online computation with advice. *Theoretical Computer Science*, 412(24):2642–2656, 2011.

[23] M. Forisek, L. Keller, and M. Steinová. Advice complexity of online coloring for paths. In *LATA*, volume 7183 of *LNCS*, pages 228–239, 2012.

[24] J. Hromkovič, R. Královič, and R. Královič. Information complexity of online problems. In *MFCS*, volume 6281 of *LNCS*, pages 24–36, 2010.

[25] J. Janssen, K. Kilakos, and O. Marcotte. Fixed preference channel assignment for cellular telephone systems. *IEEE Transactions on Vehicular Technology*, 48(2):533–541, 1999.

[26] J. Janssen, D. Krizanc, L. Narayanan, and S.M. Shende. Distributed online frequency assignment in cellular networks. *Journal of Algorithms*, 36(2):119–151, 2000.

[27] A.R. Karlin, M.S. Manasse, L. Rudolph, and D.D. Sleator. Competitive snoopy caching. *Algorithmica*, 3:79–119, 1988.

[28] D. Komm and R. Královič. Advice complexity and barely random algorithms. *RAIRO Theoretical Informatics and Applications*, 45(2):249–267, 2011.

[29] D. Komm, R. Královič, and T. Mömke. On the advice complexity of the set cover problem. In *CSR*, volume 7353 of *LNCS*, pages 241–252, 2012.

[30] C. McDiarmid and B.A. Reed. Channel assignment and weighted coloring. *Networks*, 36(2):114–117, 2000.

[31] L. Narayanan. *Channel Assignment and Graph Multicoloring*, pages 71–94. John Wiley & Sons, Inc., 2002.

[32] L. Narayanan and S.M. Shende. Static frequency assignment in cellular networks. *Algorithmica*, 29(3):396–409, 2001.

[33] L. Narayanan and S.M. Shende. Corrigendum: Static frequency assignment in cellular networks. *Algorithmica*, 32(4):679, 2002.

[34] S. Seibert, A. Sprock, and W. Unger. Advice complexity of the online coloring problem. In *CIAC*, volume 7878 of *LNCS*, pages 345–357, 2013.

[35] D.D. Sleator and R.E. Tarjan. Amortized efficiency of list update and paging rules. *Communications of the ACM*, 28(2):202–208, 1985.

[36] P. Sparl and J. Zerovnik. 2-local 4/3-competitive algorithm for multicoloring hexagonal graphs. *Journal of Algorithms*, 55(1):29–41, 2005.

[37] R. Witkowski and J. Zerovnik. 1-local 33/24-competitive algorithm for multicoloring hexagonal graphs. *Discrete Mathematics and Theoretical Computer Science*, 15(3):127–138, 2013.

Algorithm 2 The multi-coloring algorithm GreedyOptAdviceCancel.

Assume that a bipartite graph is given by the partition into L and U.
Advice: $m = \textsc{Opt}$
for $i = 1$ **to** n **do**
    Assume that the ith request, r, is to node v
    if r is a color request **then**
        if $v \in U$ **then**
            /* using the upper colors top-down */
            give r color $m - |f_{i-1}(v)|$
        else
            /* $v \in L$; using the lower colors bottom-up */
            give r color $|f_{i-1}(v)| + 1$
    else
        /* r is a cancellation */
        if $v \in U$ **then**
            if the color of r is different from $\min f_{i-1}(v)$ **then**
                recolor the request that has color $\min f_{i-1}(v)$, giving it the color of r
        else
            if the color of r is different from $\max f_{i-1}(v)$ **then**
                recolor the request that has color $\max f_{i-1}(v)$, giving it the color of r

Algorithm 3 The $\frac{3}{2}$-competitive algorithm, FPA, with advice.

Advice: $\left\lceil \frac{\omega}{2} \right\rceil$

$\textsc{Red} = \left\{1, 2, \ldots, \left\lceil \frac{\omega}{2} \right\rceil\right\}$,

$\textsc{Green} = \left\{\left\lceil \frac{\omega}{2} \right\rceil + 1, \left\lceil \frac{\omega}{2} \right\rceil + 2, \ldots, 2\left\lceil \frac{\omega}{2} \right\rceil\right\}$,

$\textsc{Blue} = \left\{2\left\lceil \frac{\omega}{2} \right\rceil + 1, 2\left\lceil \frac{\omega}{2} \right\rceil + 2, \ldots, 3\left\lceil \frac{\omega}{2} \right\rceil\right\}$

Function Class(v)

    return v's color class: R, G, or B

Function Borrow(c)

    return the next class in the wrap-around sequence R, G, or B

Function Colors(c)

    return the set of private colors of class c

for $i = 1$ **to** n **do**

    Assume that the ith request, r, is to node v

    if $|f_{i-1}(v)| < \left\lceil \frac{\omega}{2} \right\rceil$ **then**

        give r color min(Colors(Class(v)) $\setminus f_{i-1}(v)$)

    else

        give r color max(Colors(Borrow(Class(v))) $\setminus f_{i-1}(v)$)

Algorithm 4 Combining FPA and GreedyOptAdvice.

Advice: A sequence B of bits classifying each request as to whether it should be colored using the node's own private colors, by borrowing, or in which partition it falls.

Function Class(v)
  return v's color class: R, G, or B

Function Borrow(c)
  return the next class in the wrap-around sequence R, G, or B

Function Colors(c)
  return the set of private colors of class c

Function NextBit(B)
  return the next advice bit

for each node v **do**
  Phase(v) $= 1$

for $i = 1$ **to** n **do**
  Assume that the ith request, r, is to node v
  if Phase(v) $= 1$ **then**
    if NextBit(B) $= 0$ **then**
      if Colors(Class(v)) $\setminus f_{i-1}(v) = \emptyset$ **then**
        add one color to each of the three sets of private colors
      give r color min(Colors(Class(v)) $\setminus f_{i-1}(v)$)
    else
      Phase(v) $= 2$
      Phase3Min $= 3\,|f_{i-1}(v)| + 1$
      Phase3Max $= 4\,|f_{i-1}(v)| + 1$
  if Phase(v) $= 2$ **then**
    if NextBit(B) $= 0$ **then**
      give r color max(Colors(Borrow(Class(v))) $\setminus f_{i-1}(v)$)
    else
      Phase(v) $= 3$
      upper_v = NextBit(B) /* Store the partition of v */
  if Phase(v) $= 3$ **then**
    /* Use GreedyOptAdvice: */
    if $\text{upper}_v = 1$ **then**
      give r color max($\{\text{Phase3Min}, \ldots, \text{Phase3Max}\} \setminus f_{i-1}(v)$)
    else
      give r color min($\{\text{Phase3Min}, \ldots, \text{Phase3Max}\} \setminus f_{i-1}(v)$)