Advice Complexity of Online Coloring for Paths*

Michal Forišek[1], Lucia Keller[2], and Monika Steinová[2]

[1] Comenius University, Bratislava, Slovakia,
forisek@dcs.fmph.uniba.sk
[2] ETH Zurich, Switzerland,
{lucia.keller,monika.steinova}@inf.ethz.ch

Abstract. In online graph coloring a graph is revealed to an online algorithm one vertex at a time, and the algorithm must color the vertices as they appear. This paper starts to investigate the advice complexity of this problem – the amount of oracle information an online algorithm needs in order to make optimal choices. We also consider a more general problem – a trade-off between online and offline graph coloring.
In the paper we prove that precisely $\lceil n/2 \rceil - 1$ bits of advice are needed when the vertices on a path are presented for coloring in arbitrary order. The same holds in the more general case when just a subset of the vertices is colored online. However, the problem turns out to be non-trivial for the case where the online algorithm is guaranteed that the vertices it receives form a subset of a path and are presented in the order in which they lie on the path. For this variant we prove that its advice complexity is $\beta n + O(\log n)$ bits, where $\beta \approx 0.406$ is a fixed constant (we give its closed form). This suggests that the generalized problem will be challenging for more complex graph classes.
Keywords. advice complexity, online graph coloring, partial coloring

1 Overview of Advice Complexity

A great challenge in classical algorithmics are problems that work in an online fashion: The instance is not shown to the algorithm all at once. Instead, the algorithm receives it piecewise in consecutive turns. In each turn, the algorithm must produce a piece of the output which must not be changed afterwards. Such algorithms are called *online algorithms*, more on them can be found in [4].

Obviously, it is harder (and sometimes even impossible) to compute the best partial solutions without knowing the future. The output quality of online algorithms is measured by the *competitive ratio* of a particular instance, see [4, 12]. This ratio is the quotient of the cost of the solution produced by the given online algorithm and the cost of an optimal solution (i.e., the cost of the output of an optimal offline algorithm for that particular instance).

The exact advantage of the offline algorithm can be measured as the amount of additional information, "advice", the online algorithm needs in order to produce an optimal solution. This notion can be formalized by providing the online

* The research is partially funded by the SNF grant 200021-132510/1 and by VEGA grants 1/0726/09 and 1/0979/12.

algorithm with an additional source of information: an advice tape. The content of the advice tape is assumed to be prepared in advance by an oracle which has unlimited computational power and has access to the whole input instance. This model of advice complexity has first been introduced in [6].

The original model [6] had a minor technical disadvantage: the advice tape was finite, hence its length could be used to encode additional information. There were two attempts to fix this. In [8], the authors force the algorithm to use a fixed number of advice bits in every turn. The drawback of this approach is that it is not possible to determine sublinear advice complexity. A better version of the model was proposed in [3]. In this version the advice tape is considered to be infinite, and we consider the length of its prefix that is actually examined by the online algorithm. We are using this model in our paper.

There are indeed problems where already a small amount of information is enough for an online algorithm to be optimal. For example, the online problem SkiRental needs only 1 bit for being optimal, see [4, 6]. In recent years, advice complexity was also investigated for other classical online problems, such as the k-server problem [2] and the paging problem [13].

Advice complexity has its theoretical importance in measuring an exact quantity of information that directly characterizes the hardness of an online problem compared to its offline version. It is also worth noting that advice complexity is closely related to both non-determinism (in terms of the oracle) and randomization. In [14] relations between advice complexity and randomized algorithms are shown, and a new randomized algorithm is designed based on a careful computation of the advice complexity of a given problem. This makes advice complexity a new and important point of view on online algorithms.

In this paper we present the initial results of advice complexity for a very important online problem: online graph coloring. The general offline version of this problem is a well-known NP-complete problem, and all online algorithms without advice are known to be extremely poor in the worst case [11]. This huge difference between online and offline algorithms makes this problem extremely interesting from the advice complexity point of view. Our goal in this paper is to provide an exact analysis of its simplest versions. In particular, we analyze online 2-coloring of paths. We also consider a more general version of graph coloring, where only a subset of vertices is colored online. Many such problems on subgraphs, including path coloring, were considered in [1] and are known to be very hard to approximate. Finally, we outline the approach that should be taken when analyzing more general versions of online graph coloring.

2 Definitions

2.1 Advice Complexity Definitions

In this part of our paper we provide the necessary subset of definitions from [3].

Definition 1. *An online algorithm A with advice is defined as follows: The input for the algorithm is a sequence* $(x_1, \ldots, x_n)$ *and an infinite advice word*

$\varphi \in \{0,1\}^\omega$. *The algorithm produces an output sequence* $(y_1, \ldots, y_n)$ *with the restriction that* $\forall i : y_i$ *is computed only from* $x_1, \ldots, x_i$ *and* φ.

The computation of A can be seen as a sequence of turns, where in i-th turn A reads x_i and then produces y_i using all the information read so far and possibly some new bits of the advice word. Note that the definition does not restrict the computational power of the online algorithms. Still, all of the algorithms in our paper will be deterministic and they will all have a polynomial time complexity.

Definition 2. *The advice complexity of A is a function s such that $s(n)$ is the smallest value such that for no input sequence of size n the algorithm A examines more than the first $s(n)$ bits of the advice word φ.*

Definition 3. *The advice complexity of an online problem is the smallest advice complexity an online algorithm with advice needs to produce an optimal solution (i.e., a solution as good as an optimal offline algorithm would produce).*

In [3] the authors also provide definitions for online approximation algorithms with advice. For these algorithms, one can look at the tradeoff between the amount of advice available and the competitive ratio of the algorithm. Sometimes already very little advice can reduce the competive ratio significantly. One example is the paging problem, also considered in [3]. In our paper we only consider algorithms that produce optimal solutions, hence we omit these definitions. However, in the conclusion of this paper we show that this question will be very interesting when our problem is considered for more general classes of graphs.

2.2 Online Coloring Definitions

Definition 4. *In* OnlineColoring *the instance is an undirected graph $G = (V, E)$ with $V = \{1, 2, \ldots, n\}$. This graph is presented to an online algorithm in turns: In k-th turn the online algorithm receives the graph $G_k = G[\,\{1, 2, \ldots, k\}\,]$, i.e., a subgraph of G induced by the vertex set $\{1, 2, \ldots, k\}$. As its reply, the online algorithm must return a positive integer: the color it wants to assign to vertex k. The goal is to produce the optimal coloring of G – the online algorithm must assign distinct integers to adjacent vertices, and the largest integer used must be as small as possible.*

Vertex labels in G correspond to the order in which the online algorithm is asked to color the vertices. An equivalent definition: in the k-th turn the online algorithm is given a list of edges between k and vertices in $\{1, 2, \ldots, k-1\}$.

Note that the value n is not known a priori by the online algorithm. This is intentional and necessary. Announcing the value n to the online algorithm would give it additional information about the instance it will be processing, and this amount of information would then not be reflected in the advice complexity.

Definition 5. OnlinePartialColoring *is a generalization of* OnlineColoring. *In this case, the instance is G along with an integer sequence $(a_1, \ldots, a_m)$ such that $1 \leq a_1 < \cdots < a_m \leq n$.*

For a given integer k, let $\overline{G_k} = G[\{a_1, a_2, \ldots, a_k\}]$ be the subgraph induced by $\{a_1, \ldots, a_k\}$. The graph G_k can be obtained from $\overline{G_k}$ by changing vertex labels from $(a_1, \ldots, a_k)$ to $(1, \ldots, k)$. In turn k, the online algorithm is shown the graph G_k and it must return the color assigned to the most recently added vertex (the one with label k in G_k and label a_k in G). The goal is to produce a coloring of $G[\{a_1, a_2, \ldots, a_m\}]$ that can be extended to an optimal coloring of G.

This problem can be seen as a parametrized trade-off between the online and offline version of the coloring problem. We start with an uncolored graph G. First, a part of G is colored in an online fashion. At the end, the rest of G is colored offline. (By setting $m = 0$ in OnlinePartialColoring we obtain the offline version, and for $m = n$ we get the online version of classic coloring.)

Neither n nor m are known by the online algorithm. The relabeling of vertices prevents the online algorithm from deducing their location in G.

Both OnlineColoring and OnlinePartialColoring can be considered not only for general graphs, but also for specific graph classes – the online player will then know that the entire graph G belongs to the considered class of graphs. In this paper, we will be considering these two online problems for two of the simplest graph classes – arbitrarily numbered and sequentially numbered paths. (In some literature, these are denoted "paths with arbitrary presentation" and "paths with connected presentation". For many graph classes, OnlineColoring is hard even with connected presentation. See [5] for more.)

Definition 6. *An arbitrary numbered path with n vertices is a graph with vertices $\{1, \ldots, n\}$, $n - 1$ edges and max. degree 2. A sequentially numbered path with n vertices is a graph with vertices $\{1, \ldots, n\}$ and edges $\{(i, i+1) | 1 \leq i < n\}$.*

We will be analyzing the advice complexity of these problems given that G comes from one of these graph classes and that the online algorithm is deterministic. Already for these simple cases the results will be non-trivial.

Note that for any particular class of graphs, an algorithm with advice that solves OnlinePartialColoring can be directly used to solve OnlineColoring. But on the other hand, we will show that for sequentially numbered paths OnlinePartialColoring requires strictly more advice than OnlineColoring. (Here it is a slightly artificial difference, as OnlineColoring needs no advice, but we expect to see similar differences in more complex graph classes.)

3 Theorems

In this section we list our results stated as theorems.

Theorem 1. *For an arbitrary path on n vertices the advice complexity of both* OnlineColoring *and* OnlinePartialColoring *is exactly $\lceil n/2 \rceil - 1$ bits.*

Theorem 2. *For a sequentially numbered path on n vertices the advice complexity of* OnlineColoring *is zero and the advice complexity of* OnlinePartialColoring *is $\beta n + O(\log n)$, where $\beta \approx 0.40568523$ is the binary logarithm of the plastic constant.*

In all proofs, n is implicitly used as the number of vertices of G for the given instance. More details on β, including its closed form, are given in Section 5. The notation "lg" in the proofs is the base-2 (binary) logarithm.

4 Proof of Theorem 1

Lemma 1. *An online algorithm with advice that solves* OnlinePartialColoring *and knows that G is 2-colorable never needs to access advice bits whenever the degree of the currently processed vertex k in the current graph G_k is positive.*

Proof. If the degree of k in G_k is positive, in G the vertex a_k must be adjacent to some a_i such that $i < k$. The online algorithm already assigned a color to a_i, and now it must use the other color for a_k. This can be done without advice. □

Lemma 2. *There is a deterministic online algorithm solving* OnlinePartialColoring *for arbitrary paths with advice complexity $\lceil n/2 \rceil - 1$.*

Proof. As suggested by Lemma 1, our algorithm only asks for advice (i.e., reads the next bit of the advice word) whenever the current vertex k is isolated in G_k. One bit of advice is sufficient – the advice can be interpreted as the correct color to use. The above only applies for $k > 1$, as we may pick an arbitrary color for the first isolated vertex.

Let S be the set of vertices that were isolated at the moment we processed them. Clearly, it follows that no two of them are adjacent in G, hence S is an independent set in G and therefore $|S| \leq \lceil n/2 \rceil$. □

Lemma 3. *Any deterministic online algorithm solving* OnlineColoring *for arbitrary paths needs at least $\lceil n/2 \rceil - 1$ bits of advice in the worst case.*

Note that the proof of the lower bound of $\lfloor n/2 \rfloor - 1$ bits is reasonably simple; for odd n we have to use a more careful analysis to force the extra bit.

Proof. We will denote the vertices $v_1, \ldots, v_n$ in the order in which they appear on the path: for all i the vertices v_i and v_{i+1} are adjacent. Note that we do not know the exact numbers of these vertices. (Each path produces two such sequences. But in our proof we will only consider sequences where $v_1 = 1$, so different sequences $(v_1, \ldots, v_n)$ will indeed correspond to different graphs G.)

Let $k = \lfloor n/2 \rfloor$. The graph G_k will be called the prefix of an instance. We will only consider instances where the prefix consists of k isolated vertices. Out of these instances, we will pick a set of instances S with the following property: for no two instances in S can their prefixes be colored in the same way. (Note that each instance has exactly two valid colorings, hence the prefix of each instance also has exactly two valid colorings – one a complement of the other.) As for a deterministic algorithm the prefixes of instances in S are indistinguishable, all information about their correct coloring must be given as advice.

For any x ($1 \leq x \leq \lfloor n/2 \rfloor$) consider the two sets of positions on the path $P_x = \{2i - 1 \mid 1 \leq i \leq x\}$ and $Q_x = \{2i \mid x + 1 \leq i \leq \lfloor n/2 \rfloor\}$ (see Fig. 1). Note

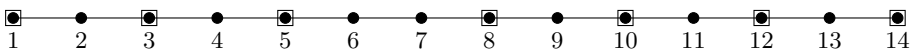

Fig. 1. Example for $n = 14$ and $x = 3$: $P_x = \{1, 3, 5\}$ and $Q_x = \{8, 10, 12, 14\}$.

that all vertices v_i for $i \in P_x$ must share one color, and all vertices v_j for $j \in Q_x$ must share the other color. (Also note, that P_x and Q_x are sets of indices of vertices v_i and not their labels.)

Let I_x be the set of all instances where the vertices on positions in $P_x \cup Q_x$ form the prefix. Formally, in these instances $\forall i \in P_x \cup Q_x : v_i \leq k$. Note that for any such instance the prefix indeed consists of k isolated vertices.

We will now define the set S: Consider all strings $w \in \{\mathtt{p}\} \cdot \{\mathtt{p}, \mathtt{q}\}^{k-1}$. For each such string, we pick into S a single instance: Let x be the number of $\mathtt{p}$s in w. We will pick the lexicographically smallest[3] instance from I_x such that $\forall i \leq n$: if $v_i \leq k$, then ($i \in P_x$ iff the i-th letter of w is $\mathtt{p}$). In words: The string w determines the value of x and gives the order in which vertices from P_x and Q_x are picked for coloring. There is always at least one such instance; if there are multiple, any will do, so we pick the lexicographically smallest one.

In this way we constructed a set S of 2^{k-1} instances (S contains one instance per each string $\{\mathtt{p}\} \cdot \{\mathtt{p}, \mathtt{q}\}^{\mathtt{k-1}}$) such that for no two instances in S the prefix can be colored in the same way. By the pigeon-hole principle, if there was a deterministic online algorithm that always uses less than $k - 1$ bits of advice, two of the instances would receive the same advice, hence the algorithm would produce the same coloring of their prefixes, which is a contradiction. Therefore any deterministic online algorithm needs at least $k-1 = \lfloor n/2 \rfloor - 1$ bits of advice. That concludes the proof for even n.

For odd n, we want to prove that any deterministic online algorithm must use at least k bits of advice. By contradiction. Assume that the algorithm always uses less than k bits of advice for paths of length n. This means that on instances from S the algorithm always reads all $k-1$ bits of advice, and different instances in S must correspond to different $k - 1$ bits of advice.

In S we have an instance J_1 that corresponds to $\mathtt{p}^{\mathtt{k}}$. For this instance all vertices in the prefix must receive the same color. Let φ_1 be the first $k - 1$ bits of advice for this instance. Consider any instance (possibly with more than n vertices) such that G_k consists of isolated vertices that should receive the same color as J_1. Clearly, for any such instance the first $k - 1$ bits of advice must be φ_1 – otherwise the deterministic algorithm would color the first k vertices in a different way.

Now consider one additional instance J_2: the lexicographically smallest one where $v_i \leq k + 1$ iff i is odd. (This instance is similar to J_1, but in J_1 we have $v_2 = k + 1$ and in J_2 we have $v_n = k + 1$. For J_2 the graph G_{k+1} has $k + 1$ isolated vertices.) As our algorithm never uses k bits of advice, it must process v_n without any additional advice. Thus the instances J_1 and J_2 are for the

[3] I.e., one for which the vector $(v_1, v_2, \dots)$ is the smallest.

algorithm undistinguishable and the algorithm cannot use extra bits of advice to color J_2. Hence whenever our algorithm is presented with an instance (of any size) such that the first k vertices are isolated and must share the same color, it will color the next isolated vertex using the same color. And this is a contradiction: we can easily create an instance of size $n+1$ where the $(k+1)$-st isolated vertex should have the opposite color. □

Proof of Theorem 1. By Lemma 3, any deterministic online algorithm for OnlineColoring needs at least $\lceil n/2 \rceil - 1$ bits of advice in the worst case. As OnlinePartialColoring is a generalization of OnlineColoring, this lower bound transfers to OnlinePartialColoring. By Lemma 2 there is a deterministic online algorithm solving OnlinePartialColoring with $\lceil n/2 \rceil - 1$ bits of advice, hence that is the exact advice complexity for both problems. □

5 Proof of Theorem 2

Lemma 4. OnlineColoring *for sequentially numbered paths can be solved without advice.*

Proof. Trivially follows from Lemma 1. □

Before we give the proof for OnlinePartialColoring, we first note some trivial upper and lower bounds. The algorithm shown in Lemma 2 gives us an upper bound of $\lceil n/2 \rceil - 1$ bits of advice. As we show below, this is not optimal. A trivial lower bound of $\lfloor n/3 \rfloor - 1$ bits of advice is obtained by considering instances where $m = \lfloor n/3 \rfloor$, $a_1 = 1$ and $a_i \in \{3i-2, 3i-1\}$ for $1 < i \leq m$. These are 2^{m-1} instances, and the algorithm needs to receive different advice for all of them, hence $\lg 2^{m-1} = m - 1$ bits of advice are necessary.

Lemma 5. *For any positive integer s and real number $r \geq 1$*

$$\frac{1}{4rs} \cdot \left(\frac{(r+1)^{r+1}}{r^r} \right)^s \leq \binom{(r+1)s}{s} \leq \left(\frac{(r+1)^{r+1}}{r^r} \right)^s.$$

Proof. These bounds are given in [15]. □

Lemma 6. *Every deterministic online algorithm that solves* OnlinePartialColoring *for sequentially numbered paths needs at least $\beta n - \lg n + O(1)$ bits of advice, where $\beta \approx 0.40568523$ is the binary logarithm of the plastic constant.*

Proof. We will describe a special set S of instances. In all these instances m will be the same and $a_{i+1} \geq a_i + 2$ for all i – hence all queries will be isolated vertices. No two instances in S will admit the same coloring, hence the amount of necessary advice bits will be bounded from below by $\lceil \lg |S| \rceil$.

The set S will be formed by all instances that have $a_1 = 1$ and $a_{i+1} \in \{a_i + 2, a_i + 3\}$, for $i \geq 1$, with the additional restriction that in all instances there are exactly k values i such that $a_{i+1} = a_i + 2$ and exactly l other values i.

Hence $|S| = \binom{k+l}{k}$, $m = k+l+1$, and the number of vertices in each instance is $n = 2k+3l+1$. (No two of these instances admit the same coloring: $a_{i+1} = a_i + 3$ always forces a color change.) To make formulas simpler, let $x = n-1$.

We are now looking for the values k and l such that the number of such instances is maximized. Mathematical intuition suggests $k = l$ as a likely optimum, but this turns out to be false. (But note that taking $k = l$ leads to a correct and good lower bound of $0.4x$ bits of advice.)

For a fixed $n = 2k+3l+1$ we are maximizing $\binom{k+l}{l}$. Let $l = \alpha x$ for some α, hence $k = (1-3\alpha)x/2$ and we are maximizing $\binom{(1-\alpha)x/2}{\alpha x}$ as a function of $\alpha \in (0,1)$. It is easily verified that the function is decreasing for $\alpha \in [1/5, 1)$ (which corresponds to the case $k < l$). Hence we just consider $\alpha \in (0, 1/5]$.

In Lemma 5, let $s = l$ and $r = k/l = (x-3l)/(2l) = (1-3\alpha)/(2\alpha)$. (Note that $r \geq 1$.) This bounds our binomial coefficient from below:

$$\begin{aligned}
\binom{k+l}{l} &= \binom{(1-\alpha)x/2}{\alpha x} \\
&\geq \frac{2\alpha}{4(1-3\alpha)\alpha x} \cdot \left(\frac{\left(\frac{1-\alpha}{2\alpha}\right)^{\frac{1-\alpha}{2\alpha}}}{\left(\frac{1-3\alpha}{2\alpha}\right)^{\frac{1-3\alpha}{2\alpha}}} \right)^{\alpha x} \\
&= \frac{1}{2(1-3\alpha)x} \cdot \left(\frac{(1-\alpha)^{(1-\alpha)/2}}{(1-3\alpha)^{(1-3\alpha)/2} \cdot (2\alpha)^{\alpha}} \right)^{x} = f(\alpha)
\end{aligned}$$

Hence we get the lower bound: $\max_{0<\alpha\leq 1/5} \binom{(1-\alpha)x/2}{\alpha x} \geq \max_{0<\alpha\leq 1/5} f(\alpha)$.

When constructing the set S, we may pick the value α that maximizes $f(\alpha)$, thereby ensuring that any deterministic online algorithm will need at least $\lceil \lg \max_{0<\alpha\leq 1/5} f(\alpha) \rceil$ bits of advice. We may now compute:

$$\begin{aligned}
&\lg \max_{0<\alpha\leq 1/5} f(\alpha) \\
&= \lg \max_{0<\alpha\leq 1/5} \frac{1}{2(1-3\alpha)x} \cdot \left(\frac{(1-\alpha)^{(1-\alpha)/2}}{(1-3\alpha)^{(1-3\alpha)/2} \cdot (2\alpha)^{\alpha}} \right)^{x} \\
&= \lg \frac{1}{2x} + \max_{0<\alpha\leq 1/5} \left(-\lg(1-3\alpha) + x \cdot \lg \left(\frac{(1-\alpha)^{(1-\alpha)/2}}{(1-3\alpha)^{(1-3\alpha)/2} \cdot (2\alpha)^{\alpha}} \right) \right) \\
&\geq \lg \frac{1}{2x} - \lg \min_{0<\alpha\leq 1/5} (1-3\alpha) + x \cdot \lg \max_{0<\alpha\leq 1/5} \underbrace{\left(\frac{(1-\alpha)^{(1-\alpha)/2}}{(1-3\alpha)^{(1-3\alpha)/2} \cdot (2\alpha)^{\alpha}} \right)}_{g(\alpha)} \\
&= -1 - \lg x - \lg(2/5) + \beta x \qquad \text{(see below for the value of } \beta)
\end{aligned}$$

Let α_m be the value of α for which the last expression, denoted $g(\alpha)$, is maximized. A numerical computation showed that $\alpha_m \approx 0.17700882$, which is

indeed in the correct range (hence indeed $l < k$). The value of the maximized expression is $g(\alpha_m) \approx 1.3247179572$, hence its binary logarithm is $\beta = \lg g(\alpha_m) \approx 0.40568523$.

Using these approximate values we were able to guess and then to verify closed forms for these constants. All three constants are related to a well-known mathematical constant P, often called the plastic constant – see [9] for more details. The constant P is the only real root of the polynomial $x^3 - x - 1$. In terms of P, we can express our constants as $\alpha_m = 1/(2P+3)$ and $g(\alpha_m) = P$, therefore we obtain $\beta = \lg\left(\sqrt[3]{9-\sqrt{69}} + \sqrt[3]{9+\sqrt{69}}\right) - \lg\sqrt[3]{18}$.

Note that in the calculation of this lower bound we worked with possibly non-integer values for k and l. For an actual lower bound, we should take their floors. Clearly, this only decreases the required advice by a constant. Hence we may conclude that the advice complexity of solving OnlinePartialColoring on a sequentially numbered path with n vertices is at least $\beta n - \lg n + O(1)$. □

Before we prove a very close upper bound, we repeat that the naive strategy "each time a vertex needs coloring, the advice is the color" is not optimal. It turns out that the instances identified in the lower bound are precisely the worst case. A general idea of the proof can be summarized as follows: whenever m is close to $n/2$ (hence the instance is dense and we would need too much advice in the naive approach), the number of color changes in the instance is necessarily small, and we may only encode their locations as advice.

Lemma 7. *There is a deterministic online algorithm that solves* OnlinePartialColoring *for sequentially numbered paths and never uses more than* $\beta n + 2\lg n + \lg\lg n + O(1)$ *bits of advice, where* β *is exactly the same constant as in Lemma 6 and before.*

Proof. The general outline of this proof: We will transform each instance to an instance where all queries are isolated vertices that are as close to the left of the path as possible, given the colors they are supposed to receive. The advice will then be a triple containing n; the number of color changes in the (optimal) coloring; and the index of the lexicographically smallest instance with the same coloring as our given instance has. We will prove that this will indeed solve the problem, and that the number of advice bits matches the claim above.

To improve our trivial upper bound and match it with the lower bound presented in Lemma 6, we need to make a few observations. First of all, we only need to consider instances in which $a_{i+1} \geq a_i + 2$ for all i: the online algorithm can handle queries where $a_{i+1} = a_i + 1$ without advice, see Lemma 1.

Hence we only consider instances where the colored vertices form an independent set. Let $(a_1, \ldots, a_m)$ be the sequence of requests for one such instance I. Let l be the number of times the value $a_{i+1} - a_i$ is odd, i.e., the number of color changes. Let $k = \lfloor (n-1-3l)/2 \rfloor$.

Now consider the set S of instances constructed in the proof of Lemma 6 for the values k and l we just defined. Clearly, for our instance I there is an instance I' in S with the following property: the sequence of colors assigned to queries in I is a prefix of the sequence of colors assigned to the queries in I'.

In fact, I' can be easily constructed from I: We start with the sequence $(a_1, \ldots, a_m)$ of queries in I. We subtract $a_1 - 1$ from all elements, getting a sequence that starts with 1. Then we process the values a_i for $i = 2..n$, and for each of these values we change a_i into $(a_{i-1} + 2 + (a_i - a_{i-1}) \bmod 2)$. (That is, we shift a_i to the left so that the distance $a_i - a_{i-1}$ becomes either 2 or 3; we preserve parity of the distance.) Finally, if now the last request a_m is less than $n - 1$, we append additional requests, each of them in distance two from the previous one.

If our deterministic online algorithm receives the instance I, the advice it will receive will consist of three parts: the value n, the value l, and a number d. To obtain d, we construct the correct set S and order all its elements lexicographically; d is the position of the instance I' in this order.

The value n can easily be encoded into the first $\lg n + \lg\lg n + 1$ bits of advice using a suitable prefix code, such as the Elias delta code [7]. As the deterministic online algorithm already knows n before reading l and d, we can avoid using a prefix code for these: after the bits representing n, the advice word will contain exactly $\lfloor \lg n \rfloor + 1$ bits for l and another $\lceil \lg |S| \rceil$ bits for d.

(We note that the computational power of computing advice is not considered in the advice complexity model, and neither is the time complexity of the online algorithm. However, the value d can in fact be computed in time polynomial in n, and also reconstructing I' from d can be done in polynomial time using suitable combinatorial algorithms. As this is not the scope of our paper, we omit these algorithms.)

We see that the amount of advice needed depends on the size of the largest possible set S. To compute it we need to bound the same value $\binom{k+l}{l} = \binom{(1-\alpha)x/2}{\alpha x}$ as in the previous Lemma, but this time from above. Again, we apply Lemma 5:

$$\begin{aligned}
\binom{k+l}{l} &= \binom{(1-\alpha)x/2}{\alpha x} \\
&\leq \left(\frac{\left(\frac{1-\alpha}{2\alpha}\right)^{\frac{1-\alpha}{2\alpha}}}{\left(\frac{1-3\alpha}{2\alpha}\right)^{\frac{1-3\alpha}{2\alpha}}} \right)^{\alpha x} \\
&= \left(\frac{(1-\alpha)^{(1-\alpha)/2}}{(1-3\alpha)^{(1-3\alpha)/2} \cdot (2\alpha)^{\alpha}} \right)^{x} = g(\alpha)^x
\end{aligned}$$

Hence we get the upper bound:

$$\max_{0<\alpha\leq 1/5} \binom{(1-\alpha)x/2}{\alpha x} \leq \max_{0<\alpha\leq 1/5} g(\alpha)^x$$

And using the same computations as in Lemma 6 we arrive at the conclusion:

$$\lg \max_{0<\alpha\leq 1/5} g(\alpha)^x = x \cdot \lg \max_{0<\alpha\leq 1/5} \left(\frac{(1-\alpha)^{(1-\alpha)/2}}{(1-3\alpha)^{(1-3\alpha)/2} \cdot (2\alpha)^{\alpha}} \right) = \beta x,$$

where β is the same constant as in Lemma 6. As in the previous lemma, we may need to round the optimal k and l to integers (this time to ceilings of their

actual values), which may increase $2k + 3l$ by a constant, hence it may increase the number of advice bits by a constant. Therefore $\beta n + 2 \lg n + \lg \lg n + O(1)$ bits of advice are sufficient. □

Proof of Theorem 2. Follows trivially from Lemma 6 and Lemma 7. □

6 Conclusion

In the paper we showed that already the simplest versions of online graph coloring are non-trivial in terms of advice complexity. We analyzed two such versions and obtained exact (in one case) and almost exact (in the other case) bounds on their advice complexity.

One direction of future research is clear. All of the problems considered in this paper generalize to more complex graph classes. In particular, coloring of sequentially numbered paths generalizes to coloring of graphs numbered according to their depth-first or breadth-first traversal. It should be possible to generalize our results to more complex graph classes. We also expect that this point of view may lead to new randomized approximation algorithms and/or new inapproximability results for some graph classes.

However, for more complex graph classes a more fine-grained analysis should also be possible. We will now give an outline of its general idea. Note that for all problems presented in the paper there is a trivial online approximation algorithm that never uses more than three colors. Hence it does not make sense to consider online approximation algorithms – either the online algorithm gets it right, or it does not, in which case its competitive ratio is trivially 3/2. However, as we move to more complex graph classes, the difference between online and offline algorithms increases. For instance, every tree is trivially 2-colorable, but it has been shown [10] that any online algorithm without advice can be forced to use $\Theta(\log n)$ colors on a n-vertex tree.

The advice complexity of obtaining the optimal coloring can be high. For instance, there are n-vertex trees with $n - 2$ leaves, and for these we obviously need $n - 3$ bits of advice in order to produce the optimal coloring. A natural question here is "what can we get for less?" That is, it should be possible to analyze the tradeoff between the amount of advice available to the online algorithm and the quality of the obtained approximation. This more detailed analysis may then also lead to new, interesting approximation algorithms.

Such analysis can be done for all of the important graph classes. We expect most of these problems to be hard, but worth solving – their solutions should give us a better understanding of online coloring.

Acknowledgement

The authors are thankful to Juraj Hromkovič for bringing this problem to their attention, and to Hans-Joachim Böckenhauer and Maria Paola Bianchi for the interesting and helpful discussions about this problem.

References

1. Bartal, Y., Fiat, A., Leonardi, S.: Lower bounds for on-line graph problems with application to on-line circuit and optical routing. SIAM J. Comput. 36, 354–393 (August 2006)
2. Böckenhauer, H.J., Komm, D., Královič, R., Královič, R.: On the advice complexity of the k-server problem. In: Aceto, L., Henzinger, M., Sgall, J. (eds.) Proceedings of the 38th International Colloquium on Automata, Languages and Programming (ICALP 2011), Part I. Lecture Notes in Computer Science, vol. 6755, pp. 207–218. Springer-Verlag (2011)
3. Böckenhauer, H.J., Komm, D., Královič, R., Královič, R., Mömke, T.: On the advice complexity of online problems. In: Dong, Y., Du, D.Z., Ibarra, O.H. (eds.) Algorithms and Computation, 20th International Symposium, ISAAC 2009, Honolulu, Hawaii, USA, December 16-18, 2009. Proceedings. Lecture Notes in Computer Science, vol. 5878, pp. 331–340. Springer-Verlag (2009)
4. Borodin, A., El-Yaniv, R.: Online Computation and Competitive Analysis. Cambridge University Press (1998)
5. Cieślik, I.: On-line graph coloring. Ph.D. thesis, Jagiellonian University Kraków (2006)
6. Dobrev, S., Královič, R., Pardubská, D.: How much information about the future is needed? In: Geffert, V., Karhumäki, J., Bertoni, A., Preneel, B., Návrat, P., Bieliková, M. (eds.) Proceeding of the 34th International Conference on Current Trends in Theory and Practice of Computer Science (SOFSEM 2008). Lecture Notes in Computer Science, vol. 4910, pp. 247–258. Springer-Verlag, Berlin (2008)
7. Elias, P.: Universal codeword sets and representations of the integers. IEEE Transactions on Information Theory 21, 194–203 (1975)
8. Emek, Y., Fraigniaud, P., Korman, A., Rosén, A.: Online computation with advice. In: Albers, S., Marchetti-Spaccamela, A., Matias, Y., Nikoletseas, S.E., Thomas, W. (eds.) Proceedings of the 36th International Colloquium on Automata, Languages and Programming (ICALP 2009). Lecture Notes in Computer Science, vol. 5555, pp. 427–438. Springer-Verlag (2009)
9. Finch, S.R.: Mathematical Constants (Encyclopedia of Mathematics and its Applications). Cambridge University Press (Aug 2003)
10. Gyrfs, A., Lehel, J.: On-line and first fit colorings of graphs. Journal of Graph Theory 12(2), 217–227 (1988)
11. Halldórsson, M.M., Szegedy, M.: Lower bounds for on-line graph coloring. In: Proceedings of the third annual ACM-SIAM symposium on Discrete algorithms. pp. 211–216. SODA '92, Society for Industrial and Applied Mathematics, Philadelphia, PA, USA (1992)
12. Hromkovič, J.: Design and analysis of randomized algorithms. Texts in Theoretical Computer Science. An EATCS Series, Springer-Verlag, Berlin (2005)
13. Hromkovič, J., Královič, R., Královič, R.: Information complexity of online problems. In: MFCS. Lecture Notes in Computer Science, vol. 6281, pp. 24–36. Springer (2010)
14. Komm, D., Královič, R.: Advice complexity and barely random algorithms. In: Černá, I., Gyimóthy, T., Hromkovič, J., Jeffery, K.G., Královič, R., Vukolic, M., Wolf, S. (eds.) SOFSEM. Lecture Notes in Computer Science, vol. 6543, pp. 332–343. Springer (2011)
15. Sondow, J., Zudilin, W.: Eulers constant, q-logarithms, and formulas of Ramanujan and Gosper. The Ramanujan Journal 12, 225–244 (2006)