

Online Chromatic Number is PSPACE-Complete

Martin Böhm^(✉) and Pavel Veselý^(✉)

Computer Science Institute of Charles University, Prague, Czech Republic
{bohm,vesely}@iuuk.mff.cuni.cz

Abstract. In the online graph coloring problem, vertices from a graph G , known in advance, arrive in an online fashion and an algorithm must immediately assign a color to each incoming vertex v so that the revealed graph is properly colored. The exact location of v in the graph G is not known to the algorithm, since it sees only previously colored neighbors of v . The *online chromatic number* of G is the smallest number of colors such that some online algorithm is able to properly color G for any incoming order. We prove that computing the online chromatic number of a graph is PSPACE-complete.

1 Introduction

In the classical graph coloring problem we assign a color to each vertex of a given graph such that the graph is properly colored, i.e., no two adjacent vertices have the same color. The chromatic number χ of a graph G is the smallest k such that G can be colored with k distinct colors. Deciding whether the chromatic number of a graph is at most k is well known to be NP-complete, even in the case with three colors.

The online variant of graph coloring can be defined as follows: The vertices of G arrive one by one, and an online algorithm must color vertices as they arrive so that the revealed graph is properly colored at all times. When a vertex arrives, the algorithm sees edges to previously colored vertices. The online algorithm may use additional knowledge of the whole graph G ; more precisely, a copy of G is sent to the algorithm at the start of the input. However, the exact correspondence between the incoming vertices and the vertices of the copy of G is not known to the algorithm. This problem is called **ONLINE GRAPH COLORING**.

In this paper we focus on a graph parameter called the *online chromatic number* $\chi^O(G)$ of a graph G . This parameter is analogous to the standard chromatic number of a graph: It denotes the smallest number k such that there exists a deterministic online algorithm which is able to color the specified graph G using k colors.

The online chromatic number has been studied since 1990 [3]. One of the main open problems in the area is the computational complexity of deciding

M. Böhm—Supported by CE-ITI under grant P202/12/G061 of GA ČR and by the GAUK project 548214.

P. Veselý—Supported by the GAUK project 548214.

whether $\chi^O(G) \leq k$ for a specified simple graph G , given G and k on input; see e.g. Kudahl [10]. We denote this decision problem as ONLINE CHROMATIC NUMBER. In this paper, we fully resolve this problem:

Theorem 1. *The decision problem ONLINE CHROMATIC NUMBER is PSPACE-complete.*

As is usual in the online computation model, we can view ONLINE GRAPH COLORING as a game between two players, which we call PAINTER (representing the online algorithm) and DRAWER (often called ADVERSARY in the online algorithm literature). In each round DRAWER chooses an uncolored vertex v from G and sends it to PAINTER without telling him to which vertex of G it corresponds, only revealing the edges to the previously sent vertices. Then PAINTER must properly color (“paint”) v , i.e., PAINTER cannot use a color of a neighbor of v . We stress that in this paper PAINTER is restricted to be deterministic. The game continues with the next round until all vertices of G are colored.

Deciding the outcome of many two-player games is PSPACE-complete; among those are Amazons, Checkers and Hex, to name a few. However, in most of these games both players have roughly the same power. This does not hold for ONLINE GRAPH COLORING which is highly asymmetric, since DRAWER has perfect information (knows which vertices are sent and how they are colored), but PAINTER does not. PAINTER may only guess to which part of the graph does the colored subgraph really belong. This is the main difficulty in proving PSPACE-hardness.

Examples. Consider a path P_4 on four vertices. Initially, DRAWER sends two nonadjacent vertices. If PAINTER assigns different colors to them, then these are the first and the third vertex of P_4 , thus the second vertex must get a third color; otherwise they obtained the same color a and they are the endpoints of P_4 , therefore the second and the third vertex get different colors which are not equal to a . In both cases, there are three colors on P_4 and thus $\chi^O(P_4) = 3$, while $\chi(P_4) = 2$.

Note also that we may think of DRAWER deciding where an incoming vertex belongs at some time *after* it is colored provided that the choice still allows for at least one isomorphism to the original G . This is possible only for a deterministic PAINTER.

A particularly interesting class of graphs in terms of χ^O is the class of binomial trees. A binomial tree of order k is defined inductively: The binomial tree of order 0 is a single vertex (the root) and the binomial tree of order k is created by taking two disjoint copies of binomial trees of order $k - 1$, adding an edge between their roots and choosing one of their roots as the root for the resulting tree. Thus P_4 is a binomial tree of order 2 with root on the second vertex of P_4 .

It is not hard to generalize the example of P_4 and show that the online chromatic number of the binomial tree of order k is $k + 1$ [3]. This shows that the ratio between χ^O and χ may be arbitrarily large even for the class of trees.

History and Related Work. The online problem ONLINE GRAPH COLORING has been known since 1976 [1], originally studied in the variant where the

algorithm has no extra information at the start of the input. Bean [1] showed that no online algorithm that is compared to an offline algorithm can perform well under this metric. The notion of online chromatic number was first defined in 1990 by [3].

For the online problem, Lovász, Saks and Trotter [11] show an algorithm with a *competitive ratio* $O(n/\log^* n)$, where the competitive ratio is a ratio of the number of colors used by the online algorithm to the (standard) chromatic number. This was later improved to $O(n \log \log \log n / \log \log n)$ by Kierstad [8] using a deterministic algorithm. There is a better $O(n/\log n)$ -competitive randomized algorithm against an oblivious adversary by Halldórsson [5]. A lower bound on the competitive ratio of $\Omega(n/\log^2 n)$ was shown by Halldórsson and Szegedy [7].

Our variant of ONLINE GRAPH COLORING, where the algorithm receives a copy of the graph at the start, was suggested by Halldórsson [6], where it is shown that the lower bound $\Omega(n/\log^2 n)$ also holds in this model. (Note that the previously mentioned algorithmic results are valid for this model also.)

Kudahl [9] recently studied ONLINE CHROMATIC NUMBER as a complexity problem. The paper shows that the problem is coNP-hard and lies in PSPACE. Later [10] he proved that if some part of the graph is precolored, i.e., some vertices are assigned some colors prior to the coloring game between DRAWER and PAINTER and DRAWER also reveals edges to the precolored vertices for each incoming vertex, then deciding whether $\chi^O(G) \leq k$ is PSPACE-complete. We call this decision problem ONLINE CHROMATIC NUMBER WITH PRECOLORING. The paper [10] conjectures that ONLINE CHROMATIC NUMBER (with no precolored part) is PSPACE-complete too. Interestingly, it is possible to decide $\chi^O(G) \leq 3$ in polynomial time [4].

Keep in mind that while ONLINE GRAPH COLORING is an online problem, ONLINE CHROMATIC NUMBER is an (offline) decision problem of checking whether $\chi^O(G) \leq k$.

Proof Outline. It is not hard to see that ONLINE CHROMATIC NUMBER belongs to PSPACE: The online coloring is represented by a game tree which is evaluated using the Minimax algorithm. This can be done in polynomial space, since the number of rounds in the game is bounded by n , i.e., the number of vertices, and possible moves of each player can be enumerated in polynomial space: PAINTER has at most n possible moves, because it either uses a color already used for a vertex, or it chooses a new color, and DRAWER has at most 2^s moves where s is the number of colored vertices, since it chooses which colored vertices shall be adjacent to the incoming vertex. DRAWER must ensure that sent vertices form an induced subgraph of G , but this can be checked in polynomial space.

Inspired by [10], we prove the PSPACE-hardness of ONLINE CHROMATIC NUMBER by a reduction to Q3DNF-SAT, i.e., the satisfiability of a fully quantified formula in the 3-disjunctive normal form (3-DNF). An example of such a formula is

$$\forall x_1 \exists x_2 \forall x_3 \exists x_4 \dots : (x_1 \wedge x_2 \wedge \neg x_3) \vee (\neg x_1 \wedge x_2 \wedge \neg x_4) \vee \dots$$

The similar problem of satisfiability of a fully quantified formula in the 3-conjunctive normal form is well known to be PSPACE-complete. Since PSPACE is closed under complement, Q3DNF-SAT is PSPACE-complete as well. Note that by an easy polynomial reduction, we can assume that each 3-DNF clause contains exactly three literals.

We show the hardness in several iterative steps. First, in Sect. 2, we present a new, simplified proof of the PSPACE-hardness of ONLINE CHROMATIC NUMBER WITH PRECOLORING in which the sizes of both precolored and non-precolored parts of our construction are linear in the size of the formula.

Then, in Sect. 3, we strengthen the result by reducing the size of the precolored part to be logarithmic in the size of the formula. This is achieved by adding linearly many vertices to our construction.

Finally, in Sect. 4, we show how to remove one precolored vertex and replace it by a non-precolored part, while keeping the PSPACE-hardness proof valid. The cost for removing one vertex is that the size of the graph is multiplied by a constant, but since we apply it only logarithmically many times, we obtain a graph of polynomial size and with no precolored vertex. This will complete the proof of Theorem 1.

We remark that removing the last precolored vertex is the most difficult part of proving PSPACE-hardness of ONLINE CHROMATIC NUMBER. Still, our technique for removing a precolored vertex can be used for any graph satisfying a few assumptions.

We omit some proofs and some technical aspects of our construction due to space restrictions. A preprint version [2] with full details can be found at <https://arxiv.org/abs/1604.05940>.

In our analysis, PAINTER often uses the natural greedy algorithm FIRSTFIT, which is ubiquitous in the literature (see [6, 11]):

Definition 1. *The online algorithm FIRSTFIT colors an incoming vertex u using the smallest color not present among colored vertices adjacent to u .*

2 Construction with a Large Precolored Part

Our first construction will reduce the PSPACE-complete problem Q3DNF-SAT to ONLINE COLORING WITH PRECOLORING with a large precolored part. Given a fully quantified formula Q in the 3-disjunctive normal form, we will create a graph G_1 that will simulate this formula. We assume that the formula contains n variables x_i , ($1 \leq i \leq n$) and m clauses C_a , ($1 \leq a \leq m$), and that variables are indexed in the same order as they are quantified.

Our main resource will be a large precolored clique K_{col} on k vertices and naturally using k colors; the number k will be specified later. Using such a precolored clique, we can restrict the allowed colors on a given uncolored vertex v by connecting it with the appropriate vertices in K_{col} , i.e., we connect v to all vertices in K_{col} which do not have a color allowed for v .

For simplicity we use the precoloring in the strong sense, i.e., PAINTER is able to recognize which vertex in K_{col} is which. We use this to easily recognize colors.

However, it is straightforward to avoid the strong precoloring by modifying the precolored part; for example by creating i independent and identical copies of the i -th vertex in K_{col} , each having the same color and the same edges to other vertices in K_{col} and the rest of the graph. With such a modification, PAINTER would be able to recognize each color by the number of its vertices in K_{col} .

Each vertex in K_{col} thus corresponds to a color. Colors used by PAINTER are naturally denoted by numbers $1, 2, 3, \dots, k$, but we shall also assign meaningful names to them.

We want to construct a graph G_1 that has the online chromatic number k if and only if the quantified 3-DNF formula can be satisfied. See Fig. 1 for an example of G_1 and an overview of our construction. We use the following gadgets for variables and clauses:

1. For a variable x_i which is quantified universally, we will create a gadget consisting of *universal* vertices $x_{i,t}$ and $x_{i,f}$, connected by an edge. The vertex $x_{i,t}$ represents the positive literal x_i , while $x_{i,f}$ represents the negative literal $\neg x_i$. Both vertices have exactly two allowed colors: set_i and $unset_i$. If $x_{i,t}$ is assigned the color set_i , it corresponds to setting the variable x_i to 1, and vice versa.

Note that if DRAWER presents a vertex $x_{j,t}$ to PAINTER, PAINTER is able to recognize that it is a vertex corresponding to the variable x_j , but it is not able to recognize whether it is the vertex $x_{j,t}$ or $x_{j,f}$.

2. For a variable x_j which is quantified existentially, we will create a gadget consisting of three *existential* vertices $x_{j,t}$ (for the positive literal x_j), $x_{j,f}$ (for the literal $\neg x_j$) and $x_{j,h}$ (the helper vertex), connected as a triangle. Coloring of the first two variables also corresponds to setting the variable x_j to true or false, but in a different way: $x_{j,t}$ has allowed colors $set_{j,t}$ and $unset_{j,t}$, while $x_{j,f}$ has allowed colors $set_{j,f}$ and $unset_{j,f}$. We want to avoid both $x_{j,t}$ and $x_{j,f}$ to have the color of type set , and so the “helper” vertex $x_{j,h}$ can be colored only by $set_{j,t}$ or $set_{j,f}$.

Note that the color choice for the vertices of x_j means that if PAINTER is presented any vertex of this variable, PAINTER can recognize it and decide whether to set x_j to 1 (and color accordingly) or to 0.

We call existential and universal vertices together *variable vertices*.

3. For each clause C_a , we will add four new vertices. First, we create a vertex $l_{a,i}$ for each literal in the clause, which is connected to one of the vertices $x_{i,t}$ and $x_{i,f}$ corresponding to the sign of the literal. For example if $C_a = (x_i \wedge \neg x_j \wedge x_k)$, then $l_{a,i}$ is connected to $x_{i,t}$, $l_{a,j}$ is connected to $x_{j,f}$ and $l_{a,k}$ to $x_{k,t}$. The allowed colors on a vertex $l_{a,i}$ are $\{f_a, unset_i\}$.

Finally, we add a fourth vertex d_a connected to the three vertices $l_{a,i}, l_{a,j}, l_{a,k}$. This vertex can be colored only using the color f_a or the color $false_a$. The color $false_a$ is used to signal that this particular clause is evaluated to be 0. If the color f_a is used for the vertex d_a , this means that the clause is evaluated to 1, because f_a is not present on any of $l_{a,i}, l_{a,j}, l_{a,k}$, thus they have colors of type $unset_i$ and their neighbors corresponding to literals have colors of type set .

4. The last vertex we add to the construction will be F , a final vertex. The vertex F is connected to all the vertices d_a corresponding to the clauses. The allowed colors of the vertex F are $\text{false}_1, \text{false}_2, \text{false}_3, \dots, \text{false}_m$. This final vertex corresponds to the final evaluation of the formula. If all clauses are evaluated to 0, the vertex F has no available color left and must use a new color.

We have listed all the vertices and colors in our graph G_1 and the functioning of our gadgets, but we will need slightly more edges. The reasoning for the edges is as follows: If DRAWER presents any vertex of the type $l_{a,i}, d_a$ or F before presenting the variable vertices, or in the case when the variable vertices are presented out of the quantifier order, we want to give an advantage to PAINTER so it can finalize the coloring.

This will be achieved by allowing PAINTER to treat all remaining universal vertices as existential vertices, i.e., PAINTER can recognize which of the two universal vertices $x_{j,t}, x_{j,f}$ corresponds to setting x_j to 1.

To be precise, we add the following edges to G_1 :

- Every existential vertex $x_{j,t}, x_{j,f}, x_{j,h}$ is connected to all previous universal vertices $x_{i,t}$, that is to all such $x_{i,t}$ for which $i < j$.
- Every universal vertex $x_{j,t}, x_{j,f}$ is connected to all previous universal vertices $x_{i,t}$ such that $i < j$.
- Every vertex of type $l_{a,i}$ is connected to all the universal vertices $x_{i',t}$ for $i' \neq i$. Note that $l_{a,i}$ is connected either to $x_{i,t}$, or to $x_{i,f}$; we do not add an edge to such vertices.
- Every vertex of type d_a is connected to all universal vertices $x_{i,t}$ for all i .
- The vertex F is connected to all the universal vertices $x_{i,t}$ for all i .

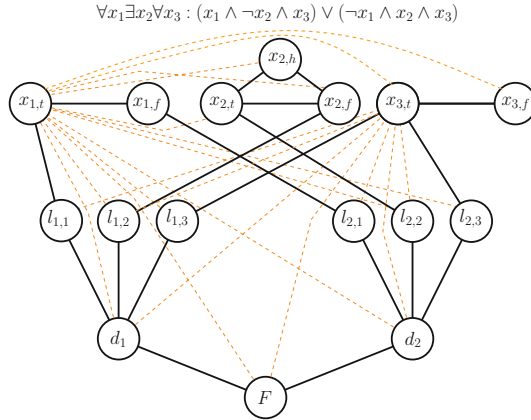


Fig. 1. The construction for a sample formula. The thick black edges are the normal edges of the construction, and the dashed orange edges are the additional edges that guarantee precedence of vertices. The lists of allowed colors of each vertex are not listed in the figure. (Color figure online)

We call all non-precolored vertices the *gadgets* for variables and clauses.

The number of colors allowed for PAINTER (the same as the size of K_{col}) is $k = 2m + 2n_{\forall} + 3n_{\exists}$ where m is the number of clauses, n_{\forall} the number of universally quantified variables and n_{\exists} the number of existentially quantified variables.

The analysis of our construction is fairly straightforward (see [2] for details).

3 Construction with a Precolored Part of Logarithmic Size

We now make a step to the general case without precoloring by reducing the size of the precolored part so that it has only logarithmic size. Our construction is based on the one with a large precolored part; namely, all the vertices $x_{i,t}, x_{i,f}, x_{j,t}, x_{j,f}, x_{j,h}, l_{a,i}, d_a, F$ (the gadgets for variables and clauses) and the whole color clique K_{col} will be connected the same way. Let G_1 denote the gadgets for variables and clauses and K_{col} .

Since K_{col} is now not precolored and DRAWER may send it after the gadgets, we help PAINTER by a structure for recognizing vertices in G_1 or for saving colors.

We remark that there is also a simpler construction with a logarithmic number of precolored vertices. If we just add precolored vertices to recognize vertices in G_1 , the following proof would work and be easier. However, when we replace a precolored vertex v by some non-precolored graph in Sect. 4, we will use some conditions on the graph G_2 that this construction would not satisfy.

3.1 Nodes

Our structure will consist of many small *nodes*, all of them have the same internal structure, only their adjacencies with other vertices vary.

Each node consists of three vertices and a single edge; vertices are denoted by p_1, p_2, p_3 and the edge leads between p_2 and p_3 . We call the vertices p_1 and p_2 the *lower partite set* of the node, p_3 form the *upper partite set*. See Fig. 2 for an illustration of a node. Clearly, the online chromatic number of a node is two. The intuition behind the nodes is as follows:

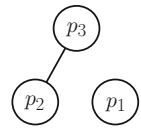


Fig. 2. Node

- If DRAWER presents vertices of a node in the correct way, PAINTER needs to use two colors in the lower partite set of every node.
- No color can be used in two different nodes.
- Each vertex $v \in G_1$ (in the gadgets and in K_{col}) has its own associated node A . If the vertex p_3 from A does not arrive before v is sent, PAINTER can color p_3 and v with the same color, thus save a color. Otherwise, PAINTER can use the node to recognize v .
- Universal vertices $x_{i,t}, x_{i,f}$ for each universally quantified variable x_i should be distinguishable only by the same vertices as in the previous section. Therefore they are both associated with the same two nodes.

Let N be the number of vertices in G_1 . We create N nodes, denoted by A_1, \dots, A_N , one for each vertex in G_1 . For any two distinct nodes A_i and A_j ($i \neq j$), there is an edge between each vertex in A_i and each vertex in A_j . Therefore, no color can be used in two nodes.

We have noted above that each node is associated with a vertex; we now make the connection precise. Let v_1, \dots, v_N be the vertices in G_1 (in an arbitrary order). Then we say that A_i *identifies* the vertex v_i . Moreover, if v_i is a vertex $x_{k,t}$ or $x_{k,f}$ for a universally quantified variable x_k and v_j is the other vertex, then A_j also identifies v_i and A_i also identifies v_j . Thus each node identifies one or two vertices and each vertex is identified by one or two nodes.

Edges between a vertex v in the original construction G_1 and a node depend on whether the node identifies v , or not. For a vertex $v \in G_1$ and for a node A , if A identifies v , we connect only the whole lower partite set of A to v , i.e., we add two edges from v to both p_1 and p_2 of A . Otherwise, we add three edges – one between v and every vertex in A .

3.2 Precolored Vertices

The only precolored part P of the graph is intended for distinguishing nodes. Since there are N nodes in total, we have $p = \lceil \log_2 N \rceil$ precolored vertices z_1, z_2, \dots, z_p with no edges among them. Precolored vertices have a color that may be used later for coloring G_1 (the gadgets and K_{col}). For simplicity, we again use the precoloring in the strong sense, i.e., PAINTER is able to recognize which precolored vertex is which.

We connect all vertices in the node A_i to z_j if the j -th bit in the binary notation of i is 1; otherwise z_j is not adjacent to any vertex in A_i .

Clearly, the node to which an incoming vertex belongs can be recognized by its adjacency to the precolored vertices. Note that a vertex from nodes is connected to at least one precolored vertex and there is no edge between G_1 and precolored vertices.

So far, we have introduced all vertices and edges in our construction of the graph G_2 . We omit the rest of the analysis due to space restrictions; see [2] for details.

4 Removing Precoloring

In this section we show how to replace one precolored vertex by a large nonprecolored graph whose size is a constant factor of the size of the original graph, while keeping PAINTER's winning strategy in the case of a satisfiable formula. DRAWER's winning strategy in the other case is of course preserved also and easier to see. We prove the following lemma which holds for all graphs with precolored vertices satisfying a few assumptions.

Lemma 1. *Let G be a graph with precolored subgraph G_p created from a fully quantified formula ϕ , and let $v_p \in G_p$ be a precolored vertex of G .*

Let D be the induced subgraph with all nonprecolored vertices that are not connected to v_p and let E be the induced subgraph with all nonprecolored vertices that are connected to v_p .

Let k be an integer. Assume that the following holds:

1. $\chi^O(G) \leq k$ if and only if ϕ is satisfiable,
2. in the winning strategy of PAINTER in the case if ϕ is satisfiable, PAINTER can color E using FIRSTFIT before two nonadjacent vertices from D arrive. Moreover, in this case if FIRSTFIT assigns the same color to a vertex in D and to a vertex in E before two nonadjacent vertices from D arrive, PAINTER can still color G using k colors.

Then there exists an integer k' and a graph G' with the following properties:

- G' has only $|V(G_p)| - 1$ precolored vertices, and $|V(G')| \leq 25|V(G)|$,
- G' can be constructed from G in polynomial time,
- it holds that $\chi^O(G') \leq k'$ if and only if ϕ is satisfiable.

Theorem 1 follows by an iterative application of Lemma 1; the details of this application can be found in [2].

Construction of G' . Let N be the total number of vertices in D and E and let $S = 8N$. Our graph G' consists of precolored part $G'_p := G_p \setminus \{v_p\}$, graphs D and E and three huge cliques A, B and C of size S ; cliques A, B and C together form a *supernode*. We keep the edges inside and between D and E and the edges between G'_p and $D \cup E$ as they are in G .

We add a complete bipartite graph between cliques B and C , i.e., $B \cup C$ forms a clique of size $2S$. No vertex in A is connected to B or C . In other words, the supernode is created from a node by replacing each vertex by a clique of size S and the only edge in the node by a complete bipartite graph.

There are no edges between the supernode (cliques A and $B \cup C$) and a precolored vertex in G'_p . It remains to add edges between the supernode and $D \cup E$. There is an edge between each vertex in E and each vertex in the supernode, while every vertex in D is connected only to the whole A and B , but not to any vertex in C . The fact that D and C are not adjacent at all is essential in our analysis. Our construction is depicted in Fig. 3.

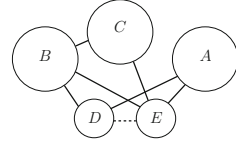


Fig. 3. Our construction G' . (The remaining precolored vertices are not shown.)

Proof (Proof of Lemma 1). Let G' be the graph defined as above. Note that the number of vertices in G' is at most $25|V(G)|$, G' can be constructed from G in polynomial time and G' has only $|V(G_p)| - 1$ precolored vertices. Therefore, it remains to prove $\chi^O(G') \leq k'$ if and only if ϕ is satisfiable for some k' . We set k' to $k + 2S$, since there will be at most $2S$ colors used in the supernode.

Assuming that ϕ is not satisfiable, it is straightforward to design a winning strategy for DRAWER on G' ; we only need to adapt the approach of Sects. 2 and 3. See [2] for a full description of the strategy.

In the rest of this section we focus on the opposite direction: assuming that ϕ is satisfiable, we show that PAINTER can color G' with k' colors regardless of the strategy of DRAWER. In the following, when we refer to the colored part of G' , we do not take precolored vertices into account. PAINTER actually does not look at precolored vertices unless it uses its winning strategy for coloring G , which exists by the assumptions.

Intuition. At the beginning PAINTER has too little data to infer anything about the vertices. Therefore, PAINTER shall wait for two nonadjacent vertices from D and for two large cliques (larger than $S/2$) with a small intersection. Before such vertices arrive, it will color greedily.

Note that the greedy coloring algorithm eventually stops before everything is colored. Having two large cliques, one mostly from A and the other mostly from $B \cup C$, and two nonadjacent vertices from D , PAINTER is able to recognize where an incoming vertex belongs. Therefore, PAINTER can use the supernode like a precolored vertex and colors the remaining vertices from D and E by its original winning strategy on G .

This approach may fail if a part of D is already colored by PAINTER's application of the greedy strategy. To remedy this, we prove that colors used on D so far are also used in C or E , or will be used on C later.

The other obstacle is that PAINTER might not be able to distinguish between one clique from D and vertices in A if nothing from B arrives. Nevertheless, each vertex u in such a "hidden" clique is connected to all other colored vertices in D and to the whole colored part of E , otherwise it would be distinguishable from vertices in A . Hence, it does not matter on the color of u .

In summation, the sheer size of the supernode should allow PAINTER to be able to use it as if it would be precolored. Still, we need to allow for some small margin of error. This leads us to the following definition:

Definition 2. Let N be the number of vertices of $D \cup E$ as in the construction of G' . For subgraphs $X, Y \subseteq G'$, we say that X is *practically a subgraph of* Y if $|V(X) \setminus V(Y)| \leq N$, and X is *practically disjoint with* Y if $|V(X) \cap V(Y)| \leq N$.

We also say that a vertex v is *practically universal to a subgraph* $X \subseteq G'$ if it is adjacent to all vertices in X except at most N of them. Similarly, we say a vertex v is *practically independent of a subgraph* $X \subseteq G'$ if v has at most N neighbors in X .

At first, the player PAINTER uses the following algorithm for coloring incoming vertices, which may stop when it detects two useful vertices d_1 and d_2 :

Algorithm WAITFOR D : For an incoming vertex u sent by DRAWER:

1. Let G'_A be the revealed part of G' (i.e., colored vertices and u , but not precolored vertices)
2. Find a maximum clique in G'_A and denote it as K_1 .
3. Find a maximum clique in G'_A practically disjoint with K_1 and denote it as K_2 .
4. If $|K_2| \geq S/2$ and there are two nonadjacent vertices d_1 and d_2 in G'_A which are both *not* practically universal to K_1 or both *not* practically universal to K_2 :
5. Stop the algorithm.
6. Otherwise, color u using FIRSTFIT.

While the algorithm may seem to use a huge amount of computation for one step, we should realize that we are not concerned with time complexity when designing the strategy for PAINTER. In fact, even a non-constructive proof of existence of a winning strategy would be enough to imply existence of a PSPACE algorithm for finding it – we have observed already in Sect. 1 that ONLINE CHROMATIC NUMBER lies in PSPACE.

Let v be the incoming vertex u when WAITFORD stops; note that v is not colored by the algorithm and v can be from any part of G' .

One of the cliques K_1 and K_2 is practically a subgraph of $B \cup C$ and we denote this clique by K_{BC} . The other clique must be practically a subgraph of A and we denote it by K_A . (Keep in mind that both cliques may contain up to N vertices from $D \cup E$.) We remark that some vertices from C must have arrived, as A and B alone are indistinguishable by Step 4 of WAITFORD. By the same argument, the player PAINTER knows whether $K_1 = K_A$ or $K_1 = K_{BC}$.

Let d_1 and d_2 be the nonadjacent vertices that caused the algorithm to stop. We observe that $d_1, d_2 \in D$ by eliminating all other possibilities:

- Neither of d_1 and d_2 can be from E , since any vertex of E is practically universal to both cliques.
- Both d_1 and d_2 cannot be from $B \cup C$ or both from A , as they would be adjacent.
- If d_1 is in $B \cup C$ and d_2 in A , then we have a contradiction with the fact that d_1 and d_2 are not practically universal to the same clique.
- If $d_1 \in D$ and d_2 would be from A or B , then d_1 and d_2 are adjacent.
- Finally, if $d_1 \in D$ and $d_2 \in C$, then the clique to which they are not practically universal cannot be the same for both, since d_1 is universal to the whole A and d_2 to the whole $B \cup C$.

Having cliques K_A and K_{BC} and vertices $d_1, d_2 \in D$, PAINTER uses the following rules to recognize where an incoming or a colored vertex u belongs:

- If u is practically universal to both K_{BC} and K_A , then $u \in E$.
- If u is practically universal to K_{BC} and practically independent of K_A and u is adjacent to d_1 , then $u \in B$.
- If u is practically universal to K_{BC} and practically independent of K_A , but there is no edge between d_1 and u , then $u \in C$.
- If u is not practically universal to K_{BC} , but it is practically universal to K_A , then $u \in A$ or $u \in D$.
 - Among such vertices, if there is a vertex not adjacent to u or u is not adjacent to a vertex in E or u is adjacent to a vertex in B , then $u \in D$; we say that such u is *surely in D*.
 - Otherwise, PAINTER cannot yet recognize whether $u \in A$ or $u \in D$.

The reader should take a moment to verify that indeed, the set of rules covers all possible cases for u .

Let $\tilde{A}, \tilde{B}, \tilde{C}, \tilde{D}, \tilde{E}$ be the colored parts of G' when WAITFORD stops. We observe that in the last case of the recognition the vertices from \tilde{D} which are

indistinguishable from A form a clique; we denote it by K_D . Note that all vertices in K_D are connected to all vertices surely from D that arrived and K_D contains all vertices in \tilde{D} that are not surely in D . We stress that PAINTER does not know K_D or even its size.

Intuition for the Next Step. Since PAINTER can now recognize the parts of the construction (with an exception of K_D), we may basically use the winning strategy for PAINTER on G and FIRSTFIT on the rest. More precisely, PAINTER creates a virtual copy of G , adds vertices into it and simulates the winning strategy on this virtual graph.

Our main problem is that some part of D (namely \tilde{D}) is already colored. We shall prove that if \tilde{D} is not a clique, PAINTER can ignore colors used in \tilde{D} (but not the colors that it will use on D), as they are already present in C or E or they may be used later in C . If \tilde{D} is a clique, it may be the case that C and A arrived fully and have the same colors, thus PAINTER cannot ignore colors on \tilde{D} .

Another obstacle in the simulation is K_D , the hidden part of D . To overcome this, PAINTER tries to detect vertices in K_D and reclassify them as surely in D . PAINTER shall keep that all vertices in K_D are connected to all currently colored vertices in D and E , therefore it does not matter much on colors in K_D .

When PAINTER discovers a vertex from K_D , it adds the vertex immediately to its simulation of G . On the other hand, the size of K_D increases when DRAWER sends a vertex from D which is indistinguishable from A .

The details of the algorithm used by PAINTER to finish the coloring of G' using k' colors are omitted due to space restrictions and can be found in [2]. \square

Acknowledgments. The authors thank Christian Kudahl and their supervisor Jiří Sgall for useful discussions on the problem.

References

1. Bean, D.R.: Effective coloration. *J. Symbolic Logic* **41**(2), 469–480 (1976)
2. Böhm, M., Veselý, P.: Online chromatic number is PSPACE-complete, arXiv preprint (2016). <https://arxiv.org/abs/1604.05940>
3. Gyárfás, A., Lehel, J.: First fit and on-line chromatic number of families of graphs. *Ars Combinatoria* **29C**, 168–176 (1990)
4. Gyárfás, A., Kiraly, Z., Lehel, J.: On-line graph coloring and finite basis problems. In: *Combinatorics: Paul Erdos is Eighty*, vol. 1, pp. 207–214 (1993)
5. Halldórsson, M.M.: Parallel and on-line graph coloring. *J. Algorithms* **23**, 265–280 (1997)
6. Halldórsson, M.M.: Online coloring known graphs. *Electron. J. Combinatorics* **7**(1), R7 (2000)
7. Halldórsson, M.M., Szegedy, M.: Lower bounds for on-line graph coloring. *Theor. Comput. Sci.* **130**(1), 163–174 (1994)
8. Kierstad, H.: On-line coloring k -colorable graphs. *Israel J. Math.* **105**, 93–104 (1998)
9. Kudahl, C.: On-line graph coloring. Master’s thesis, University of Southern Denmark (2013)

10. Kudahl, C.: Deciding the on-line chromatic number of a graph with pre-coloring is PSPACE-complete. In: Paschos, V.T., Widmayer, P. (eds.) CIAC 2015. LNCS, vol. 9079, pp. 313–324. Springer, Heidelberg (2015)
11. Lovász, L., Saks, M., Trotter, W.T.: An on-line graph coloring algorithm with sublinear performance ratio. *Ann. Discrete Math.* **43**, 319–325 (1989)

Combinatorial Algorithms

27th International Workshop, IWOCA 2016, Helsinki,

Finland, August 17-19, 2016, Proceedings

Mäkinen, V.; Puglisi, S.J.; Salmela, L. (Eds.)

2016, XX, 462 p. 100 illus., Softcover

ISBN: 978-3-319-44542-7