Optimal Online Edge Coloring of Planar Graphs with Advice

Jesper W. Mikkelsen
jesperwm@imada.sdu.dk

Department of Mathematics and Computer Science
University of Southern Denmark

Abstract. We study the amount of knowledge about the future that an online algorithm needs to color the edges of a graph optimally (i.e. using as few colors as possible). Assume that along with each edge, the online algorithm receives a fixed number of advice bits. For graphs of maximum degree $\Delta$, it follows from Vizing’s Theorem that $\lceil \log(\Delta + 1) \rceil$ bits per edge suffice to achieve optimality. We show that even for bipartite graphs, $\Omega(\log \Delta)$ bits per edge are in fact necessary. However, we also show that there is an online algorithm which can color the edges of a $d$-degenerate graph optimally using $O(\log d)$ bits of advice per edge. It follows that for planar graphs and other graph classes of bounded degeneracy, only $O(1)$ bits per edge are needed, independently of how large $\Delta$ is.

1 Introduction

An edge coloring of a graph is an assignment of colors to the edges of the graph such that no two adjacent edges share the same color. Many scheduling and assignment problems can be modeled as edge coloring problems. The online edge coloring problem, which we refer to simply as EDGE-COLORING, was introduced by Bar-Noy et al. [1]. In this problem, the edges of a graph arrive one by one. The edges are specified by their endpoints, but the vertices of the graph are not known in advance. Each edge must be assigned a color before the next edge arrives, under the constraint that no two adjacent edges are assigned the same color. The color assigned to an edge cannot be changed later on. The goal is to use as few colors as possible.

Traditionally, worst-case competitive analysis [16,23] is used to measure the performance of an online algorithm. The solution produced by the online algorithm is compared to that of an optimal offline algorithm, $OPT$, which knows the entire input in advance. In [1] it is shown that any EDGE-COLORING algorithm, which never introduces a new color unless forced to do so, is 2-competitive and that no online algorithm, even if we allow randomization and restrict the input graph to being a tree, can achieve a better competitive ratio.

The underlying assumption of competitive analysis, that nothing is known about future parts of the input, is sometimes unrealistic. Therefore, for many online problems, various relaxations of this assumption have been suggested, including look-ahead [3], locality of reference [7] and several models where the input is generated from some known probability distribution [17,19,21]. In this paper, we consider the recent idea of advice complexity [5,11,12,14]. Advice complexity provides a quantitative and problem-independent approach for relaxing the online constraint by providing the algorithm partial knowledge of the future. Our main goal in applying the framework of advice complexity to EDGE-COLORING is to better understand the online hardness of the problem. How much (and which kind of) information about the future are we lacking in order to produce an optimal edge coloring in the online setting?

* Supported in part by the Villum Foundation and the Danish Council for Independent Research, Natural Sciences.
1.1 Advice complexity model

We study deterministic algorithms in the advice-with-request model of Emek et al. [12]. In this model, an online algorithm, ALG, receives some fixed number, $b$, of advice bits along with each request. These advice bits are provided by an oracle which has unlimited computational power and knows the entire input. For EDGE-COLORING, the input is a sequence of edges $(e_1, \ldots, e_m)$. When the edge $e_i$ arrives, the algorithm receives some advice $b_i \in \{0, 1\}^b$ from the oracle. The algorithm then decides which color to assign to $e_i$ based on the edges $e_1, \ldots, e_i$ that have been revealed up until now and the advice $b_1, \ldots, b_i$ received thus far. We refer to [12] for a formal definition and an in-depth discussion of the model. In section 4 we discuss the possibility of using other models of advice complexity.

In this paper, we are only concerned with how much advice is needed in order to produce an optimal solution. For problems where the number, $r$, of possible responses for each request is finite, there is a simple approach for obtaining an optimal solution: For each request, the oracle specifies the exact response of OPT. This requires $\lceil \log r \rceil$ bits per request. We say that $\lceil \log r \rceil$ is the trivial upper bound on the advice complexity. For many problems, it has been shown that the trivial upper bound is asymptotically tight, i.e., at least $\Omega(\log r)$ bits per request are required to be optimal. This includes online problems such as bin packing [8], $k$-server [4], metrical task systems [12], vertex coloring [13], various scheduling problems [22], knapsack [6], bipartite matching [20], graph exploration [10], independent set [9] and minimum Steiner tree [2]. One notable example where the trivial upper bound is known not to be asymptotically tight is the paging problem [5].

1.2 Preliminaries

All graphs considered are simple. We denote the number of edges in a graph by $m$, the number of vertices by $n$ and the maximum degree by $\Delta$. A graph $G$ is $k$-edge-colorable if there exists an edge coloring of $G$ with at most $k$ different colors. The chromatic index $\chi'(G)$ of $G$ is the smallest integer $k$ such that $G$ is $k$-edge-colorable. We assume that colors are represented by consecutive positive integers. For a bipartite graph $G$, we write $G = (L, R)$ if $L$ and $R$ form a bipartition of the vertices of $G$. We let $K_{a,b}$ denote the complete bipartite graph $G = (L, R)$ where $|L| = a$ and $|R| = b$.

In addition to bipartite graphs, we consider trees, planar graphs and, more generally, $d$-degenerate graphs. A graph is $d$-degenerate if there is an ordering $v_1, v_2, \ldots, v_n$ of its vertices such that, for $1 \leq i \leq n$, the vertex $v_i$ is adjacent to at most $d$ vertices in $\{v_1, \ldots, v_{i-1}\}$. The degeneracy of a graph $G$ is the least integer $d$ such that $G$ is $d$-degenerate. An edge $e = (v_i, v_k)$ where $i < k$ is said to be a front-edge at $v_i$ and a back-edge at $v_k$. Furthermore, $d_f(v_i)$ is the number of front-edges at $v_i$.

The notion of degeneracy has appeared under other names and many equivalent definitions exist (see e.g. [15]). Note that the degeneracy is at most $\Delta$. A graph is 1-degenerate if and only if it is a forest. Planar graphs are 5-degenerate. Other graph classes of bounded degeneracy include graphs of bounded genus, bounded tree-width, and graphs excluding a fixed minor.

It is clear that $\Delta \leq \chi'(G)$ for any graph $G$. The celebrated Vizing’s Theorem [25] states that $\chi'(G) \in \{\Delta, \Delta + 1\}$. The following relationship between edge coloring and degeneracy, which is also due to Vizing, will be used extensively in the design of our algorithm.

**Theorem 1 (Vizing [24,26]).** Let $G$ be a $d$-degenerate graph of maximum degree $\Delta$. If $\Delta \geq 2d$, then $\Delta$ colors suffice for edge-coloring $G$. 
1.3 Our contribution

By Vizing’s theorem, the trivial upper bound on the advice complexity of edge-coloring is \(\lceil \log(\chi'(G)) \rceil = O(\log \Delta)\) bits per edge. We show that for bipartite graphs, the trivial upper bound is asymptotically tight in the advice-with-request model. On the other hand, we show that if the input graph is \(d\)-degenerate, then there exists an optimal online algorithm which receives only \(O(\log d)\) bits of advice per edge. In particular, only a constant number of advice bits per edge is needed to optimally color the edges of a planar graph online. Thus, there are graph classes where the trivial upper bound is asymptotically tight, but for many other interesting graph classes, there is a large gap between the actual advice complexity and the trivial upper bound.

2 A lower bound for bipartite graphs

It is a well-known result of König [18] that bipartite graphs are \(\Delta\)-edge-colorable. In order to prove the lower bound on the advice complexity for bipartite graphs, we need the following gadget. The gadget can be used to ensure that two edges cannot receive different colors in an optimal edge coloring.

Definition 1. Let \(n \geq 2\). The graph \(H_n\) consists of a complete bipartite graph \(K_{n,n} = (L, R)\) together with vertices \(v_l, v_r\) and edges \(\{(v_l, v) : v \in L\}, \{(v_r, v) : v \in R\}\). The vertex \(v_l\) (\(v_r\)) is denoted the leftmost (rightmost) vertex.

We say that two edges \(e_l = (x_l, y_l)\) and \(e_r = (x_r, y_r)\) are connected by an \(H_n\) if \(y_l\) is the leftmost vertex and \(y_r\) is the rightmost vertex of the same \(H_n\) and neither \(x_l\) nor \(x_r\) is part of that \(H_n\). See Figure 1.

\[\text{Fig. 1. } G_4: \text{ Two edges } e_l \text{ and } e_r \text{ (dashed lines) connected by an } H_4 \text{ (solid lines).}\]

Lemma 1. For \(n \geq 2\), let \(G_n\) be the graph consisting of two edges, \(e_l\) and \(e_r\), connected by an \(H_n\). Then, \(G_n\) is \((n+1)\)-edge-colorable. On the other hand, an edge coloring of \(G_n\) in which \(e_l\) and \(e_r\) are assigned different colors must use at least \(n+2\) colors.

Proof. \(G_n\) can be edge colored using \(n+1\) colors since it is a bipartite graph of maximum degree \(n+1\). Let \(C\) be an edge coloring of \(G_n\) such that \(C(e_l) \neq C(e_r)\), where \(C(e)\) is the color assigned to the edge \(e\). Suppose, by way of contradiction, that only \(n+1\) different colors are used in \(C\). Since \(e_l\) and \(e_r\) are colored differently, the set of colors used for edges between \(v_l\) and \(L\) cannot be identical to the set of colors used for edges between \(v_r\) and \(R\), since this would contradict that \(C\) uses only \(n+1\) colors. Thus, there exists a color, \(c\), such that there is an edge \(e = (v_l, v), v \in L\) colored with the color \(c\), while no edge between \(v_r\) and \(R\) is colored with the color \(c\). It follows that for each \(u \in R\), there must be an edge between \(u\) and a vertex
in $L$ colored with the color $c$, since $u$ has degree $n + 1$, $C$ uses $n + 1$ colors and the edge $(v_x, u)$ is not colored with the color $c$. In particular, since $|L| = |R|$, there must be an edge from a vertex in $R$ to $v$ colored with the color $c$. This is a contradiction, since $(v_i, v)$ is also colored with the color $c$.

**Theorem 2.** An optimal algorithm for edge-coloring must use at least $\Omega(\log \Delta)$ bits of advice per edge, even for bipartite graphs, where $\Delta$ is the maximum degree of the input graph.

**Proof.** At the beginning, the adversary reveals two stars $K_{1, \Delta}$. Let $\{x_1, \ldots, x_\Delta\}$ and $\{y_1, \ldots, y_\Delta\}$ be the vertices of degree 1 in each of these two stars, and let $x$ and $y$ be the center vertices. Furthermore, let $t$ be the time step right after both stars have been revealed. At time $t$, the adversary picks a permutation $\pi$ of $\{1, 2, \ldots, \Delta\}$. For each $1 \leq i \leq \Delta$, the edge $(x_i, x_i)$ is then connected to the edge $(y_i, y_{\pi(i)})$ by an $H_{\Delta - 1}$ through $x_i$ and $y_{\pi(i)}$. Since the resulting graph is bipartite and has maximum degree $\Delta$, it can be colored using $\Delta$ colors.

Let $\text{ALG}$ be an algorithm such that, at time $t$, the total number of advice bits received by $\text{ALG}$ is strictly less than $\log(\Delta!)$! We claim that $\text{ALG}$ cannot be optimal. Note that the adversary has $\Delta!$ different permutations to choose from. This implies that there must exist two different permutations $\pi, \pi'$ such that up until time $t$, the algorithm receives exactly the same bits of advice for both of these permutations. Thus, $\text{ALG}$ produces the same coloring, $C$, of the two stars no matter which of $\pi$ and $\pi'$ the adversary chooses to use. Let $C(u, v)$ be the color assigned to the edge $e = (u, v)$ in $C$. Fix $i$ such that $\pi(i) \neq \pi'(i)$. Because the edges are adjacent, $C(y_i, y_{\pi(i)}) \neq C(y_i, y_{\pi'(i)})$. Since $C(x, x_i)$ cannot be the same as both $C(y, y_{\pi(i)})$ and $C(y, y_{\pi'(i)})$, we may assume without loss of generality that $C(x, x_i) \neq C(y, y_{\pi(i)})$. By Lemma[1] this implies that $\text{ALG}$ will use at least $\Delta + 1$ colors when the adversary chooses the permutation $\pi$. We conclude that an optimal algorithm must have received at least $\log(\Delta!)$ bits of advice at time $t$. Since only $2\Delta$ edges are revealed before time $t$, this is only possible if the algorithm receives at least $\frac{\log(\Delta!)}{2\Delta} = \Omega(\log \Delta)$ bits of advice per edge.

For the adversary graph used in Theorem 2, the number, $m$, of edges is $m = O(\Delta^3)$. Thus, we may restate the lower bound of $\Omega(\log \Delta)$ in terms of $m$ and get a lower bound of $\Omega(\log(m^{1/3})) = \Omega(\log m)$. This shows that even if we insist on measuring the amount of advice as a function of $m$ (instead of $\Delta$), the trivial upper bound (of $\lceil \log m \rceil$) on the advice complexity is still asymptotically tight. Furthermore, the graph is $\Delta$-regular. It follows that the degeneracy $d$ of the graph is $d = \Delta$. Thus, the lower bound may also be stated in terms of the degeneracy as $\Omega(\log d)$. In the next section, we will give an algorithm which matches this lower bound.

3 An algorithm for $d$-degenerate graphs

In this section, we present the algorithm for $d$-degenerate graphs.

**Theorem 3.** Let $d \in \mathbb{N}$. For the class of $d$-degenerate graphs, there exists an edge-coloring algorithm which always produces an optimal coloring and uses

$$1 + \lceil \log(2d) \rceil + \lceil \log(d + 1) \rceil = O(\log d)$$

bits of advice per edge.
Theorem 3 assumes that the degeneracy of the input graph is at most \( d \), where \( d \) is a constant hard-wired into both the algorithm and the oracle. In Theorem 1 we show how the assumption that \( d \) is constant can be removed by communicating \( d \) as part of the advice.

In order to prove Theorem 3 we start by assuming that \( 2d \) divides the maximum degree, \( \Delta \), of the input graph. Later on, we will show how to (easily) reduce the general case to this special case.

Let \( G = (V, E) \) be a \( d \)-degenerate input graph of maximum degree \( \Delta \) and let \( a = \frac{\Delta}{2d} \in \mathbb{N} \). We will first give a high-level description of the oracle and the corresponding algorithm. The main idea is to partition the edges \( E \) into disjoint subsets, \( E_1, \ldots, E_n \), such that the maximum degree of the graph \( (V, E_j) \) is \( 2d \) for \( 1 \leq j \leq a \). By Theorem 1 \( (E, V) \) is \( 2d \)-edge-colorable. Thus, using an additional \( O(\log d) \) bits of advice per edge, the algorithm can make an optimal edge coloring of each \( E_j \) and, because of how the partition is created, all of \( G \).

However, the oracle cannot afford to compute such a partition and then simply encode the index \( j \) such that \( e \in E_j \) for each edge, since this would require too much advice per edge if \( a \) is large. Instead, the oracle finds a specific partition which is based on the arrival time of the edges and the fact that \( G \) is \( d \)-degenerate. This partition is such that when an edge \( e \) is revealed, the algorithm itself can (without advice) compute a small set of indices which always contains the correct index \( j \). This makes it possible to reduce the number of advice bits needed for the algorithm to learn the index \( j \).

In order to produce this partition, the oracle orders the vertices of the \( d \)-degenerate input graph such that no vertex has more than \( d \) back-edges. Starting with the first vertex in this ordering, the oracle processes the front-edges of each vertex ordered by (increasing) time of arrival. For each edge, the oracle determines the smallest index \( j' \) such that the edge can be assigned to \( E_{j'} \) while maintaining that \( (V, E_{j'}) \) has maximum degree at most \( 2d \). Note that whenever a front-edge, \( e \), at \( v \) is begin processed, the oracle has already assigned all back-edges of \( v \) to some sets in the partition. Since these back-edges may arrive later than \( e \), they may be unknown to the algorithm at the time when \( e \) is revealed. Therefore, the advice for the front-edge \( e \) will warn the algorithm not to assign \( e \) to \( E_j \) if this would later on prevent the intended assignment of some back-edge at \( v \) to \( E_j \).

### 3.1 The oracle for the case where \( 2d \) divides \( \Delta \)

We now give a formal description of the oracle. To each edge \( e \in E \), the oracle associates a bit string, \( B(e) \), of length \( \lceil \log(2d) \rceil + \lceil \log(d + 1) \rceil \) by following Procedure 1 (on page 6).

In order to prove the correctness of Procedure 1 we introduce the following terminology: At any point during the execution of Procedure 1 we say that an edge can legally be assigned to a subset \( E_j \) if this assignment does not make the maximum degree of \( (V, E_j) \) larger than \( 2d \). Also, we let \( \mathcal{P} = \{E_1, \ldots, E_n\} \). We will show that the index \( j' \) in line 12 is such that \( e \) can legally be assigned to \( E_{j'} \) and that the number in line 12 can be encoded using \( \lceil \log(d + 1) \rceil \) bits.

**Lemma 2.** Suppose that during the execution of Procedure 1 the second for-loop has just been entered and that \( e = e_s \). Let \( j' \) be the smallest index such that \( |E_{j'} \cap E(v_i)| \leq 2d - 1 \). Then, \( e \) can legally be assigned to \( E_{j'} \). Furthermore, \( j' \) is among the \( d + 1 \) smallest indices in \( J(e) \).

**Proof.** Assume that \( e = (v_i, v_k) \) is a front-edge at \( v_i \) and a back-edge at \( v_k \). Because \( i < k \), none of the front-edges at \( v_k \) has yet been assigned to any subset in \( \mathcal{P} \). Since \( v_k \) has at most
Procedure 1 Constructing the advice in the case where $2d$ divides $\Delta$.

**Input:** A $d$-degenerate graph $G = (V, E)$ of maximum degree $\Delta$ where $a \cdot 2d = \Delta$ for some $a \in \mathbb{N}$, together with arrival times of the edges.

**Output:** A bit string $B(e)$ of length $\lceil \log 2d \rceil + \lceil \log(d + 1) \rceil$ for each edge $e \in E$.

1. $E_j \leftarrow \emptyset$ for $1 \leq j \leq a$  
2. Compute an ordering $\{v_1, \ldots, v_n\}$ of the vertices of $G$ such that, for $1 \leq i \leq n$, the vertex $v_i$ is adjacent to at most $d$ vertices in $\{v_1, \ldots, v_{i-1}\}$.
3. Let $E(v_i)$ denote the edges incident to $v_i \in V$.
4. $\text{Prev}(e, v_i) \leftarrow \{f \in E(v_i) : f \text{ arrives before } e\}$ for $e \in E, v_i \in V$.
5. for $i = 1$ to $n$ do
6. Let $\{e_1, \ldots, e_{d_j(v_i)}\}$ be the front-edges at $v_i$, ordered by time of arrival.
7. for $s = 1$ to $d_j(v_i)$ do
8. $e \leftarrow e_s$
9. $J(e) \leftarrow \{j : |E_j \cap \text{Prev}(e, v_i)| \leq 2d - 1\}$
10. Let $j'$ be the smallest index such that $|E_{j'} \cap E(v_i)| \leq 2d - 1$.
11. $E_{j'} \leftarrow E_{j'} \cup \{e\}$
12. Use the last $\lceil \log(d + 1) \rceil$ bits of $B(e)$ to encode the number $|\{j \in J(e) : j < j'\}|$.
13. Compute $2d$-edge-colorings $C_j$ of $(V, E_j)$ for all $1 \leq j \leq a$.
14. For each edge $e \in E$, use the first $\lceil \log(2d) \rceil$ bits of $B(e)$ to encode the color assigned to $e$ in $C_j$, where $j$ is such that $e \in E_j$.

$d$ back-edges (including $e$), it follows that no subset in $\mathcal{P}$ currently contains more than $d - 1$ edges incident to $v_i$. Thus, if $e$ cannot legally be assigned to some subset $E_j$, this can only be because it would violate the degree constraint at $v_i$. That is, $e$ can be legally assigned to $E_j$ if and only if $|E_j \cap E(v_i)| \leq 2d - 1$. Since at most $\Delta - 1 = a \cdot 2d - 1$ edges incident to $v_i$ have arrived earlier than $e$, and since there are $a$ subsets in $\mathcal{P}$, this implies that $e$ can legally be assigned to at least one subset in $\mathcal{P}$.

Let $j'$ be the smallest index such that $e$ can legally be assigned to $E_{j'}$. Clearly, $j' \in J(e)$ since $\text{Prev}(e, v_i) \subseteq E(v_i)$. We will show that $j'$ is in fact among the $d + 1$ smallest indices in $J(e)$. Let $j \in J(e)$. By definition of $J(e)$, the number of edges in $E_j$ which are incident to $v_i$ and arrive before $e$ is at most $2d - 1$. Thus, if $e$ cannot legally be assigned to $E_j$, then there must be an edge $f \in E_j$ which is incident to $v_i$ but arrives later than $e$. The front-edges at $v_i$ arriving later than $e$ have not yet been assigned to any subset in $\mathcal{P}$, and so $f$ must be a back-edge at $v_i$. Since $v_i$ has at most $d$ back-edges, there can be at most $d$ indices $j \in J(e)$ such that $e$ cannot legally be assigned to $E_j$. It follows that $j'$ must be among the $d + 1$ smallest indices in $J(e)$.

Combining the assumption that $G$ has maximum degree $\Delta = a2d$ with Lemma 2 shows that Procedure 1 constructs a partition $E_1, \ldots, E_n$ of $E$ such that the maximum degree of $(V, E_i)$ is $2d$, for $1 \leq i \leq n$. Furthermore, the number in line 12 is at most $d + 1$ (and non-negative), and hence it can be encoded in binary using $\lceil \log(d + 1) \rceil$ bits. It follows from Theorem 1 that each of the graphs $(V, E_i)$ can be edge colored using $2d$ colors since they all have maximum degree $2d$ and are $d$-degenerate (because they are subgraphs of $G$ which is $d$-degenerate). This proves the correctness of Procedure 1.
3.2 The algorithm for the case where $2d$ divides $\Delta$

We now describe how the algorithm, $\text{ALG}$, uses the advice provided by the oracle. Note that when an edge $e$ arrives, $\text{ALG}$ is able to compute the set of indices $J(e)$ as defined in line 5 of Procedure [1], since $J(e)$ only depends on $d$ and the edges that have arrived earlier than $e$. Thus, $\text{ALG}$ can compute the index $j'$ such that $e$ was assigned to $E_{j'}$ by Procedure [1] by learning the number $|\{j \in J(e) : j < j'\}|$ from the last $\lceil \log(d+1) \rceil$ bits of $B(e)$. The algorithm (internally) assigns $e$ to $E_{j'}$. Then, $\text{ALG}$ reads the integer, $c$, encoded by the first $\lceil \log(2d) \rceil$ bits of $B(e)$ and colors $e$ with the color $(j' - 1)2d + c$.

It follows that for all $1 \leq j \leq a$, the algorithm colors the edges of $E_j$ with the colors $(j - 1)2d + 1, \ldots, j2d$ and produces a coloring of $E_j$ which is equivalent to the coloring $\mathcal{C}_j$ computed by the oracle. Thus, $\text{ALG}$ produces an optimal edge coloring of $G$.

3.3 The general case

Using the algorithm for the case where $2d$ divides $\Delta$ as a subroutine, we are now ready to prove Theorem [3].

**Proof of Theorem [3].** Fix $d \in \mathbb{N}$. We will describe an algorithm, $\text{ALG}$, and an oracle, $\phi$, satisfying the conditions of the theorem. Let $G = (V, E)$ be a $d$-degenerate input graph of maximum degree $\Delta$. To each edge $e \in E$, the oracle associates a bit string, $B(e)$, of length $1 + \lceil \log(2d) \rceil + \lceil \log(d+1) \rceil$. The definition of $B$ falls into two cases.

*Case: $\Delta < 2d$. The oracle computes an optimal edge coloring $\mathcal{C}$ of $G$. Since $\Delta < 2d$, Vizing’s Theorem implies that at most $2d$ different colors are used in $\mathcal{C}$. Let $e \in E$. The first bit of $B(e)$ will be a 0. The next $\lceil \log(2d) \rceil$ bits will encode the color assigned to $e$ in $\mathcal{C}$. The remaining $\lceil \log d \rceil$ bits of $B(e)$ are set arbitrarily.*

*Case: $\Delta \geq 2d$. Fix $a, b \in \mathbb{N}$ such that $\Delta = a2d + b$ and $0 \leq b \leq 2d - 1$. By assumption, $a \geq 1$. The oracle computes an optimal edge coloring $\mathcal{C}$ of $G$. Since $\Delta \geq 2d$, Theorem [1] implies that $\Delta = a2d + b$ colors are used in $\mathcal{C}$. Let $E_0$ be the edges colored with the colors $1, 2, \ldots, b$. For $e \in E_0$, the bit string $B(e)$ is defined as follows: The first bit is a 0. The next $\lceil \log(2d) \rceil$ bits encode the color assigned to $e$ in $\mathcal{C}$ (this is clearly possible since $b \leq 2d - 1$). The remaining $\lceil \log d \rceil$ bits of $B(e)$ are set arbitrarily.

Let $G' = (V, E \setminus E_0)$. Since $G'$ is $a2d$-edge-colorable, its maximum degree is at most $a2d$. On the other hand, no vertex in $V$ is incident to more than $b$ edges from $E_0$, and so the maximum degree is at least $\Delta - b = a2d$. Furthermore, removal of edges cannot increase the degeneracy of a graph. Thus, $G'$ must be $d$-degenerate. For edges in $G'$, the first bit of $B(e)$ is set to 1. The remaining bits of $B(e)$ are constructed by running Procedure [1] on $G'$.

We will now define the algorithm, $\text{ALG}$. For technical reasons, and since the algorithm does not know $\chi'(G)$, we begin by allowing the algorithm to use colors from $\{0,1\} \times \mathbb{N}$. The algorithm receives the advice $B(e)$ along with each edge $e \in E$. If the first bit of $B(e)$ is a 0, the algorithm learns which color, $c$, to use for $e \in E_0$ by reading the next $\lceil \log(2d) \rceil$ bits of $B(e)$. It then assigns the color $(0, c)$ to $e$. If the first bit of $B(e)$ is a 1 then $\text{ALG}$ simulates, using the remaining bits of $B(e)$, the algorithm for the case where $2d$ divides $\Delta$ with $G'$ as input graph. If that algorithm would assign the color $c$ to $e$, $\text{ALG}$ assigns the color $(1, c)$ to $e$. One can easily modify $\text{ALG}$ to use colors from the set $\{1, \ldots, \chi'(G)\}$ as follows: The first time some color $(i, c), i \in \{0,1\}$, is supposed to be used, $\text{ALG}$ selects the lowest color $c'$ from $\{1, \ldots, \chi'(G)\}$ which has not yet been used. From then on, $\text{ALG}$ always uses the color $c'$ instead of $(i, c)$.
It follows from Euler’s formula that every planar graph has a vertex of degree 5 or less. Since removing a vertex from a planar graph preserves planarity, this implies that the degeneracy of a planar graph is at most 5. Thus, we obtain the following corollary:

**Corollary 1.** There exists an algorithm for online edge coloring of planar graphs that uses 8 bits of advice per edge and produces an optimal coloring.

### 3.4 Improvements of the algorithm.

The family of algorithms from Theorem 3 can be used to create a single algorithm which works even if we do not assume that a constant upper bound on the degeneracy is known a priori. The oracle starts by computing the degeneracy, $d_{gn}(G)$, of the input graph $G$. Then, the oracle finds the largest integer $d$ such that $1 + \lceil \log(2d) \rceil + \lceil \log(d+1) \rceil = 1 + \lceil \log(2d_{gn}(G)) \rceil + \lceil \log(d_{gn}(G) + 1) \rceil$. Clearly, $d_{gn}(G) \leq d$ and hence $G$ is $d$-degenerate. When the algorithm receives the very first advice string, it can determine $d$ from the length of the advice received. From there on, Theorem 3 applies. Also, we do not increase the amount of advice by using $d$ instead of $d_{gn}(G)$ as an upper bound on the degeneracy. This gives the following theorem.

**Theorem 4.** There exists an edge-coloring algorithm which always produces an optimal coloring and uses $O(\log d)$ bits of advice per edge, where $d$ is the degeneracy of the input graph.

Recall that Theorem 2 gives a lower bound of $\Omega(\log d)$ on the advice complexity, which asymptotically matches the upper bound of Theorem 3. However, the exact number of bits used by the algorithm presented can be lowered.

Note that there are two reasons for why we require that the maximum degree, $\Delta_i$, of each $(V, E_i)$ should be $2d$. The most fundamental reason is that we need to have the maximum degree large enough so that all back-edges of the same vertex can potentially be assigned to the same subset $E_i$. Thus, $\Delta_i$ should be at least $d$ in order for Lemma 2 to remain true. However, we also need that $(V, E_i)$ is $\Delta_i$-edge-colorable. This may not be the case if $\Delta_i = d$. In order to overcome this problem, we require that $\Delta_i = 2d$ since then the subgraph is $\Delta_i$-edge-colorable by Theorem 1.

For the case of 1-degenerate graphs (i.e. forests), we will sketch how to reduce the number of bits used all the way down to a single bit per edge. The main point is that we do not need to use Theorem 1. Thus, instead of requiring that $\Delta_i = 2d = 2$, we will require that $\Delta_i = d = 1$. Trivially, a subgraph of maximum degree 1 can be colored using a single color since it is just a matching. Also, the algorithm does not need any advice on how to produce this coloring. Instead, it only needs the $\lceil \log(d+1) \rceil = 1$ bit of advice used to assign edges to the correct set $E_i$. Note that the algorithm just defined has a very simple description. If the advice bit for an edge $e$ is a 0, the algorithm uses the lowest available color, $c$. If the advice bit is a 1, then $c$ should not be used since it will be needed later on to color the parent edge of $e$. Instead, the algorithm uses the second lowest available color.

### 4 Concluding remarks and open problems

The lower bound of Theorem 2 relies on the assumption that an algorithm receives some fixed number of advice bits per edge. The lower bound may be viewed as a worst-case lower bound: We show that there exists some edge for which $\Omega(\log \Delta)$ bits of advice are required. Using the advice-on-tape model [5,14], one can also consider the amortized number of advice
bits per edge. In this model, the algorithm is given access to an advice tape prepared by the oracle and may, at any point in time, read some advice bits from the tape. The algorithm from Theorem 3 works in the advice-on-tape model but, as mentioned, the lower bound from Theorem 2 does not carry over. Determining the advice complexity of edge-coloring in the advice-on-tape model is left as an interesting open problem. Another line of research is to study the trade-off between the number of advice bits and the achievable competitive ratio.

Acknowledgements

The author wishes to thank Joan Boyar and Lene M. Favrholdt for helpful discussions.

References

1. Amotz Bar-Noy, Rajeev Motwani, and Joseph Naor. The greedy algorithm is optimal for on-line edge coloring. *Inf. Process. Lett.*, 44(5):251–253, 1992.
2. Kfir Barhum. Tight bounds for the advice complexity of the online minimum steiner tree problem. In *SOFSEM*, pages 77–88, 2014.
3. Shai Ben-David and Allan Borodin. A new measure for the study of on-line algorithms. *Algorithmica*, 11(1):73–91, 1994.
4. Hans-Joachim Böckenhauer, Dennis Komm, Rastislav Královič, and Richard Královic. On the advice complexity of the k-server problem. In *ICALP (1)*, pages 207–218, 2011.
5. Hans-Joachim Böckenhauer, Dennis Komm, Rastislav Královič, Richard Královic, and Tobias Mömke. On the advice complexity of online problems. In *ISAAC*, pages 331–340, 2009.
6. Hans-Joachim Böckenhauer, Dennis Komm, Richard Královic, and Peter Rosmanith. The online knapsack problem: Advice and randomization. *Theor. Comput. Sci.*, 527:61–72, 2014.
7. Allan Borodin, Sandy Irani, Kim S. Larsen, and Alejandro López-Ortiz. Online bin packing with advice. In *STACS*, pages 174–186, 2014.
8. Stefan Dobrev, Rastislav Královič, and Richard Královic. Independent set with advice: The impact of graph knowledge - (extended abstract). In *WAOA*, pages 2–15, 2012.
9. Stefan Dobrev, Rastislav Královič, and Euripides Markou. Online graph exploration with advice. In *SIROCCO*, pages 267–278, 2012.
10. Stefan Dobrev, Rastislav Královič, and Dana Pardubská. Measuring the problem-relevant information in input. *RAIRO Theor. Inform. Appl.*, 43(3):585–613, 2009.
11. Yuval Emek, Pierre Fraigniaud, Amos Korman, and Adi Rosén. Online computation with advice. *Theor. Comput. Sci.*, 412(24):2642–2656, 2011.
12. Michal Foríšek, Lucia Keller, and Monika Steinová. Advice complexity of online graph coloring. unpublished manuscript.
13. Juraj Hromkovic, Rastislav Královič, and Richard Královic. Information complexity of online problems. In *MFCS*, pages 24–36, 2010.
14. Tommy R. Jensen and Bjarne Toft. *Graph Coloring Problems*. Wiley, 2011.
15. Anna R. Karlin, Mark S. Manasse, Larry Rudolph, and Daniel Dominic Sleator. Competitive snoopy caching. *Algorithmica*, 3:77–119, 1988.
16. Claire Kenyon. Best-fit bin-packing with random order. In *SODA*, pages 359–364, 1996.
17. Dénes König. Über Graphen und ihre Anwendung auf Determinantentheorie und Mengenlehre. *Math. Ann.*, 77(4):453–465, 1916.
18. Elias Koutsoupias and Christos H. Papadimitriou. Beyond competitive analysis. *SIAM J. Comput.*, 30(1):300–317, 2000.
19. Shuichi Miyazaki. On the advice complexity of online bipartite matching and online stable marriage. *Inf. Process. Lett.*, 114(12):714–717, 2014.
20. Prabhakar Raghavan. A statistical adversary for on-line algorithms. In *On-Line Algorithms*, DIMACS Series in Discrete Mathematics and Theoretical Computer Science, pages 79–83, 1991.
21. Marc P. Renault, Adi Rosén, and Rob van Stee. Online algorithms with advice for bin packing and scheduling problems. *CoRR*, abs/1311.7589, 2013.
23. Daniel D. Sleator and Robert E. Tarjan. Amortized efficiency of list update and paging rules. *Commun. ACM*, 28(2):202–208, 1985.

24. Michael Stiebitz, Diego Scheide, Bjarne Toft, and Lene M. Favrholdt. *Graph Edge Coloring: Vizing’s Theorem and Goldberg’s Conjecture*. Wiley, 2012.

25. V. G. Vizing. On an estimate of the chromatic class of a $p$-graph (in Russian). *Metody Diskret. Analiz.*, 3:25–30, 1964.

26. V. G. Vizing. Critical graphs with given chromatic class (in Russian). *Metody Diskret. Analiz.*, 5:9–17, 1965.