Advice Complexity of the Online Coloring Problem

Advice Complexity of the Online Coloring Problem

Sebastian Seibert[1], Andreas Sprock[2], and Walter Unger[1]

[1] Lehrstuhl für Informatik I, RWTH Aachen, Germany,
{seibert, quax}@cs.rwth-aachen.de
[2] Department of Computer Science, ETH Zurich, Switzerland,
andreas.sprock@inf.ethz.ch

Abstract. We study online algorithms with advice for the problem of coloring graphs which come as input vertex by vertex. We consider the class of all 3-colorable graphs and its sub-classes of chordal and maximal outerplanar graphs, respectively.
We show that, in the case of the first two classes, for coloring optimally, essentially $\log_2 3$ advice bits per vertex (bpv) are necessary and sufficient. In the case of maximal outerplanar graphs, we show a lower bound of 1.0424 bpv and an upper bound of 1.2932 bpv.
Finally, we develop algorithms for 4-coloring in these graph classes. The algorithm for 3-colorable chordal and outerplanar graphs uses 0.9865 bpv, and in case of general 3-colorable graphs, we obtain an algorithm using < 1.1583 bpv.

1 Introduction

An online algorithm deals with the natural situation that the input arrives piecemeal. In contrast to the offline case, the algorithm must compute a part of the solution for the already given piece of input at every time step. Once a part of the solution is computed, it must not be changed. The standard way to measure the quality of an online algorithm is the *competitive analysis*. Here, the quality of the solution given by the online algorithm is compared to the quality of the best possible solutions computable offline, i.e., after knowing the whole input. This concept was introduced in [16], for a more detailed introduction we refer to the standard literature, e.g., [3,9].
Measuring the quality of an online algorithm by comparing its output to an optimal offline solution gives a universally applicable yardstick. However, it has the disadvantage, that for many problems, the output of the best possible online algorithms will still be far from the offline solutions.
In order to better understand and quantify this gap, the idea of online algorithms with advice was introduced by [4] and has been further investigated, e.g., in [1,2,5,10]. Here, one asks how much additional information (advice) an online algorithm needs to close this gap, respectively, which progress can be made with limited advice.
Formally, one introduces an advisor (an oracle that knows the whole input) who provides an unlimited advice bit string to the online algorithm. For any online algorithm with advice, the *advice complexity* measures the number of bits read by the online algorithm. The *advice complexity* of a problem is defined as the advice complexity of the best online algorithm (achieving a given competitive ratio).

Coloring vertices of a graph such that no adjacent vertices get the same color is a very well known and intensively studied problem. For an online version, the most obvious input order is the following which will be studied here. In every step, a new vertex gets revealed, together with all edges between this one and previously revealed vertices. Now, the newly revealed vertex has to be colored immediately.

It turns out that online coloring is hard and no constant competitive ratio is possible [14]. For the class of k-colorable graphs on n vertices, it has been proven that any online coloring algorithm needs $\Omega\left((\log n/(4k))^{k-1}\right)$ colors in the worst case [17]. For an overview of classical online coloring see [12, 13].

A first study of online path coloring with advice was done in [6]. In this paper, we study online coloring with advice on the class of all 3-colorable graphs, and on its sub-classes of 3- colorable chordal and outerplanar graphs, respectively. We want to know how much advice is necessary, respectively sufficient, in order to color these graphs optimally. Also, we investigate how much advice can be saved if we allow the use of a fourth color. The results mentioned above imply that using only a constant number of colors is a big improvement over coloring without advice.

For a lower bound, we show that at least $1.0424 \cdot n$ advice bits are necessary to color a maximal outerplanar graph with n vertices optimally. For 3-colorable chordal graphs (and thus for general 3-colorable graphs), we get the lower bound of $(\log_2 3 - \varepsilon) \cdot n$ bits (for arbitrarily small ε).

On the other hand, we describe an algorithm to color general 3-colorable graphs optimally using $1.5863 \cdot n$ bits, and we can color maximal outerplanar graphs optimally using $1.2932 \cdot n$ bits (where n again is the number of vertices). Note that, here, the power of the advisor becomes apparent since 3-coloring is known to be $\mathcal{NP}$-hard [7].

Moreover, we analyze the advice needed for coloring 3-colorable graphs with a competitive ratio of 4/3. In other words, we want to color 3-colorable graphs with four colors. Note that, this problem is also known to be $\mathcal{NP}$-hard [8, 11]. Here, we show how to obtain a 4-coloring for any 3- colorable graph with 1.1583 bits per vertex.

Additionally, we develop an algorithm to color 3-colorable chordal graphs with four colors by using less than one bit (0.9865) per vertex. An overview is shown in Table 1.

number of bits per vertex	lower bounds	upper bounds	
	3-coloring	3-coloring	4-coloring
3-colorable graphs	$\log_2 3 - \varepsilon^3$	1.5863	1.1583
3-colorable chordal graphs	$\log_2 3 - \varepsilon$	1.5863	0.9865
maximal outerplanar graphs	1.0424	1.2932	0.9865

Table 1. Overview of the results on the number of bits per vertex for online coloring.

In Section 2, we fix our notation, and we show how the string of advice bits can efficiently be used for one-out-of-three decisions. Section 3 is dedicated to show our main results. Due to space restrictions, mainly we have restricted ourselves to describing the ideas, and we have moved proofs and formal algorithm descriptions to the appendix (see also [15]). We conclude in Section 4.

2 Preliminaries

We use the following notation for online algorithms analogous to [2], for coloring graphs, and for the problem at hand, respectively.

Definition 1. *[2] Consider an input sequence $I = (x_1, \ldots, x_n)$ for some minimization problem U. An* online algorithm $\mathtt{A}$ *computes the output sequence* $\mathtt{A}(I) = (y_1, \ldots, y_n)$*, where $y_i = f(x_1, \ldots, x_i)$ for some function f. The* cost *of the solution is given by* $\mathrm{cost}(\mathtt{A}(I))$*. An algorithm $\mathtt{A}$ is c-*competitive*, for some $c \geq 1$, if there exists a constant α such that, for every input sequence I,* $\mathrm{cost}(\mathtt{A}(I)) \leq c \cdot \mathrm{cost}(\mathtt{Opt}(I)) + \alpha$*, where $\mathtt{Opt}$ is an optimal offline solution for the problem. If $\alpha = 0$, then $\mathtt{A}$ is called* strictly c-competitive*. Finally, $\mathtt{A}$ is* optimal *if it is strictly 1-competitive.*

Definition 2. *Given a graph $G = (V, E)$, a coloring function c is a function that maps every vertex $v_i \in V$ to one color from $\{1, \ldots, k\}$, for some $k \in \mathbb{N}$, such that $c(v_i) \neq c(v_j)$, for all v_i, v_j with $\{v_i, v_j\} \in E(G)$. We denote the the minimal number of colors necessary for a coloring of G with $\chi(G)$.*
For any vertex $v \in V$, we denote by $Neigh(v) = \{w \in V \mid \{v, w\} \in E\}$ the set of neighbor vertices of v in G. If G is directed, E contains ordered pairs, and the set of predecessors of v is $Pred(v) = \{w \in V \mid (w, v) \in E\}$.

Definition 3. *The* ***Online Coloring Problem with Advice, in Vertex-Revealing Mode ($OColA_V$)*** *is the following online problem: The input is an unweighted, undirected graph $G = (V, E)$ with $|V(G)| = n$ and an order $\prec$ of revealing on the set of vertices. The goal is to find a minimum-cost coloring function $c : V \to \{1, \ldots, n\}$ for the vertices in G.*
In each time step i, the next vertex $v_i \in V$ (in the order $\prec$) is revealed, together with all edges $\{\{v_i, v_j\} \mid j < i\}$, and the online algorithm has to decide which color $c(v_i)$ the vertex v_i gets. To this end, it can ask for certain number of advice bits.

For every instance $I = (G, \prec)$, where $G = (V, E)$, we get a directed graph $G^{\prec} = (V, E')$ by giving a direction on every edge $e \in E$ depending on the order of revealing the vertices. Every edge $e = \{v_i, v_j\} \in E$ is directed from v_i to v_j, i.e., $(v_i, v_j) \in E'$, iff v_i is revealed before v_j. Additionally, for a subset of vertices $V_x \subset V(G)$, we denote by $G_{V_x} = G \mid_{V_x}$ the subgraph of G induced by V_x. For developing algorithms to color a 3-colorable graph optimally, we need a method to read a one-out-of-three decision from a Boolean advice string. For this we use the following lemma.

Lemma 1. *Reading several one-out-of-three decisions from a bit string costs $46/29 < 1.5863 \approx \log_2 3$ bits on average.*

Proof: When the first one-out-of-three decision is necessary, 46 bits get read from the advice string. By these 46 bits and the corresponding 2^{46} different possible bit allocations, 29 three-way decisions can be encoded, because $2^{46} \geq 3^{29}$. With this, for the first three-way decision, the algorithm gets the results of the next 28 three-way decisions at the same time and keeps them in its memory. This leads to an average of the information needed for a three-way decision of $46/29 < 1.5863 \approx \log_2 3$ bits. In general, if n one-out-of-three decisions have to be done, this costs at most $1.5863(n-1) + 46 = 1.5863n + d$ bits. $\square$

3 Algorithms and Lower Bounds

In this section, we first turn to the lower bounds. We will show that more than one bit of advice per vertex is necessary to color a maximal outerplanar graph optimally, i.e., by three colors.

Theorem 1. *For any $k \in \mathbb{N}$, there exists a maximal outerplanar graph G_k on $4k+9$ vertices and an ordering $\prec$ on the vertices of G_k such that every deterministic online algorithm for OColA$_V$ on $(G, \prec)$ needs at least $\frac{1}{2} \cdot \left(\log_2 3 + \frac{1}{2}\right) \cdot n - 12 > 1.0424 \cdot n - 12$ advice bits to generate an optimal coloring.*

The proof is given in Appendix A. Additionally, we want show that $\log_2 3$ bits per vertex are necessary for coloring any 3-colorable graph online optimally. For this, we prove (in Appendix B) the following lemma.

Theorem 2. *For any $k \in \mathbb{N}$, there exists a 3-colorable graph G on $n = k+3$ vertices and an ordering $\prec$ such that every deterministic online algorithm for OColA$_V$ on $(G, \prec)$ needs at least* $\log_2 3 \cdot (k-1) - 1 = \log_2 3 \cdot (n-4) - 1$ *advice bits to be optimal.*

Now we are ready to investigate several algorithms for coloring graphs online with given advice. Let G be a graph with an order $\prec$, and let $G^{\prec}$ be the corresponding directed graph. Depending on the direction of the edges in $G^{\prec}$, we define the function $p : V(G) \to \{1, 2, 3\}$ for all vertices in $G^{\prec}$, where $p(v) = i$ describes which position v has in a triangle.
Every vertex v_x, that was revealed as isolated, i.e., has outgoing edges only, is the first vertex of any triangle it belongs to. Such a vertex gets the number one, that is $p(v_x) = 1$. Every vertex v_y that is connected to one or more already revealed vertices, but not closing a triangle, has $p(v_y) = 2$. (In any triangle, there is at most one ingoing edge). Finally, every vertex v_z which closes one or more triangles (has two ingoing edges in one triangle) gets $p(v_z) = 3$. That way, we partition the vertices of a given input instance $G^{\prec}$ into three classes $V_i = \{v \in V(G) \mid p(v) = i\}$, for $i \in \{1, 2, 3\}$. For an example showing the different types of vertices, see Figure 1.

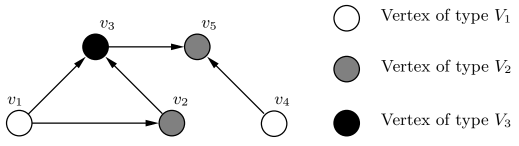


Fig. 1. Vertices of different types.

In every step of the coloring algorithm, when a new vertex v occurs, there exists a set of colors by which v may be colored. We denote, for every vertex v, the set of allowed colors by $C_v = \{1, \ldots, 3\} \setminus \{c(w) \mid w \in \mathit{Neigh}(v)\}$. Note that, since we use an ordered set of colors, we may speak of a 'smallest' color in C_v.
We start with Algorithm 1 (see Appendix E), which colors an arbitrary 3-colorable graph G online, with 3 colors, where G is revealed according to an order $\prec$. For this, we need 1.5863 bits per vertex on average.

The idea of the first algorithm is quite simple. For every isolated vertex $v \in V_1$, the algorithm asks for the optimal color (one out of three). For every vertex $w \in V_2$ connected to an already colored vertex, the algorithm asks for the correct color from the the remaining colors C_w (at most one out of two), and every vertex $x \in V_3$ gets colored by the only remaining color. This leads to the following lemma.

Lemma 2. *Let G be a graph with $\chi(G) = 3$, and let $G^{\prec}$ be an input instance for the OColA$_V$. Algorithm 1 colors G optimally with at most $(n-3)$ one-out-of-three decisions and at most one one-out-of-two decision. With this, Algorithm 1 uses less than* $1.5863 \cdot (n-3) + 1 + 45 = 1.5863 \cdot n + d$ *advice bits*[4].

Proof: Let $G^{\prec}$ be an input instance of an graph G with $\chi(G) = 3$ with $|V(G)| = n$ vertices. In the worst case, $G^{\prec}$ contains $n-2$ vertices in V_1. Otherwise, G could not contain a cycle, and it would be a forest and thus two-colorable. Algorithm 1 does not use information for the first vertex, because here the coloring can be arbitrary. For the second revealed vertex, even if it is revealed as isolated, only one bit of advice is necessary for knowing whether it gets the same color as the first vertex or a different one. For all further vertices, except the last one, a one-out-of-three decision might be necessary. Summing up, Algorithm 1 needs at most $(n-3)$ one-out-of-three and one one-out-of two decisions. We know from Lemma 1 that a one-out-of-three decision needs less than 1.5863 bits in the average. This leads to less than $1.5863 \cdot (n-3) + 1 + 45 = 1.5863 \cdot n + d$ bits at all. $\square$

Now, we observe that, since vertices in V_1 have outgoing edges only, no two of them can be connected.

Observation 1 *Let $G^{\prec}$ be the directed graph resulting from G and the order $\prec$ of revealing. Then the set V_1 is an independent set in $G^{\prec}$, respectively in G.*

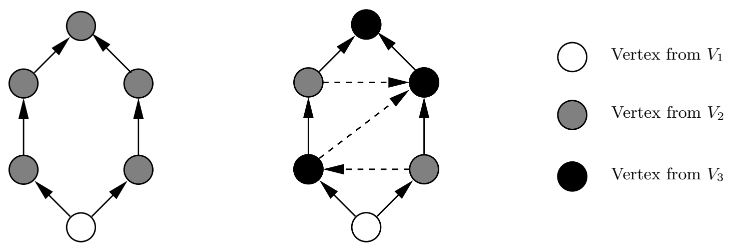
This leads us to the following lemma, which holds for general chordal graphs.

Lemma 3. *Let G be a chordal graph, $(G, \prec)$ be an input instance for OColA$_V$, and let $G^{\prec}$ be the corresponding directed graph. For the set $A = V_1 \cup V_2$, the subgraph $G^{\prec}_A$ is a forest.*

Proof: If $G^{\prec}_A$ is not a forest, it contains at least one cycle. This cycle has to be extended by edges to triangles because G is chordal. Hence, such a cycle contains at least three vertices from A which build a triangle. The vertex v_l from this triangle, which is revealed last, has two incoming edges in $G^{\prec}_A$ and is consequently an element of V_3 (see Figure 2(a),(b)), which is a contradiction to the assumption. $\square$

In the following, we analyze OColA$_\mathrm{V}$ on the class of outerplanar graphs. Therefore, we need some observations and lemma for this class of graphs. The following observations holds for input instances for OColA$_\mathrm{V}$ in general.

[4] The constant 45 is a result of the method to encode the one-out-of-three decisions in the advice.



(a) A hypothetical cycle in $G^{\prec}$ (b) Types of the vertices when all edges are included

Fig. 2. Example of a cycle in a chordal graph

Lemma 4. *Let G be an outerplanar graph with $\chi(G) = 3$ on n vertices and let $G^{\prec}$ be an input instance for OColA$_V$. For the set V_1 of vertices that are revealed as isolated, $|V_1| \leq 1/2 \cdot n$.*

Proof: Let G be an outerplanar graph and $G^{\prec}$ a corresponding input instance. This implies that all vertices in G lie on an outer cycle. We know from Lemma 1 that all vertices of type V_1 are independent in G. This implies that, between two vertices $v, w \in V_1$, there has to be at least one vertex $x \in V_2$ to connect them. This yields $|V_2| \geq |V_1| - 1$. Additionally, at least at the end one vertex $y \in V_3$ is necessary to close the cycle, otherwise G would be two- colorable. □

Using Lemma 4, we can analyze Algorithm 1 with the following result.

Lemma 5. *Let G be a maximal outerplanar graph with $V(G) = n$, and $G^{\prec}$ a corresponding input instance for the OColA$_V$. Then Algorithm 1 colors G optimally, using less than $1.29315 \cdot n + 45$ advice bits.*

Proof: Let G be a maximal outerplanar graph with $V(G) = n$, and $G^{\prec}$ a corresponding input instance for the OColA$_\mathrm{V}$. Let V_1, V_2, and V_3 be the corresponding sets of vertices. According to Lemma 4, $|V_1| \leq |V_2| + |V_3|$. This leads to the following inequalities: $\frac{|V_1|}{n} \leq 0.5$ and $\frac{|V_1|}{n} + \frac{|V_2|}{n} < 1$.
For the number of advice bits per vertex A_{BpV} for Algorithm 1, we have

$$A_{BpV} \leq \frac{|V_1|}{n} \cdot 1.5863 + \frac{|V_2|}{n}. \tag{1}$$

We maximize the right hand side of Equation 1 by letting $\frac{|V_1|}{n} = 0.5$. This implies $\frac{|V_2|}{n} \leq 0.5$, and thus $A_{BpV} \leq 0.5 \cdot 1.5863 + 0.5 = 1.2931$.
For the upper bound on the number of advice bits used by Algorithm 1, this means: $A_b \leq 1.29315 \cdot n + d$ where $d \leq 45$ is the number of bits needed to encode the last ≤ 28 one-out-of-three decisions. □

In addition to the results for an optimal coloring, we now give an alternative online algorithm, which colors an arbitrary 3-colorable graph G with 4 colors. The idea is to color all vertices of V_1, which are revealed as isolated, with an additional color 4 and

to ask for every revealed vertex from V_2 and V_3 for advice according to an optimal coloring of G using the colors $\{1,2,3\}$ (see Algorithm 2 in Appendix E).
Following this strategy, advice is only necessary for vertices of V_2 (1.5863 bits) and for vertices of V_3 (1 bit). So this strategy is efficient for instances with a high number of isolated vertices.
This leads us to Algorithm 3 (see Appendix E), which combines the strategies of Algorithm 1 and Algorithm 2. With it, we can color all 3-colorable graphs with at most four colors. To know what to do, the algorithm reads at the beginning the first bit of the advice tape and, depending on this bit, it decides which of the two strategies it follows.
The following lemma shows that Algorithm 3 colors G optimally if, for the vertices of $G^{\prec}$, $|V_1|/n \cdot 1.5863 + |V_3|/n \leq 1.15822$. Otherwise it colors G with four colors. In both cases, it needs at most $1.1582196 \cdot n + d$ advice bits.

Lemma 6. *Let G be a graph with $V(G) = n$ and $\chi(G) = 3$, and let $G^{\prec}$ be a corresponding input instance for the OColA$_V$. There exists an advice tape with which Algorithm 3 colors G with four colors, using at most* $1.1582196 \cdot n + 45$ *bits.*

Proof: There exists a 4-coloring for G, where all vertices revealed as isolated have the same color, because $\chi(G) = 3$ and the vertices from set V_1 are independent (see Lemma 1). In such a coloring, the algorithm needs a one-out-of-three decision for every vertex from V_2 and a one-out-of-two decision for every vertex from V_3, because it is already connected to at least one already colored vertex from V_2.
Now, we compute the maximum of advice bits used, by combining both algorithms. Algorithm 1 uses $\leq |V_1| \cdot 1.5863 + |V_2|$ bits, and Algorithm 2 uses $\leq |V_2| \cdot 1.5863 + |V_3|$ many bits. This leads to a maximal number of advice bits per vertex at $\frac{|V_1|}{n} = 0.26986$, $\frac{|V_2|}{n} = 0.73014$ and thus to at most 1.1582196 bits per vertex. This leads to an upper bound of $1.1582196 \cdot n + d$ bits overall, for every 3-colorable graph of n vertices. $\square$

For giving the idea of the next algorithm, we analyze, for a chordal graph G, the graph G' which is obtained from G by edge contraction. This leads to some observations and lemmata.

Lemma 7. *Let G be a chordal graph with $\chi(G) = c$. For every graph G', which is obtained by contracting an edge of G, G' is chordal and $\chi(G') \leq c$.*

The proof is given in Appendix C. Now, we present the Algorithm 4 for coloring 3-colorable chordal graphs with 4 colors. For this, we separate the vertices $V(G)$ into two sets $A := V_1 \cup V_2$ and $B := V_3$. We know that, for every chordal graph G, the graph G_A restricted to the vertices in A is a forest (see Lemma 3).
The idea is to color each tree of G_A by a pair of colors $(i, 4)$, for some $i \in \{1,2,3\}$. The remaining vertices in V_3 will be colored using only colors $\{1,2,3\}$. We will show later that such a coloring always exists.
Before we can describe Algorithm 4, we have to move a few vertices inside A. It might happen that a vertex v from V_2 is revealed as the first one of a tree in G_A. This can occur when all its predecessors in $G^{\prec}$ are in V_3 (see v_7, v_8 in Figure 3). However, in this case, we note that v cannot have a neighbor in V_1. Such a neighbor w would be

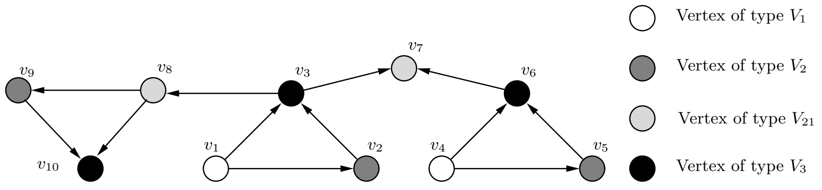


Fig. 3. Vertices of types V_{21}.

revealed after v, and consequently the edge orientation would be (v, w), an ingoing edge for w, thus $w \notin V_1$. Therefore, we can define

$$V_{21} = \{v \in V_2 | \mathit{Pred}(v) \subseteq V_3\},$$

and move V_{21} to V_1, more precisely $V_1' = V_1 \cup V_{21}$ and $V_2' = V_2 \setminus V_{21}$, while still preserving independence of V_1'.

Remark 1. V_1' is an independent set in G, respectively in $G^{\prec}$.

Looking again at Algorithm 2, we observe that all it needs from V_1 is that it is an independent set since all vertices from V_1, and only those, are colored by color 4. Consequently, the algorithm works the same when using V_1' instead of V_1. Let us call this variant Algorithm 2′.
Back to the new Algorithm 4. Here, V_1' contains exactly those vertices from A which are revealed without a predecessor from A, while for all vertices in V_2' such a predecessor exists. Consequently, the Algorithm 4 asks for every vertex $x \in V_1$ for two pieces of advice. First, it wants to know which pair of colors $(i, 4)$ shall be used to color the tree x belongs to. Secondly, it asks which of the two colors x gets itself.
There are three possible pairs of colors. This leads to a combination of a one-out-of-three and a one-out-of-two decision. Thus, at most 2.5863 bits are needed for every vertex from V_1.
With this information, obviously the algorithm is able to color all vertices $x \in V_1'$. Also, all vertices from V_2', can be colored, because at the moment a vertex v from V_2' is revealed, it has a predecessor w in A, and for w the color pair of the tree both belong to is known as well as the color w gets. Hence, v is colored by the other color from that pair without further advice. Inside the trees of G_A, such a coloring is clearly possible, but we still have to show later that this way a correct coloring of the whole graph develops.
Finally, for every vertex $z \in V_3$, which closes one or more triangles, the algorithm asks for a one-out-of-two decision, because such vertices have to be connected to at least two already colored and connected vertices (x, y), with different colors and so, in the worst case, there remain two possible colors for z ($|C_z| \leq 2$). The new algorithm needs at most $|V_1'| \cdot 2.5863 + |V_3| + \mathit{const}$ many bits.
To prove that such a coloring exists for every 3-colorable chordal graph, we give a further algorithm, which describes how an oracle can find the related coloring and with this the right advice tape. Again, we use $A = V_1 \cup V_2$ and $B = V_3$. We build the graph G' by subsuming every connected component of G_A in one vertex. When G is a 3-colorable chordal graph, the graph G' is 3-colorable as well (see Lemma 7). Thus, we use a 3-coloring c' of G'.

The 4-coloring c for G can be derived from c' in the following way. For a vertex $v' \in V(G')$, we distinguish two cases. If v' was constructed by an edge contraction, we color the contracted tree in G with the colors $\{c'(v'), 4\}$, and if v' corresponds directly to a vertex $v \in V(G)$ we define $c(v) := c(v')$. With this procedure, we get an coloring which satisfies the needed properties for Algorithm 4 in the appendix. With the corresponding advice tape, Algorithm 4 needs $\frac{|V_1|}{n} \cdot 2.5863 + \frac{|V_3|}{n}$ bits of advice per vertex. This leads us to the following lemma.

Lemma 8. *Let $G = (V, E)$ be a 3-colorable chordal graph and let $G^{\prec}$ be the corresponding input instance for OColA$_V$. There exists a coloring $c : V(G) \to \{1, 2, 3, 4\}$ and with this an advice tape such that Algorithm 4 can color $G^{\prec}$ with the coloring function c.*

The proof is given in Appendix D. Putting everything together, we can combine the previous algorithms into a final one, Algorithm 5, formally stated in the appendix. This algorithm uses the first two advice bits to decide which of the Algorithms 1, 2', 4 it shall use. Consequently, it always makes use of the best possible advice-per-vertex ratio among those three. This results in the following analysis.

Theorem 3. *Let $G = (V, E)$ be a 3-colorable chordal graph with $|V(G)| = n$ and let $G^{\prec}$ be the corresponding input instance for OColA$_V$. Algorithm 5 colors $G^{\prec}$ with 4 colors using at most $0,9865 \cdot n + 47$ advice bits.*

Proof: We have seen before that there exists an advice tape for any of the three Algorithms 1, 2', 4. Now, we show that, in any case, there is one of the three strategies that colors G using $0,9865 \cdot n + 47$ advice bits. Let $z_1 = \frac{|V_1|}{n}$, $z_1' = \frac{|V_1'|}{n}$, $z_2 = \frac{|V_2|}{n}$, $z_2' = \frac{|V_2'|}{n}$ and $z_3 = \frac{|V_3|}{n}$.
For the needed advice bits A_1 for Algorithm 1, A_2 for Algorithm 2', and A_4 for Algorithm 4, we have (with $z_1 + z_2 = z_1' + z_2'$, $z_1 \le z_1'$)

$$\begin{aligned}
A_1 &\le 1.5863 \cdot z_1 + z_2 \le 1.5863 \cdot z_1' + z_2' \\
A_2 &\le 1.5863 \cdot z_2' + z_3 \\
A_4 &\le 2.5863 \cdot z_1' + z_3
\end{aligned}$$

Additionally, $z_1' + z_2' + z_3 = 1$. The corresponding convex space has its maximum at $z_1' = 0.30667$, $z_2' = 0.5$, and $z_3 = 0.19333$, and there it needs $A_g = 0.98647 \cdot n$ bits. The additive constant 47 consists of the two bits read at the beginning and the usual 45 bits that can remain in each of the sub-algorithms from the block of bits read for three-way decisions. □

4 Conclusion

We introduced first research results for online coloring algorithms with advice for 3-colorable graphs. For planar, chordal and general 3-colorable graphs we presented nearly matching lower and upper bounds on the number of advice bits for the 3-coloring. We also gave 4-coloring online algorithms with advice for those graph classes (see Table 1). It remains to extend the lower bounds to the 4-coloring. The extension to other graph classes, k-coloring, and general coloring is also very interesting.

References

1. H.-J. Böckenhauer, D. Komm, R. Královič, and R. Královič. On the advice complexity of the k-server problem. In L. Aceto, M. Henzinger, and J. Sgall, editors, *Proc. of the 38th International Colloquium on Automata, Languages and Programming (ICALP 2011)*, volume 6755 of *LNCS*, pages 207–218. Springer-Verlag, 2011.
2. H.-J. Böckenhauer, D. Komm, R. Královič, R. Královič, and T. Mömke. On the advice complexity of online problems. In Y. Dong, D.-Z. Du, and O. H. Ibarra, editors, *Algorithms and Computation, 20th International Symposium, ISAAC 2009, Honolulu, Hawaii, USA, December 16-18, 2009. Proceedings*, volume 5878 of *LNCS*, pages 331–340. Springer-Verlag, 2009.
3. A. Borodin and R. El-Yaniv. *Online Computation and Competitive Analysis.* Cambridge University Press, 1998.
4. S. Dobrev, R. Královič, and D. Pardubská. How much information about the future is needed? In V. Geffert, J. Karhumäki, A. Bertoni, B. Preneel, P. Návrat, and M. Bieliková, editors, *Proc. of the 34th International Conference on Current Trends in Theory and Practice of Computer Science (SOFSEM 2008)*, volume 4910 of *LNCS*, pages 247–258, Berlin, 2008. Springer-Verlag.
5. Y. Emek, P. Fraigniaud, A. Korman, and A. Rosén. Online computation with advice. In S. Albers, A. Marchetti-Spaccamela, Y. Matias, S. E. Nikoletseas, and W. Thomas, editors, *Proc. of the 36th International Colloquium on Automata, Languages and Programming (ICALP 2009)*, volume 5555 of *LNCS*, pages 427–438. Springer-Verlag, 2009.
6. M. Forisek, L. Keller, and M. Steinová. Advice complexity of online coloring for paths. In *LATA*, pages 228–239, 2012.
7. M. R. Garey and D. S. Johnson. *Computers and Intractability.* W. H. Freeman and Co., San Francisco, Calif., 1979.
8. V. Guruswami and S. Khanna. On the hardness of 4-coloring a 3-colorable graph. In *Computational Complexity 2000. Proc. 15th Annual IEEE Conf.*, pages 188 –197, 2000.
9. J. Hromkovič. *Design and analysis of randomized algorithms.* Texts in Theoretical Computer Science. An EATCS Series. Springer-Verlag, Berlin, 2005.
10. J. Hromkovič, R. Královič, and R. Královič. Information complexity of online problems. In *MFCS*, volume 6281 of *LNCS*, pages 24–36. Springer, 2010.
11. S. Khanna, N. Linial, and S. Safra. On the hardness of approximating the chromatic number. In *Theory and Computing Systems, 1993., Proceedings of the 2nd Israel Symposium on the*, pages 250 –260, jun 1993.
12. H. Kierstead. Recursive and on-line graph coloring. In Y. L. Ershov, S. Goncharov, A. Nerode, J. Remmel, and V. Marek, editors, *Handbook of Recursive Mathematics Volume 2: Recursive Algebra, Analysis and Combinatorics*, volume 139 of *Studies in Logic and the Foundations of Mathematics*, pages 1233–1269. Elsevier, 1998.
13. H. Kierstead and W. Trotter. On-line graph coloring. In L. A. McGeoch and D. D. Sleator, editors, *On-line Algorithms*, volume 7 of *DIMACS Series in Discrete Mathematics and Theoretical Computer Science*, pages 85–92. AMS—DIMACS—ACM, 1992.
14. L. Lovász, M. E. Saks, and W. T. Trotter. An on-line graph coloring algorithm with sublinear performance ratio. *Discrete Mathematics*, 75(1–3):319–325, 1989.
15. S. Seibert, A. Sprock, and W. Unger. Advice complexity of the online vertex coloring problem. Technical Report 765, ETH Zürich, Department of Computer Science, ftp.inf.ethz.ch/pub/publications/tech-reports/7xx/765.pdf, 2012.
16. D. D. Sleator and R. E. Tarjan. Amortized efficiency of list update and paging rules. *Communications of the ACM*, 28(2):202–208, 1985.
17. S. Vishwanathan. Randomized online graph coloring. *Journal of Algorithms*, 13(4):657–669, 1992.

A Proof of Theorem 1

Proof: We prove the claim by constructing a class $\mathcal{G}_{hard}$ of hard input instances. For every 6-tuple of numbers $(u, v, w, x, y, z) \in \mathbb{N}^6$, we construct a graph G with $4 \cdot (u + v + w + x + y + z) + 9$ vertices (see Figure 4).
For an easier notation, we use $t = u + v + w + x + y + z$. At first, we describe the graph G for a given tuple (u, v, w, x, y, z). The graph G of an input instance in $\mathcal{G}_{hard}$ contains three paths P_1, P_2, P_3. The path $P_1 = v_{(1,1)}, v_{(1,2)}, \ldots, v_{(1,4u)}, v_{(1,4u+1)}, \ldots, v_{(1,4u+4v)}$ contains $4 \cdot (u + v)$ vertices, the path $P_2 = v_{(2,1)}, \ldots, v_{(2,4w+4x)}$ contains $4 \cdot (w + x)$ vertices and $P_3 = v_{(3,1)}, \ldots, v_{(3,4y+4z)}$ contains $4 \cdot (y + z)$ many vertices. Every path P_i, for $i \in \{1, 2, 3\}$, is, at the beginning, extended by one vertex v_{D_i} (see Figure 4). The three vertices v_{D_1}, v_{D_2} and v_{D_3} are connected to each other and form a triangle. Additionally, every vertex in a path P_i is connected to one of the two vertices $v_{(i,a)}, v_{(i,b)}$. The vertex $v_{(1,4u)}$ (resp. $v_{(2,4w)}, v_{(3,4y)}$) is connected to both vertices $v_{(1,a)}$ and $v_{(1,b)}$ (resp. $v_{(2,a)}$ and $v_{(2,b)}$, $v_{(3,a)}$ and $v_{(3,b)}$).
Additionally, the vertex $v_{(1,a)}$ (resp. $v_{(2,a)}, v_{(3,a)}$) is also connected to v_{D_1} and v_{D_2} (resp. v_{D_2}, v_{D_3} or v_{D_3}, v_{D_1}).
We can see that such a graph is maximal outerplanar, because the subgraph $G_{V(P_i^+)}$ which is induced by the vertices $V(P_i^+) = V(P_i) \cup \{v_{(i,a)}, v_{(i,b)}\}$ is maximal outerplanar on its own. The triangle $v_{D_1}, v_{D_2}, v_{D_3}$ is maximal outerplanar and the three extended paths $G_{V(P_i^+)}$ are connected to the triangle twice and planar, hence the whole graph is maximal outerplanar. In Figure 4, also an optimal coloring of G is shown. As in every maximal outerplanar graph, the coloring is unique up to permutations of the colors.
For counting the numbers of instances which need a different advice string, we separate the input into three phases. In the first phase, every second vertex $v_{(i,2)}, v_{(i,4)}, \ldots$ of every path P_i is revealed as an isolated vertex. In the second phase, when all $2t$ isolated vertices have been revealed, for every pair $(v_{(i,4\cdot c-2)}, v_{(i,4\cdot c)})$ for $c \in \{1, \ldots, u+v\}$ and $i \in \{1, 2, 3\}$, the vertex $v_{(i,4\cdot c-1)}$ connected to both is revealed. Overall, there are t such pairs of isolated vertices. In the last step, the remaining vertices connecting the already revealed subpaths of length 3, as well as the vertices $v_{(i,a)}, v_{(i,b)}$ and v_{D_i} are revealed. Additionally, between the first vertex $v_{(i,2)}$ of each path and the vertex of the middle triangle v_{D_i} the vertex $v_{(i,1)}$ is revealed to connect the path and the triangle. In the example in Figure 4, the vertices which are revealed as isolated vertices are marked by a square, and the vertices revealed in the second step are marked by a circle.
Now, we count the number of instances which need a different advice string, such that a deterministic algorithm can be guaranteed to be optimal.
The instance has $2(u + v + w + x + y + z) = 2t$ isolated vertices. In every optimal solution, $2(u + v)$ many vertices $(v_{(1,2)}, v_{(1,4)}, \ldots, v_{(1,4u+4v)})$ have to be colored with the same color, $2(w + x)$ isolated vertices have to be colored with the same color, but different to the first color, and $2(y + z)$ many isolated vertices have to be colored with the third color.
This isolated vertices build t pairs $(v_{(1,2)}, v_{(1,4)}), (v_{(1,6)}, v_{(1,8)}, \ldots)$, and each pair gets connected by a vertex $(v_{(1,3)}, v_{(1,5)}, \ldots)$. In this situation, there are $u + v$ pairs of

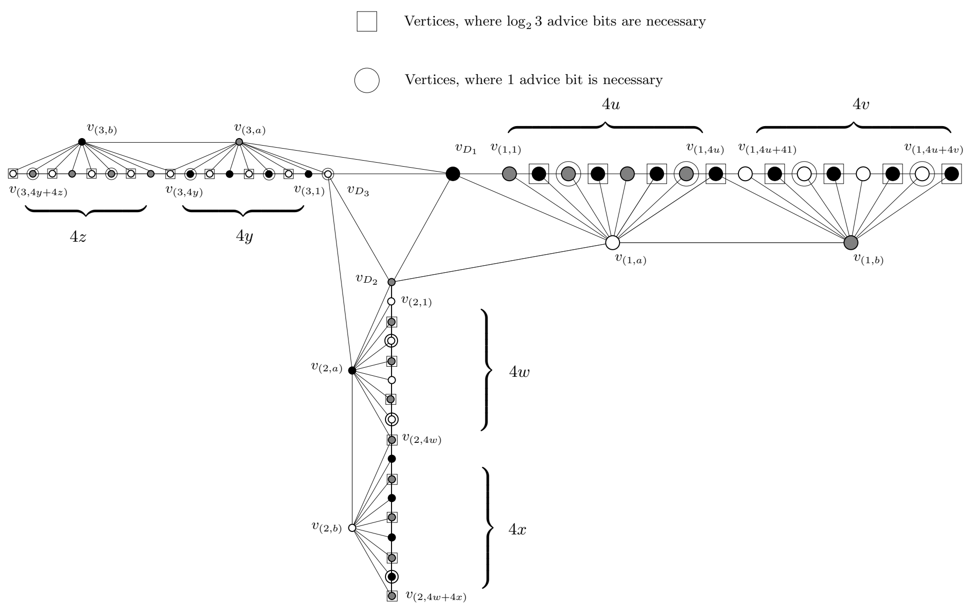


Fig. 4. Example of a hard instance for the parameters $u = v = w = x = y = z = 2$.

vertices in one color and u pairs become connected by vertices which have to get the same color and v pairs get connected by vertices, which have all to be colored in the other color. This coloring of the vertices in the middle is unique and determined by the connection to $v_{(i,a)}$ respectively $v_{(i,b)}$, but these two vertices are revealed later. This means that there are, in any possible input instance, $2t$ many isolated vertices and each vertex can be located in any of the three paths and thus requires exactly one of the three colors. This leads to

$$3^{2t} \text{ many possibilities.}$$

The optimal coloring is unique up to permutations of the colors. For the necessary advice, we have to calculate that 6 optimal colorings can use the same advice due to the 6 possibilities of renaming colors. Thus, we need $3^{2t} \cdot \frac{1}{6}$ different advice strings for an optimal coloring of the $2t$ isolated vertices.
The $2t$ isolated vertices are combined to t pairs (v_{1_2}, v_{1_4}),(v_{1_6}, v_{1_8}),.... There exist t different pairs of isolated vertices, $u + v$ pairs of vertices with color 1, $w + x$ pairs with color 2 and $y + z$ pairs of color 3.
In the second step, these t pairs get connected by one vertex in the middle each. For every vertex in the middle of a pair, there are two possible colors to choose from, but only one color leads to an optimal coloring of G. So there are $u + v$ pairs with color one, where, in u pairs, the vertex in the middle has to be colored with 2 and, in v pairs, the vertex in the middle has to be colored with 3 to obtain an optimal

coloring. The respective statements hold for the $w+x$ pairs colored with 2 and for the $y+z$ pairs colored with 3.
We now count the number of possible variations of the order in which the vertices in the middle of isolated vertices of color 1 are revealed. There are $u+v$ many pairs where, in the middle of u pairs, the revealed vertex has to be colored with color 2 and v revealed vertices which have to be colored with color 3. The two values u and v can be chosen arbitrarily and thus there exist

$$\sum_{i=0}^{u+v} \binom{u+v}{i} = 2^{u+v}$$

many different continuations for each instance from the first phase. Considering all groups of pairs, we get

$$2^{u+v} \cdot 2^{w+x} \cdot 2^{y+z} = 2^t$$

many different continuations for any input string of the first phase, which all need different advice strings. This leads to

$$3^{2t} \cdot 2^t$$

many different input strings for a graph from the class of graphs with $4t+9$ vertices overall.
For the $3^{2t} \cdot 2^t$ many different instances, their are at least $3^{2t} \cdot 2^t \cdot \frac{1}{6}$ many different advice strings necessary.
Now we show that, for an instance of $n = 4t+9$ vertices, at least $\frac{1}{2} \cdot \left(\log_2 3 + \frac{1}{2}\right) \cdot n - 12 > 1.0424 \cdot n - 12$ many advice bits are needed to be optimal. We have

$$\begin{aligned}
\log_2\left(3^{2t} \cdot 2^t \cdot \frac{1}{6}\right) &= \log_2\left(3^{2t}\right) + t + \log_2 \frac{1}{6} \\
&= 2t \cdot \log_2 3 + t - \log_2 6 \\
&= t \cdot (2\log_2 3 + 1) - \log_2 3 - 1 \\
&= \frac{n-9}{4} \cdot (2\log_2 3 + 1) - \log_2 3 - 1 \\
&= \frac{n}{2} \cdot \left(\log_2 3 + \frac{1}{2}\right) - \frac{9}{2}\log_2 3 - \frac{9}{4} - \log_2 3 - 1 \\
&= \frac{n}{2} \cdot \left(\log_2 3 + \frac{1}{2}\right) - \left(\frac{11}{2}\log_2 3 + \frac{13}{4}\right) \\
&> \frac{1}{2} \cdot n \cdot \left(\log_2 3 + \frac{1}{2}\right) - 11.9673 \\
&> \frac{1}{2} \cdot n \cdot \left(\log_2 3 + \frac{1}{2}\right) - 12 \\
&> 1.0424 \cdot n - 12.
\end{aligned}$$

□

B Proof of Theorem 2

Proof: We prove the claim by constructing a class $\mathcal{G}_{gen}$ of input instances. For every 3-tuple of numbers $(u, v, w) \in \mathbb{N}^3$, there is a graph G with $u+v+w+3$ vertices (see Figure 5).
For an easier notation, we use $k = u + v + w$. At first, we describe the graph G for a given tuple (u, v, w). The graph G of an input instance in $\mathcal{G}_{gen}$ contains three sets of vertices V_{c1}, V_{c2}, V_{c3}, with $|V_{c1}| = u, |V_{c2}| = v$, and $|V_{c3}| = w$ vertices, and three more vertices v_1, v_2, and v_3, which build a triangle. All vertices of V_{c1} are connected to v_2 and v_3, the vertices of V_{c2} are connected to v_1 and v_3, and the vertices in V_{c3} are connected to v_1 and v_2 (see Figure 5).

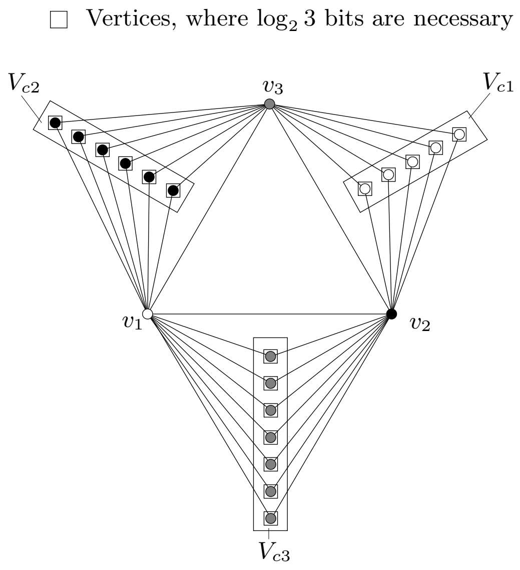


Fig. 5. Example of a hard instance for the tuple $(u, v, w) = (5, 6, 7)$.

For every instance $G \in \mathcal{G}_{gen}$, there is only one optimal coloring, except for renaming of the colors. For the three vertices v_1, v_2 and v_3 the coloring is unique, because they build a triangle and the color of all vertices in V_{ci} for $i \in \{1, 2, 3\}$ is determined by this.
For counting the numbers of instances which need a different advice string, we separate the input in two phases. In the first phase, every vertex from V_{c1}, V_{c2}, and V_{c3} is revealed as an isolated vertex.
In the last step, the vertex v_1 is revealed, connected to all vertices from V_{c2} and V_{c3}, the vertex v_2 is revealed, connected to v_1 and all vertices from V_{c1} and V_{c3}, and v_3 is revealed, connected to v_1, v_2 and all vertices from V_{c1} and V_{c2}. In the example in Figure 5, the vertices which are revealed as isolated vertices are marked by a square.

Now, we count the number of instances which need a different advice string, such that a deterministic algorithm can be guaranteed to be optimal.
In the first step, $u + v + w = k$ many isolated vertices are revealed, which can be colored in one of the possible colors $1, 2, 3$. Every isolated vertex has three possibilities to belong to one of the three sets V_{ci}. The variables u, v, w can get arbitrary values with the constraint $u + v + w = k$. This leads to

$$3^k$$

many possible input strings. We have k many isolated vertices and there are three colors possible for every position.
In the second step, there are 6 possibilities for revealing the last three vertices, but here, no advice is needed, because, for an optimal coloring, the colors of the three vertices v_1, v_2 and v_3 are determined by the edges given when the vertices are revealed. On the other hand, we have to calculate that the 6 optimal colorings, coming up by color renaming, can use the same advice.
Thus, we need $3^k \cdot \frac{1}{6}$ different advice strings for an optimal coloring of the k isolated vertices. We have

$$\begin{aligned}\log_2\left(3^k \cdot \frac{1}{6}\right) &= k \cdot \log_2 3 + \log_2\left(\frac{1}{6}\right)\\ &= (k-1) \cdot \log_2 3 - 1\\ &= (n-4) \cdot \log_2 3 - 1.\end{aligned}$$

□

C Proof of Lemma 7

Proof: Assume that G is a chordal graph with $\chi(G) = c$ and G' is obtained by contracting the edge $\{a, b\}$ in G. Assume that G' contains a vertex-induced cycle of length > 3 containing x, where x is the vertex contracted from edge $\{a, b\}$. Let be $x, v, w, \cdots, z, x$ this cycle, then either $a, v, w, \cdots, z, a$ is also a cycle of length > 3 in G or $a, v, w, \cdots, z, b, a$ is a cycle of length > 4 in G. Both alternatives are a contradiction to our assumption. It follows that G' is chordal as well.
Now we show that $\chi(G') \leq \chi(G)$. Assume $\chi(G') > \chi(G)$. We have that x is in a clique $C = \{x, v_i, v_2, \cdots, v_c\}$ of size $> \chi(G)$. But the clique size of $\{a, v_i, v_2, \cdots, v_c\}$ and $\{b, v_i, v_2, \cdots, v_c\}$ is at most $\chi(G)$. Thus there exists i, j with $\{a, v_i\} \notin E(G)$ and $\{b, v_j\} \notin E(G)$. If $i = j$ hold, then C would not be a clique of size $> \chi(G)$. Thus a, v_i, v_j, b is a vertex-induced cycle of length 4 in G, which is a contradiction. □

D Proof of Lemma 8

For proving the claim, we give the algorithm to find such a coloring and prove that, for every 3-colorable chordal graph, such a coloring will be found.

Proof: Let $G^{\prec}$ be the given input instance. From the order of the vertices, we can build three sets of vertices V_1, V_2 and V_3. We build from this two sets $A = V_1 \cup V_2$ and $B = V_3$. We know that the graph G_A is a forest (see Lemma 3). Each tree of G_A is two-colorable. In the following, we will color each tree of G_A with two colors, where one of the two colors will be 4 for all trees. To find the corresponding color for each tree, we have to determine a coloring, which fits with a coloring for the vertices in B.
For finding such a coloring, we build the graph G' out of G by contraction of edges between vertices of A. This leads us to a surjective function $m : V(G) \to V(G')$. If $T_i \subset A$ is a tree in G_A, then for all $t_j, t_k \in V(T_i), j \neq k$, we have $m(t_j) = m(t_k)$.
We know that G' is 3-colorable and chordal as well as G (see Lemma 7), so we can find an optimal coloring c' for G' with three colors, because G is a 3-colorable chordal graph.
We extend the coloring function c' to c for G by coloring all vertices $v \in B \subset V(G)$ which are also vertices in G' with $c(v) := c'(v)$. For all vertices $w_i \in A \subset V(G)$ which are represented by one vertex $x \in V(G')$, with $m(w_i) = x$ we color the corresponding tree alternatingly in $(c'(x), 4)$.
It is obvious that the coloring c needs at most 4 colors because c' was a coloring with 3 colors and the color 4 is used additionally. The new coloring is proper for G because no two vertices with color 4 are connected to each other, and each tree T_i in G_A is colored properly with the two colors $(c'(x), 4)$. If the vertex x represents a tree T_x in G_A, it follows that all vertices in G which are connected to the tree T_x represented by x are connected to x in G'. If c' is a proper coloring for G', it follows that $c'(y) \neq c'(x)$ for all vertices $y \in \mathit{Neigh}_{G'}(x)$. If now the tree which is represented by x in G' is colored only with $c(x)$ and 4, it follows that c is a proper coloring. $\Box$

E Algorithms

Algorithm 1 Optimal online 3-coloring

```
Input: Online input instance $G^{\prec}$.
 1: for every revealed vertex $v$ do
 2:   if $v$ is the first revealed vertex then
 3:     Define $c(v) := 1$
 4:   else if $v$ is the second revealed vertex then
 5:     if $v$ is connected to $v_1$ then
 6:       Define $c(v) := 2$
 7:     else
 8:       Read one bit $b$ from the advice tape
 9:       if $b = 0$ then
10:         Define $c(v) := 1$
11:       else
12:         Define $c(v) := 2$
13:       end if
14:     end if
15:     Decide of which type $v$ is
16:     if $v \in V_1$ then
17:       Ask for a one-out-of-three decision from the advice tape $c_v \in \{1, 2, 3\}$ and define
          $c(v) := c_v$ ($< 1.5863$ bits)
18:     else if $v \in V_2$ then
19:       Determine the set *Pred(v)* and the remaining colors $C_v$
20:       if $|C_v| = 1$ then
21:         define $c(v) := c_v \in C_v$
22:       else $\{(|C_v| = 2)\}$
23:         Ask for a one-out-of-two decision from the advice tape $c_v \in C_v$ and define
            $c(v) := c_v$ (1 bit)
24:       end if
25:     else $\{(v \in V_3)\}$
26:       Determine the remaining color $c_v \in C_v$ and define $c(v) := c_v$.
27:     end if
28:   end if
29: end for
Output: The coloring function $c$
```

Algorithm 2 Online 4-coloring

```
Input: Online input instance $G^{\prec}$, with $\chi(G) = 3$.
 1: for every revealed vertex $v \in V(G)$ do
 2:     if $v \in V_1$ {isolated vertex} then
 3:         Define $c(v) := 4$
 4:     else if $v \in V_2$ {connected to at least one already colored vertex} then
 5:         Define $C_v$ to be the set of possible colors for $v$ in a 4-coloring
 6:         if $|C_v| = 1$ {only one color remains} then
 7:             Define $c(v) := c_v \in C(V)$
 8:         else if $|C_v| = 2$ {two of the colors $\{1, 2, 3\}$ are possible} then
 9:             Ask for a one-of-two decision from the advice tape $c_v \in C_v$ and define $c(v) := c_v$
                {1 bit}
10:         else
11:             Ask for a one-of-three decision from the advice tape $c_v \in \{1, 2, 3\}$ and define
                $c(v) := c_v$ {$< 1.5863$ bits}
12:         end if
13:     else {$v \in V_3$}
14:         Define $C_v$ to be the set of possible colors for $v$ in a 4-coloring
15:         if $|C_v| = 1$ then
16:             Define $c(v) := c_v \in C(V)$
17:         else {$|C_v| = 2$}
18:             Ask for an one-of-two decision $c_v \in C_v$ and define $c(v) := c_v$ {1 bit}
19:         end if
20:     end if
21: end for
Output: The coloring function $c$
```

Algorithm 2' results from Algorithm 2 by substituting V_1' and V_2' for V_1, and V_2 respectively.

Algorithm 3 Online graph coloring

```
Input: Online input instance $G^{\prec}$, with $\chi(G) = 3$.
 1: Read the first bit $b_1$ of the advice tape
 2: if $b_1 = 0$ then
 3:     Use Algorithm 1
 4: else
 5:     Use Algorithm 2
 6: end if
Output: The coloring function $c$
```

Algorithm 4 Online chordal graph coloring

```
Input: Online input instance $G^{\prec}$, with $\chi(G) = 3$.
 1: for every revealed vertex $v \in V(G)$ do
 2:   if $x \in V_1'$ {$x$ has only predecessors in $V_3$} then
 3:     Ask from the advice tape for a one-out-of-three decision, by which pair $p_x \in$
        $\{(1,4),(2,4),(3,4)\}$ the tree containg $x$ gets colored, and for a one-out-of-two deci-
        sion, by which color $c_x \in p_x$ this vertex gets colored, and define $c(x) := c_x$ (2.5863
        bits).
 4:   else if $y \in V_2'$ {connected to at least one revealed vertex $x \in V_1'$} then
 5:     Identify the tree $x$ belongs to and the pair $p(x)$ for this tree from $Pred(x)$.
 6:     Define $C_y \in p_x \setminus c(x)$, and define $p_y := p_x$.
 7:   else {$z \in V_3$}
 8:     Define $C_z$ {Set of possible colors for $z$ in a 4-coloring without color 4.}
 9:     if $|C_z = 2|$ then
10:       Ask for an one-of-two decision $c_z \in C_z$ and define $c(z) := c_z$ {1 bit}
11:     else {$|C_z = 1$}
12:       Define $c(z) := c_z \in C_z$ {0 bit}
13:     end if
14:   end if
15: end for
Output: The coloring function $c$
```

Algorithm 5 Online graph coloring

```
Input: Online input instance $G^{\prec}$, with $\chi(G) = 3$.
 1: Read the first two bits $b_1, b_2$ of the advice tape
 2: if $b_1 = 1$ then
 3:   Use Algorithm 1
 4: else if $b_1 = 0, b_2 = 1$ then
 5:   Use Algorithm 2
 6: else
 7:   Use Algorithm 4
 8: end if
Output: The coloring function $c$
```