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 *online chromatic number* which is analogous to the standard chromatic number of a graph.

Definition 1. The *online chromatic number* $\chi^O(G)$ of a graph G is the smallest number k such that there exists a deterministic online algorithm which is able to color the specified graph G using k colors for any incoming order of vertices.

The online chromatic number has been studied since 1990 [6]. One of the main open problems in the area is the computational complexity of deciding whether $\chi^O(G) \leq k$ for a specified simple graph G, given G and k on input; see e.g. Kudahl [13].

[*]Supported by project 17-09142S of GA ČR and by the GAUK project 634217. A preliminary version of this work appeared in [3].

Definition 2. The Online Chromatic Number problem is as follows:
Input: An undirected simple graph G and an integer k.
Goal: Decide whether $\chi^O(G) \leq k$.

In this paper, we fully resolve the computational complexity of this problem:

Theorem 1.1. *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 without any information 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.

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 Drawer's 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, denoted B_k, is defined inductively: B_0 is a single vertex (the root) and B_k is created by taking two disjoint copies of B_{k-1}, adding an edge between their roots and choosing one of their roots as the root for the resulting tree. Thus P_4 with the root on the second vertex is B_2.

It is not hard to generalize the example of P_4 and show $\chi^O(B_k) = k + 1$ [6]. This shows that the ratio between χ^O and χ may be arbitrarily large even for the class of trees.

History. 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 [6].

For the online problem, Lovász, Saks and Trotter [14] show an algorithm with a *competitive ratio* $O(n/\log^* n)$, where the competitive ratio is the 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 [11] using a deterministic algorithm. There is a better $O(n/\log n)$-competitive randomized algorithm against an oblivious adversary by Halldórsson [8]. A lower bound on the competitive ratio of $\Omega(n/\log^2 n)$ even for randomized algorithms against an oblivious adversary was shown by Halldórsson and Szegedy [10].

Our variant of Online Graph Coloring, where the algorithm receives a copy of the graph at the start, was suggested by Halldórsson [9], 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 [12] recently studied Online Chromatic Number as a complexity problem. The paper shows that the problem is coNP-hard and lies in PSPACE. Later [13] 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 [13] 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 [7].

Related work. Deciding the outcome of many two-player games is PSPACE-complete; among those are (generalizations of) Amazons, Checkers and Hex, to name a few.

The closest PSPACE-complete two-player game to our model is arguably Sequential Coloring Game, where two players color vertices in a fixed order and the first player to use more than k colors loses the game. This game was defined and analyzed by Bodlaender [2].

However, in all of the games mentioned above, 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.

An example of an asymmetric two-player game somewhat related to Online Graph Coloring is the Online Ramsey Game in which Builder draws edges and Painter colors each edge either red, or blue. Builder wins if it forces a monochromatic copy of a graph H, otherwise Painter wins. The condition for Builder is that at the end the revealed graph must belong to a certain class of graphs. Deciding whether Builder wins was recently shown to be PSPACE-complete [5], however, the proof assumes that some of the edges may be precolored.

Another recently studied asymmetric model is the Chooser-Picker Positional Game. In it, the player Chooser selects a pair of objects, and the player Picker selects one of them for itself, leaving the other object for the player Chooser. The winning condition is then similar to Maker-Breaker games, such as Online Ramsey Game described above. A recent paper [4] proves NP-hardness for this problem, but PSPACE-hardness remains open.

Proof outline. The fact that Online Chromatic Number belongs to PSPACE is not hard to see: 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 [13], we prove the PSPACE-hardness of Online Chromatic Number by a reduction from 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

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

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 Section 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 Section 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 Section 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 1.1.

In our analysis, Painter often uses the natural greedy algorithm FirstFit, which is ubiquitous in the literature (see [14], [9]):

Definition 3. The online algorithm FirstFit colors an incoming vertex u using the smallest color not present among colored vertices adjacent to u.

We remark that removing the last precolored vertex is the most difficult part of proving PSPACE-hardness of Online Chromatic Number. While there might be an easier way how to remove the penultimate precolored vertex (using the last precolored vertex) and similarly for previous precolored vertices, for simplicity we use the same technique for removing all precolored vertices as for removing the last precolored vertex. Also, our technique can be used for any graph satisfying an assumption.

We note that if we would give Painter some advantage like parallel edges, precolored vertices (as in [13]) or something that helps Painter distinguish different parts of the graph, the proof of PSPACE-hardness would be much simpler. However, our theorem holds for simple graphs without any such help.

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}. We also remark that working with the strong precoloring is easier in the reduction, and since we eventually obtain a graph without a precolored vertex, it does not matter which precoloring we use.

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

We now 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 Figure 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 $\forall$-*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 $x_{i,t}$ to Painter, Painter is able to recognize that it is a vertex corresponding to the variable x_i, but it is not able to recognize whether it is the vertex $x_{i,t}$ or $x_{i,f}$.
2. For a variable x_j which is quantified existentially, we will create a gadget consisting of three $\exists$-*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 $\text{set}_{j,t}$ and unset_j, while $x_{j,f}$ has allowed colors $\text{set}_{j,f}$ and unset_j. 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 $\text{set}_{j,t}$ or $\text{set}_{j,f}$.
   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 $\exists$-vertices and $\forall$-vertices together *variable vertices.*
3. For each clause C_a, we will add four new vertices.
   (a) 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 $\{f_a, \text{unset}_i\}$.
   (b) 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 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 $\text{false}_1, \text{false}_2, \text{false}_3, \ldots, \text{false}_m$. 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 $\forall$-vertices as $\exists$-vertices, i.e., Painter can recognize which of the two $\forall$-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 $\exists$-vertex $x_{j,t}, x_{j,f}, x_{j,h}$ is connected to all previous $\forall$-vertices $x_{i,t}$, that is to all such $x_{i,t}$ for which $i < j$.
- Every $\forall$-vertex $x_{j,t}, x_{j,f}$ is connected to all previous $\forall$-vertices $x_{i,t}$ such that $i < j$. (We remark that these edges are not necessary, but make the following analysis simpler.)
- Every vertex of type $l_{a,i}$ is connected to all the $\forall$-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 a new edge to such vertices.
- Every vertex of type d_a is connected to all the $\forall$-vertices $x_{i,t}$ for all i.
- The vertex F is connected to all the $\forall$-vertices $x_{i,t}$ for all i.

See Figure 1 for an example of our graph G_1. We call all non-precolored vertices the *gadgets* for variables and clauses.

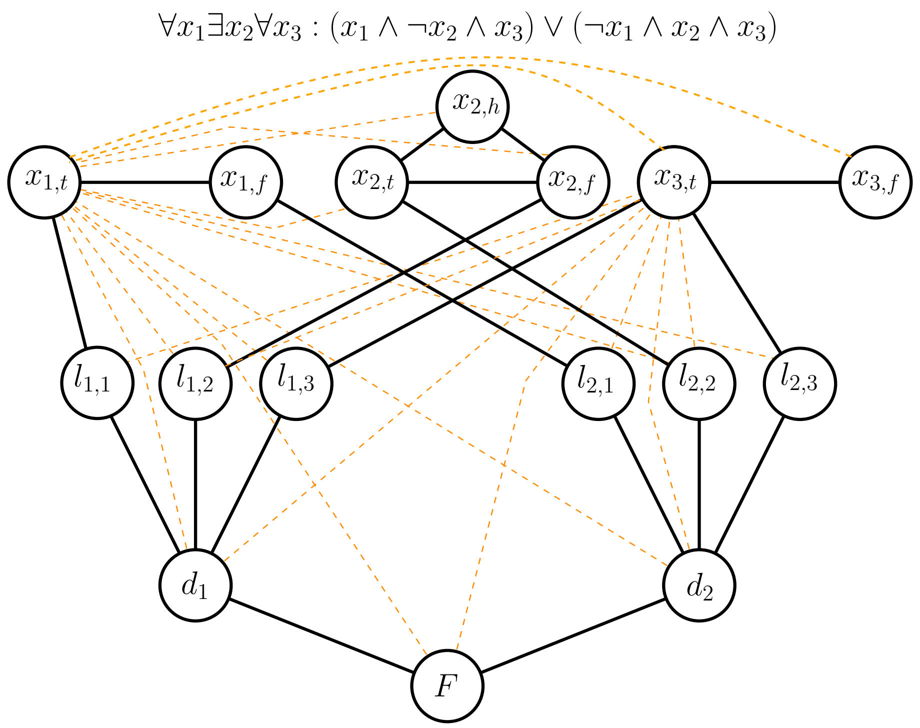

Figure 1: The construction for a sample formula. The thick black edges are the normal edges of the construction, and the dashed orange edges are the additional edges that guarantee precedence of vertices. The lists of allowed colors of each vertex are not listed in the figure.

It is easy to see that any two vertices outside K_{col} have different sets of allowed colors except $\forall$-vertices $x_{i,t}$ and $x_{i,f}$.

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 next two lemmas contain the analysis of our construction.

Lemma 2.1. *For a given fully quantified formula in the 3-DNF form that is not satisfiable,* Drawer *can force* Painter *to use $k+1$ colors.*

Proof. If anytime during the game Painter uses a color not present in the color clique K_{col}, the lemma is proven. We therefore assume this does not happen and show that the vertex F cannot be colored with any color present in the precolored clique K_{col}.

Drawer's strategy is to first present the vertices x_* in the the order in which the variables are quantified in the formula. Whenever Drawer sends a $\forall$-vertex $x_{i,t}$ or $x_{i,f}$, Painter is not able to detect whether it is setting the value of x_i to 1 or 0. As the formula is not satisfiable, Drawer can therefore present these vertices in a sequence such that Painter chooses the value of x_i so that the final evaluation is false.

After the vertices of the variables, Drawer will present the vertices $l_{a,i}$ for each clause, then the vertex d_a for each clause, and finally the vertex F.

We now know that all clauses are evaluated to 0. This means that at least one of the vertices $l_{a,i}, l_{a,j}, l_{a,k}$ of each clause a is assigned the color f_a. It follows that the vertex d_a of this clause must be assigned the color false$_a$. This holds for all clauses, thus the vertex F has a neighbor of the color false$_a$ for each clause a. Hence, F cannot get any of the k colors allowed and needs to be assigned a new color, which completes the proof. □

Lemma 2.2. *For a given fully quantified formula in the 3-DNF form that is satisfiable, and for any order of sending vertices by* Drawer, Painter *has a strategy that uses k colors.*

Proof. Painter's goal is to use the knowledge of a satisfiable evaluation to paint the vertices with few colors. We note the following two observations. The first follows easily from the sets of allowed colors.

Observation 2.3. *If* Drawer *presents a vertex,* Painter *is able to recognize it by the set of edges to the clique K_{col}, i.e., by the set of allowed colors, with the exception of $\forall$-vertices $x_{i,t}, x_{i,f}$ for a universally quantified variable x_i.*

Observation 2.4. *If* Drawer *presents a vertex of type $l_{a,i}, d_a$ or F before any of the $\forall$-vertices $x_{i,t}, x_{i,f}$ is presented for a universally quantified variable x_i,* Painter *will be able to distinguish the vertex $x_{i,t}$ from the vertex $x_{i,f}$ and therefore choose the assignment of x_i.*

Furthermore, for $\forall$-vertices $x_{i,t}, x_{i,f}$ and for a vertex v of a variable x_j with $i < j$, if Drawer *presents v before any of these $\forall$-vertices is sent,* Painter *is then able to distinguish the $\forall$-vertices $x_{i,t}$ and $x_{i,f}$.*

The second observation is easy to see by noting that presenting $v, l_{a,i}, d_a$ or F means that it has an edge with $x_{i,t}$, but not with $x_{i,f}$ (or, in the case of $l_{a,i}$, it may have an edge with $x_{i,f}$, but not with $x_{i,t}$, but this is symmetric). This means that we can distinguish $x_{i,t}$ from $x_{i,f}$ by the presence or absence of an edge.

We continue with the proof of the lemma. We say that a variable x_i is *set*, if at least one of its vertices was colored or if Painter has assigned a value to it (e.g., if a vertex of a variable x_j was sent before a vertex of x_i for $j > i$); otherwise x_i is *unset.* Until Drawer presents a vertex of type $l_{a,i}, d_a$, or F, Painter colors an incoming vertex v that corresponds to a variable x_i using the following strategy:

- Let U be the set of all unset variables x_j with $j < i$, i.e., all previous unset variables. Painter chooses an assignment for all variables in U according to satisfiability of the formula. Then Painter can color vertices for variables in U according to this assignment, since it is now able to distinguish $\forall$-vertices of each universally quantified variable in U. Therefore every variable in U becomes set.
- If the variable x_i is not set and it is quantified universally, Painter chooses an arbitrary allowed color for v (set$_i$ or unset$_i$), thus x_i becomes set.
- If x_i is not set and it is quantified existentially, Painter knows how to set this variable to satisfy the formula, thus Painter colors v according to the value of x_i, since v can be recognized.
- If x_i is set and Painter can recognize which vertex is v, it colors v according to the setting of x_i.
- If x_i is set and Painter cannot recognize v, then x_i must be quantified universally and the other vertex for x_i is colored, therefore there is a single color left for v. (Note that in this case no vertex for a variable x_j with $j > i$ arrived.)

When the vertex u of type $l_{a,i}, d_a$, or F is sent, let U be the set of all unset variables. Painter chooses an assignment for all variables in U according to satisfiability of the formula. Then Painter can color vertices for variables in U according to this assignment, since it is now able to distinguish vertices of each universally quantified variable in U. Then all variables are set and Painter decides how to color all remaining vertices in the graph using the following rules:

- A variable vertex obtains color according to the setting of the corresponding variable.
- A vertex of type $l_{a,i}$ gets unset$_i$ if its adjacent variable vertex $x_{i,t}$ or $x_{i,f}$ has a color of type set; otherwise it gets f_a
- A vertex of type d_a obtains color f_a if all of its adjacent vertices of type $l_{a,i}$ have colors of type unset, i.e., the clause a evaluates to 1; otherwise it obtains color false$_a$.
- The final vertex F obtains a color false$_a$ for a clause a that evaluates to 1. Since Painter set variables such that the formula is satisfied, there must be such clause.

Now, Painter colors an incoming vertex v with the color assigned to it using these rules. Hence, no matter in which order Drawer sends the vertices, the graph is colored using k colors as desired. □

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} now starts uncolored 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 a clique of logarithmically many 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 Section 4, we will use some conditions that this simple construction would not satisfy. Namely, we shall need that precolored vertices are not adjacent to G_1.

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 Figure 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:

- If Drawer presents vertices of a node in the correct way, Painter needs to use two colors in the lower partite set of the 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 saving a color. Otherwise, Painter can use the node to recognize v.
- $\forall$-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.

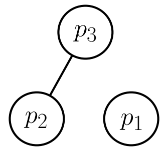

Figure 2: Node

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 $x_{\ell,t}$ or $x_{\ell,f}$ for a universally quantified variable x_ℓ 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 the same 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. To summarize, our graph G_2 consists of three *parts*:

1. The graph G_1 from the previous section consisting of the color clique K_{col} and of the gadgets for variables and clauses, i.e., vertices $x_{i,t}, x_{i,t}, x_{j,t}, x_{j,f}, x_{j,h}, l_{a,i}, d_a, F$;
2. the nodes for recognizing vertices in G_1;
3. the precolored vertices P for recognizing the nodes.

See Figure 3 for a simplified example of the graph G_2.

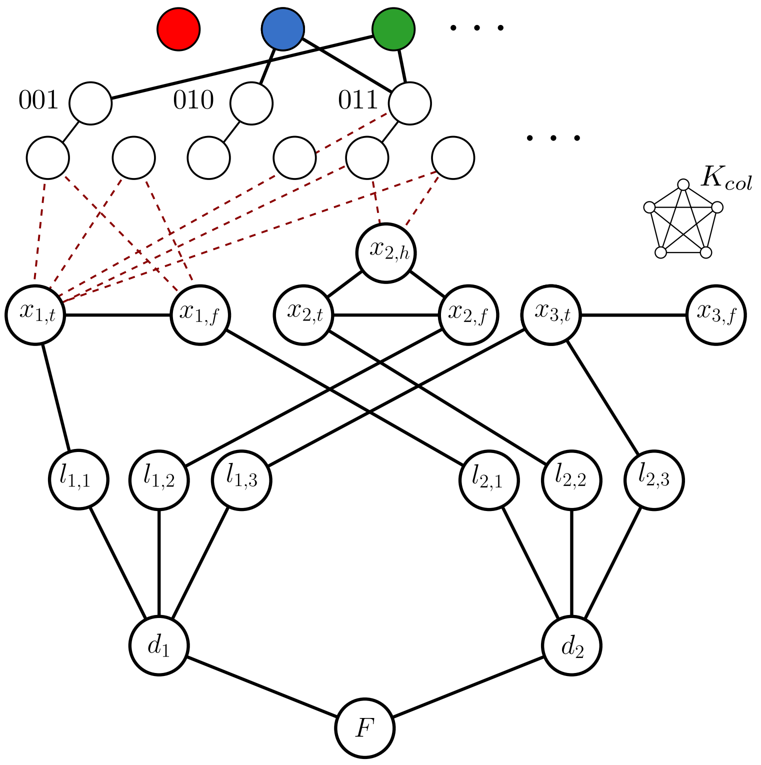

Figure 3: A visualisation of the construction with logarithmically many precolored vertices. Not all edges of the construction are present, e.g., dashed edges from Figure 1, most edges between nodes and G_1, and precolored vertices are connected only to p_3 of a displayed node, although they are connected either to all vertices in a node, or to none of them. The now uncolored clique K_{col} is connected the same way as it was in Section 2. Notice that $x_{1,t}$ and $x_{1,f}$ are identified by the first node (as they are connected only to p_1 and p_2 of the first node), but $x_{1,t}$ is not identified by the third node (as it is connected to all vertices of the third node).

We start with an observation about nodes.

Observation 3.1. *If* Drawer *reveals all the nodes A_i before it sends any vertex from G_1 (the gadgets and K_{col}),* Drawer *can force* Painter *to use two colors in the lower partite set of each node and these colors are different for each node.*

Proof. First we see that Painter is not able to distinguish incoming vertices from a node A by their edges to vertices in other nodes, since there is a complete bipartite graph between any two nodes.

Thus Drawer uses the following strategy for each node A independently: Drawer first presents the vertex p_1 from A; let c be its color. Then Drawer sends a vertex q that is one of vertices p_2 and p_3. Painter cannot deduce which of them is q, because both are not connected to p_1 (and both are connected to all vertices in other nodes). If Painter assigns the color c to q, then $q = p_3$ and Drawer sends p_2 which must get another color than c. Otherwise if q obtains another color than c, then $q = p_2$. □

Painter has $2N$ colors for the N nodes and k colors for G_1 (with the same names as in the previous section), thus overall Painter is allowed to use $k' = 2N + k = 2N + 2m + 2n_\forall + 3n_\exists$ colors 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 precolored vertices have a color that may be used later in G_1.

Lemma 3.2. *For a given fully quantified formula Q in the 3-DNF form that is not satisfiable,* Drawer *can force* Painter *to use $k' + 1$ colors.*

Proof. If at any point of the game there are $k' + 1$ colors in the graph, the lemma is proven. Drawer's strategy is to first present all nodes (in any order) and force to use two colors in the lower partite set of each node. Moreover, Painter has to use different colors in distinct nodes. Forcing such a coloring is possible by Observation 3.1.

Then Drawer sends K_{col} and the situation is similar to the one with the precolored K_{col}, since none of two colors used in the lower partite set of a node is allowed for a vertex in G_1, thus Painter must use the k colors in K_{col} for coloring the gadgets. Painter is able to recognize vertices in K_{col} by nodes, but nodes do not give Painter additional knowledge compared to K_{col}. In particular, Painter is not able to distinguish $\forall$-vertices $x_{i,t}, x_{i,f}$ for a universally quantified variable x_i when one of them arrives. We conclude the proof by applying Lemma 2.1. □

For a satisfiable formula nodes and precolored vertices become important. We give another useful observation about nodes.

Observation 3.3. *Suppose that* Painter *is using* FirstFit*, i.e.,* Painter *always assigns the smallest color not present among colored vertices adjacent to the incoming vertex. Then there are at most two colors used on each node.*

Moreover, if vertices p_1 and p_2 from a node A arrive before p_3 from A, then p_1 and p_2 have the same color.

Proof. Consider the last vertex q from a node A that is sent from A. We distinguish three cases:

- $q = p_1$: Let the color of p_2 be c and the color of p_3 be d (clearly $c \neq d$). Since the edges from p_1 and p_2 to other vertices in the construction except p_3 are exactly the same, Painter can use c for p_1, but it cannot use any color forbidden for p_2 when p_2 was colored with the possible exception of d. Hence p_1 obtains c or d, but in both cases the node A has two colors.
- $q = p_2$: Let the color of p_1 be c and the color of p_3 be d. If $c \neq d$, Painter uses c for p_2 for the same reason as in the previous case. Otherwise $c = d$ and Painter must color p_2 with another color, but there are two colors on A.
- $q = p_3$: We show that p_1 and p_2 have the same color. Without loss of generality, p_1 arrives first and obtains a color c. When p_2 arrives, it can be colored by c, but by no other color $c' < c$, since c' would be available for p_1 also when p_1 was colored, because the edges from p_1 and p_2 to other vertices except p_3 are the same. Hence p_1 and p_2 have the same color and the lemma follows.

□

PAINTER's strategy to win is basically the following: Color vertices in nodes greedily until a vertex u from G_1 arrives. At this time, some (but maybe not all) vertices in the gadgets can be recognized by their nodes. PAINTER uses the winning strategy from Lemma 2.2 on the gadgets for vertices that it can recognize, even if K_{col} has arrived only partially. For each vertex v that cannot be recognized, PAINTER is able to color (or already colored) the lower partite set of the v's node A with only one color, therefore it can use the same color for v and the vertex $p_3 \in A$, since they are not adjacent.

In the following proof, PAINTER waits for two nonadjacent vertices in G_1, even although one vertex from G_1 suffices. The reason is the we shall need such a condition in Section 4 when we remove precolored vertices.

Lemma 3.4. *For a given fully quantified formula Q in the 3-DNF form that is satisfiable, and for any order of sending vertices by* DRAWER*,* PAINTER *has a strategy that uses k' colors.*

Proof. Let $\mathcal{C}$ be the set of $2N$ colors for nodes and let $\mathcal{D}$ be the set of k colors for G_1 (the gadgets and K_{col}) including the color used on precolored vertices; both $\mathcal{C}$ and $\mathcal{D}$ are ordered arbitrarily and $\mathcal{C} \cap \mathcal{D} = \emptyset$.

PAINTER utilizes the precolored part P to decide whether an incoming vertex belongs to G_1 (recall that G_1 is the part of G_2 not adjacent to any precolored vertex), or to a node (and to which node). First, PAINTER uses the following algorithm:

```
Algorithm GREEDY: For an incoming vertex $u$ sent by DRAWER:
  1. If there are two nonadjacent vertices $u_1$ and $u_2$ in $G_1$ that arrived, stop the algorithm.
  2. If $u$ is from nodes, assign $u$ the smallest color from $\mathcal{C}$ not present among colored neighbors of $u$.
  3. Otherwise, if $u$ is from $G_1$, assign $u$ the smallest color from $\mathcal{D}$ not present among colored neighbors of $u$.
```

Note that the last u considered by GREEDY is not yet colored and u is from G_1. Observe that only a clique is colored in G_1 and this clique has at most k vertices.

Let u_1 and u_2 be the two nonadjacent vertices from the stopping condition of GREEDY. Observe that the nodes identifying u_1 and u_2 are different, since these nodes can be the same only for both $\forall$-vertices of one variable, but these vertices are adjacent.

PAINTER continues by the following algorithm. We remark that by "PAINTER can recognize $u \in G_1$" we mean that PAINTER knows which vertex in G_1 corresponds to u unless u is one of the two vertices for a universally quantified variable x_i – then PAINTER knows that one of $u = x_{i,t}$ and $u = x_{i,f}$ holds.

Algorithm WINNING**:** PAINTER creates a virtual graph G_{virt} with all colored vertices in G_1; the colors of such vertices are inherited from the graph G_2.

PAINTER shall simulate its winning strategy on G_{virt} using colors from $\mathcal{D}$. Since only a clique is colored in G_{virt}, it renames colors so that the colored vertices have colors according to the winning strategy on G_{virt}. PAINTER remembers the colors of vertices in the virtual graph G_{virt}.

For an incoming vertex u sent by DRAWER:

```
1. If $u$ is from $G_1$:
2.     If there is a vertex $v$ in nodes that is not adjacent to $u$, then $v$ is from one of at most two nodes identifying $u$ and PAINTER can recognize $u$. A virtual DRAWER sends $u$ to $G_{\text{virt}}$ and let $c \in \mathcal{D}$ be the color that it obtains by the winning strategy of PAINTER on $G_{\text{virt}}$ by Lemma 2.2. Color $u$ in $G_2$ using $c$ and call $u$ recognized.
3.     Otherwise, assign $u$ the smallest color from $\mathcal{C}$ not present among colored neighbors of $u$ and not used in $G_1$.
4. If $u$ is from a node $A$:
5.     If a vertex $v$ from $G_1$ identified by $A$ is colored by $c$, there is no edge between $v$ and $u$, $c \in \mathcal{C}$ and $c$ is not present among nodes, then color $u$ using $c$.
6.     Otherwise, assign $u$ the smallest color from $\mathcal{C}$ not present among colored neighbors of $u$.
```

If u is from G_1 and PAINTER can recognize it by a nonadjacent vertex v from nodes (line 2), then coloring it with the winning strategy is correct, since vertices that have a color from $\mathcal{D}$ in G_2 have the same color in G_{virt}.

Otherwise, if PAINTER cannot recognize u from G_1 (line 3), the vertex p_3 from each node A that serves for identifying u is not yet sent. It follows that there is at most one color used in the lower partite set of A by Observation 3.3. In this case, u gets some color c from $\mathcal{C}$ that is not used in G_1. We want to show that c will be used on the vertex p_3 from a node that identifies u.

To see this, observe that when the vertex p_3 from a node A that identifies u arrives, the condition on line 5 is satisfied and p_3 obtains c unless the color c is already used in nodes. Thus if the unrecognized vertex u from G_1 is not a $\forall$-vertex, it has the same color as the vertex p_3 from the single node A identifying v, since c can be used in nodes only for p_3 of the node A.

Otherwise, if u is an unrecognized $\forall$-vertex, then u is identified by two nodes A and A'. Both vertices p_3 from A or A' arrive after u and one of them obtains c, since the algorithm prefers to use colors of v and possibly of the other $\forall$-vertex for the same variable (if its color is in $\mathcal{C}$).

Recognized vertices from G_1 are colored by the wining strategy on G_{virt} using k colors from $\mathcal{D}$. As we have shown above, each unrecognized vertex v has the same color as a vertex p_3 in one of the nodes identifying v. There are at most $2N$ colors used in nodes by Observation 3.3 and all of them are from $\mathcal{C}$. Thus PAINTER uses at most $k' = 2N + k$ colors. □

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 as well and easier to see. We prove the following lemma which holds for all graphs with precolored vertices satisfying an assumption.

Lemma 4.1. *Let G be a graph with precolored subgraph G_p 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 such that if $\chi^O(G) \le k$, then there exists a winning strategy of Painter *where* Painter *colors 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 $|V(G')| \le 25|V(G)|$,*
- *G' can be constructed from G in polynomial time with respect to the size of G,*
- *it holds that $\chi^O(G') \le k'$ if and only if $\chi^O(G) \le k$.*

Note that we do not assume anything about D or E except that E may be colored by Painter using FirstFit before two nonadjacent vertices from D arrive. We give a proof of Theorem 1.1 using Lemma 4.1 in Section 4.1.

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 any 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 Figure 4.

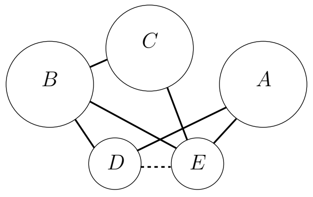

Figure 4: An illustration of our construction of G' where parts A, B and C form a supernode and parts D and E form the original graph G (the remaining precolored vertices are not shown). A thick line connecting two parts of G' corresponds to a complete bipartite graph between them. The edges between D and E remain unchanged from the original graph G. If there is no line between two parts, then there is no edge between them in G'.

Proof of Lemma 4.1. 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 $\chi^O(G') \le k'$ for some k' if and only if $\chi^O(G) \le k$. We set k' to $k + 2S$, since at most $2S$ colors will be used in the supernode.

We start with the "only if" direction which is easier. Suppose that $\chi^O(G) > k$ and consider G' constructed from G. Drawer starts by sending A and then $B \cup C$ such that no color is both in A and in B; this is possible, since cliques A and C have the same size. More precisely, if Painter would assign a color from A to a vertex from $B \cup C$, Drawer decides that the vertex is in C; otherwise the vertex is in B (or in C if every vertex in B has been colored). This forces Painter to use $2S$ different colors on A and B and no such color can be used for D and E.

Now the colored part of G' (precolored vertices and the supernode) may be viewed as a precolored part in G except that Painter may not be able to use the supernode in the same way as a precolored vertex. Since this only helps Drawer, Drawer sends vertices according to its winning strategy for G. This proves the "only if" direction of the third proposition of Lemma 4.1.

In the rest of this section we focus on the opposite direction: assuming that $\chi^O(G) \leq k$, 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 with k colors.

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 which color u receives (since it is universal to colored vertices in D and E).

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

Definition 4. 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 stops when it detects two useful vertices d_1 and d_2:

```
Algorithm WaitForD: For an incoming vertex $u$ sent by Drawer:
  1. Let $G'_{\mathrm{R}}$ be the revealed part of $G'$ (i.e., colored vertices and $u$, but not precolored vertices)
  2. Find a maximum clique in $G'_{\mathrm{R}}$ and denote it as $K_1$.
  3. Find a maximum clique in $G'_{\mathrm{R}}$ from those which are 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 $G'_{\mathrm{R}}$ 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 Section 1 that Online Chromatic Number lies in PSPACE.

Let v be the incoming vertex u when WaitFor$D$ 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 in Step 4 of WaitFor$D$. 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 d_1 nor d_2 is from E, since any vertex of E is practically universal to both cliques.
- Vertices d_1 and d_2 cannot both be from $B \cup C$, nor can both be from A, as they would be adjacent.
- If d_1 is in $B \cup C$ and d_2 in A (or vice versa), 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 (or vice versa), then d_1 and d_2 are adjacent.
- Finally, if $d_1 \in D$ and $d_2 \in C$ (or vice versa), 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 an already 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, the player 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 WaitFor$D$ 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. To start, we give a few extremal examples of which parts of the graph can be colored when WaitFor$D$ terminates:

1. The whole C and the whole A are colored, but only a clique from D is colored.
2. Some pairs of nonadjacent vertices in D are colored (or even the whole D) and the whole A is colored, but only $S/2$ vertices from $B \cup C$ arrived.
3. There are again some pairs of nonadjacent colored vertices in D and now the whole C is colored, but only $S/2$ vertices from A arrived.

Moreover, in all cases, some part of B and some part of E may also be colored.

Continuing with the intuition, as Painter can now recognize the parts of the construction (with an exception of K_D), we would like to 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), since 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 completely and have the same colors, thus Painter cannot ignore colors used 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 which colors vertices in K_D receive.

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.

After the algorithm WaitFor$D$ finishes, the player Painter applies an algorithm which simulates its winning strategy on G and maintains disjoint sets of colors $\mathcal{C}$ (mainly for the supernode) and $\mathcal{E}$ (for E and vertices surely in D). Recall that Painter is not able to distinguish between A and K_D, thus vertices in K_D are treated in the same way as vertices in A. More precisely, we define three color sets $\mathcal{C}$, $\mathcal{S}$ and $\mathcal{E}$ as follows:

Definition 5.

- If colored vertices surely in D form a clique, let $\mathcal{S}$ be the set of colors that are used on a colored vertex surely in D and that are not used in $\tilde{C}$. Otherwise, if there are two nonadjacent colored vertices surely in D, let $\mathcal{S}$ be an empty set.
- Let $\mathcal{C}$ be the set of colors present currently in the supernode or in K_D and among vertices surely in D (i.e., in $\tilde{A}, \tilde{B}, \tilde{C}$, or $\tilde{D}$) except colors from $\mathcal{S}$ and colors present also in $\tilde{E}$.
- Let $\mathcal{E}$ be the set of colors assigned to vertices in $\tilde{E}$ and colors in $\mathcal{S}$.

We will update the color sets when we apply our algorithms and as more vertices arrive on input; the precise updating procedure is specified throughout the algorithms. We shall keep that $\mathcal{C}$ and $\mathcal{E}$ are disjoint. The winning algorithm for Painter is split into two parts, initialization and coloring:

```
Algorithm InitSimulation:
  1. Painter initializes a virtual graph $G_{\text{virt}}$ by copying all vertices from $\tilde{E}$; the colors of such
     vertices are inherited from $G'$.
  2. Next, update the virtual graph with vertices surely in $D$ as follows:
  3. Set an arbitrary (virtual) ordering on already arrived vertices surely in $D$.
  4. For every vertex $u$ in the ordering with a color $c$ in $G'$:
  5.     If $c \in \mathcal{S}$, color $u$ in $G_{\text{virt}}$ using $c$.
  6.     Otherwise, color $u$ in $G_{\text{virt}}$ using a color $c'$ from $\mathcal{E}$ or a new color $c'$ not used in $G'$ according
         to the winning strategy. Add $c'$ to $\mathcal{E}$ if it is not there.
```

When adding a colored vertex u that is surely in D to the virtual graph G_{virt}, if $c \in \mathcal{S}$, coloring u with c in G_{virt} does not harm the winning strategy by the assumptions of Lemma 4.1 — if $\mathcal{S}$ is nonempty, colored vertices surely in D form a clique and Painter may use any proper coloring for them in the winning strategy. Moreover, some colors may be both in $\tilde{D}$ and in $\tilde{E}$, but the winning strategy still works by the assumptions.

Otherwise, if $c \notin \mathcal{S}$, u may obtain a different color in G_{virt} than in the actual G'.

```
Algorithm ColorBySimulation: For an incoming vertex $u$ sent by Drawer:
  1. For each vertex $w$ colored with $c$ in $G'$ that is surely in $D$ now, but that was indistinguishable
     from vertices in $A$ before $u$ arrived (i.e., in $K_D$ before $u$ arrived):
     (a) Add $w$ to the virtual graph $G_{\text{virt}}$.
     (b) As all colored vertices in $G_{\text{virt}}$ are connected to $w$ (otherwise it would not be in $K_D$), $w$
         would obtain a new color if we add $w$ to $G_{\text{virt}}$ and use the winning strategy of Painter
         on $G_{\text{virt}}$.
     (c) If $c$ is not used in $C$, remove $c$ from $\mathcal{C}$, add $c$ to $\mathcal{E}$ and color $w$ using $c$ in $G_{\text{virt}}$.
     (d) Otherwise, if $c$ is present in $C$, color $w$ in $G_{\text{virt}}$ using a new color $c'$ not contained in
         $\mathcal{C}$ and $\mathcal{E}$ and add $c'$ to $\mathcal{E}$. (In this case, $w$ has a different color in $G'$ and $G_{\text{virt}}$, but
         Painter pretends that its color is $c'$ for the purpose of simulation and does not use $c$
         for vertices in $D \cup E$ in $G'$.)
  2. If $u$ is from $C$:
     (a) If there is a color $c$ among vertices surely in $D$ such that $c \in \mathcal{C}$ and $c$ is not present
         among colored vertices adjacent to $u$, color $u$ using the smallest such color.
     (b) Otherwise, if there is a color in $\mathcal{C}$ not present among colored vertices adjacent to $u$, color
         $u$ using the smallest such color.
     (c) Otherwise, choose a new color $c$ not contained in $\mathcal{C}$ and $\mathcal{E}$, color $u$ with $c$, add $c$ to $\mathcal{C}$.
  3. If $u$ is from $A$ or $B$ or from $K_D$, i.e., indistinguishable from $A$:
     (a) If there is a color in $\mathcal{C}$ not present among colored vertices adjacent to $u$, color $u$ using
         the smallest such color.
     (b) Otherwise, choose a new color $c$ not contained in $\mathcal{C}$ and $\mathcal{E}$, color $u$ with $c$, add $c$ to $\mathcal{C}$.
  4. If $u$ is from $E$ or surely from $D$:
     (a) The virtual Drawer sends $u$ to $G_{\text{virt}}$. Simulate the winning strategy on $G_{\text{virt}}$ and color
         $u$ using a color $c \in \mathcal{E}$ or a new color $c$ that is not in $\mathcal{E}$ or $\mathcal{C}$; in the latter case, add $c$ to
         $\mathcal{E}$.
     (b) Color $u$ in $G'$ using the same color $c$.
```

We remark that the first vertex u colored by the algorithm ColorBySimulation is v, the vertex on which WaitFor$D$ stops. In a sense, if $\tilde{D}$ (the part of D colored by WaitFor$D$) is not a clique, Painter pretends that in $\tilde{D}$ there are colors from the simulation, not the colors assigned by WaitFor$D$. Otherwise, if $\tilde{D}$ is a clique, Painter uses colors from G' in G_{virt} which may result in renaming the colors in the winning strategy on G_{virt}.

Observe that any color used in G' is in $\mathcal{C}$ or $\mathcal{E}$ and that these sets of colors are disjoint. Thus the colors used for $D \cup E$ are different from those in the supernode except colors used in $\tilde{D}$ or colors of vertices that were in K_D when they arrived (which may happen also during the execution of ColorBySimulation).

Note that if u is from E or surely from D, coloring with c is sound; this follows from the winning strategy on G and from the fact that colors in $\mathcal{E}$ are not used in the supernode (recall that colors in $\mathcal{S}$ are not used in $\tilde{C}$).

The key part of the analysis is captured by the following claim and its proof which explains the design of ColorBySimulation.

Claim 4.2. *For any color $c \in \mathcal{C}$ it holds that c is used on a vertex in the supernode.*

Proof. Any color added to $\mathcal{C}$ by ColorBySimulation and not removed from $\mathcal{C}$ in Step 1.(c) is used in the supernode and no color used in E is in $\mathcal{C}$, thus it remains to show that any color $c \in \mathcal{C}$ used in $\tilde{D}$ (either on a vertex surely from D or on a vertex in K_D when ColorBySimulation starts) is present in C when the whole G' is colored.

We assume that c is not used in G_{virt} and thus also not in $\tilde{E}$, otherwise c is not in $\mathcal{C}$. Colors in $\tilde{D}$ and not in G_{virt} (thus also not in $\mathcal{S}$) are in $\mathcal{C}$, thus ColorBySimulation can use c only for a vertex in C. Let u be the first vertex in $\tilde{D}$ that obtains color c and let t be the time of coloring u.

Note that if u is in K_D at any time, then we are done, since u becomes surely in D at some time (e.g., when the first vertex from B arrives) and at that time, either c is already used in C, or ColorBySimulation uses c also in G_{virt} and removes it from $\mathcal{C}$. Also, if colored vertices surely from D form a clique when ColorBySimulation starts and u is surely in D at that time, then c is either used in $\tilde{C}$, or it is in $\mathcal{S}$ and thus not in $\mathcal{C}$.

Therefore we assume that u is not in K_D at any time, thus u is surely in D, and there are two nonadjacent colored vertices surely in D when ColorBySimulation starts. This implies $|K_{BC}| = |S|/2$ or $|K_A| = |S|/2$ when ColorBySimulation starts which means that $|\tilde{C}| \leq |S|/2$ or $|\tilde{A}| \leq |S|/2$.

If $|\tilde{C}| \leq |S|/2 < |S| - N$, then at least N vertices from C are colored by ColorBySimulation. In this case, ColorBySimulation must assign c to a vertex in C at some point, since it prefers colors used in $\tilde{D}$ before colors in A or new colors and $N \geq |\tilde{D}|$.

Otherwise $|\tilde{C}| > |S|/2$, thus $|\tilde{A}| \leq |S|/2$. If at time t there is some color in C not used in A, let c' be the smallest such color in the ordering of colors. We observe that WaitFor$D$ uses the color c' for u, since other colors allowed for coloring u are some colors used in E at time t (which we assume that u does not get), or colors in D (which u also does not get as it is the first vertex in D that has color c), or a new color not yet used anywhere, but any new color is after c' in the ordering of colors. Hence the greedy algorithm WaitFor$D$ prefers c' and $c = c'$, thus c is already used in C.

Otherwise, when Painter colors u, all colors used in C are also present in A which also means that there are at least as many colored vertices in A as in C at time t. Let r be the number of colored vertices in C at time t. WaitFor$D$ colors more than $|S| - N - r$ vertices from C after time t, at most $|S|/2 - r$ of them are colored by a color already used in A, thus after time t WaitFor$D$ colors more than $|S| - N - r - (|S|/2 - r) = |S|/2 - N$ vertices from C which do not get a color from A. Hence WaitFor$D$ must use c for a vertex in C as $|S|/2 - N \geq N \geq |\tilde{D}|$. This concludes the proof of the claim. □

There are always $2S$ colors used in the supernode, since otherwise there must be a color c in A that is not in $B \cup C$, but both WaitFor$D$ and ColorBySimulation prefer coloring B using colors in A and coloring A using colors from $B \cup C$. Thus $|\mathcal{C}| = 2S$ by Claim 4.2.

Since all colors in $\mathcal{E}$ are used in the virtual graph G_{virt} (this includes also colors in $\mathcal{S}$) and PAINTER uses the winning strategy on G_{virt}, we know that $|\mathcal{E}| \leq k$. Overall, PAINTER uses at most $2S + k = k'$ colors and we proved $\chi^O(G') \leq k'$ if $\chi^O(G) \leq k$. This concludes the proof of Lemma 4.1. □

4.1 Proof of the main theorem

We show how to apply Lemma 4.1 on the construction from Section 3 and prove the PSPACE-completeness of ONLINE CHROMATIC NUMBER.

Proof of Theorem 1.1. Let ϕ be a formula of size n. We first construct a graph G_2 with $p = \mathcal{O}(\log n)$ precolored vertices $z_1, z_2, \ldots z_p$ as described in Sections 2 and 3. By Lemmas 3.4 and 3.2 we have $\chi^O(G_2) \leq k'$ if and only if ϕ is satisfiable. Then for each precolored vertex in G_2, we apply Lemma 4.1 iteratively until we obtain a graph G_3 with no precolored vertex such that $\chi^O(G_3) \leq k''$ iff ϕ is satisfiable (for some k'').

The number of vertices in G_3 is polynomial in n, because G_2 has linearly many vertices and the number of vertices is multiplied by a constant with each of $\mathcal{O}(\log n)$ applications of Lemma 4.1. The constructions in Sections 2 and 3 and in Lemma 4.1 yield a polynomial-time algorithm for computing G' from ϕ.

It remains to check the assumption of Lemma 4.1 in each iteration i; we recall it here: If $\chi^O(G) \leq k$, then there exists a winning strategy of PAINTER where PAINTER colors 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.

Let H_0 be G_2 and for each iteration i let H_{i-1} be the graph before removing the i-th precolored vertex z_i and let H_i be the graph created from H_{i-1} by the construction in Lemma 4.1. Let the nonprecolored vertices of H_{i-1} be partitioned into two disjoint induced subgraphs D_i and E_i such that all vertices from E_i are connected to z_i and no vertex in D_i is connected to z_i. We denote by A_i, B_i and C_i the cliques in the supernode in H_i. Thus H_i consists of parts A_i, B_i, C_i, D_i, and E_i (and possibly some precolored vertices which we do not take into account).

Note that E_i contains only some nodes and D_i contains the whole G_1 (i.e., K_{col} and the gadgets for variables and clauses) together with supernodes from previous iterations and nodes that are not in E_i. We also remark that for $i > 1$ parts A_{i-1}, B_{i-1}, and C_{i-1} of H_{i-1} are in D_i and that there may be some nodes both in D_i and in E_{i-1} and also some nodes both in E_i and in D_{i-1}.

We start with the first iteration. The first part of the assumption holds for $H_0 = G_2$, since the algorithm GREEDY from the winning strategy from Lemma 3.4 colors the nodes by FIRSTFIT before two nonadjacent vertices from G_1 arrive and D_1 contains G_1.

For the "moreover" part in the first iteration, if FIRSTFIT assigns the same color c to a vertex d in D_1 and a vertex e in E_1 before two nonadjacent vertices from D_1 arrive, then e must be from a node that identifies d and in this case PAINTER can just pretend that d is an unrecognized vertex. In other words, the color c is intended for nodes and not used in G_1.

For an iteration $i > 1$, note that in the winning strategy on H_{i-1} PAINTER colors H_{i-1} using FIRSTFIT before two large and practically disjoint cliques from the supernode arrive, one mostly from A_{i-1} and the other mostly from $B_{i-1} \cup C_{i-1}$. Since there are many pairs of nonadjacent vertices between these cliques and these cliques are contained in D_i, PAINTER can color E_i using FIRSTFIT before two nonadjacent vertices from D_i arrive.

For the "moreover" part, if FirstFit assigns the same color c to a vertex d in D_i and a vertex e in E_i before two nonadjacent vertices from D_i arrive, then e is from nodes (as E_i contains only nodes) and d is either from G_1 as in the first iteration, or from the supernode created in the previous iteration; in the latter case, e is in D_j and d is in C_j for $j < i$, since otherwise they would be connected. (Note that d cannot be from nodes, since there is a complete bipartite graph between every two nodes and a node is never split between E and D.) In both cases, this does not harm the winning strategy of Painter on H_{i-1}, because the algorithm ColorBySimulation assumes that FirstFit may use the same colors for more vertices in $D_{i-1} \cup E_{i-1}$ or in $D_{i-1} \cup C_{i-1}$. This concludes that the assumption of Lemma 4.1 is satisfied in each iteration. □

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

References

[1] D. R. Bean. *Effective coloration.* The Journal of Symbolic Logic, Vol. 41, No. 2, pp. 469-480 (1976).

[2] H. Bodlaender. *On the complexity of some coloring games.* International Journal of Foundations of Computer Science 2.02, pp. 133-147 (1991).

[3] M. Böhm, P. Veselý. *Online Chromatic Number is PSPACE-Complete.* In proceedings of *27th International Workshop on Combinatorial Algorithms* (IWOCA 2016). LNCS 9843, 16–28 (2016).

[4] A. Csernenszky, R. R. Martin, A. Pluhár. *On the complexity of Chooser-Picker positional games.* ArXiv preprint arXiv:1605.05430 (2016).

[5] P. Dvořák, T. Valla. *On the Computational Complexity and Strategies of Online Ramsey Theory.* In proceedings of *8th European Conference on Combinatorics, Graph Theory and Applications* (EuroComb 2015). Electronic Notes in Discrete Mathematics 49, 729–736 (2015).

[6] A. Gyárfás, J. Lehel. *First fit and on-line chromatic number of families of graphs.* Ars Combinatoria 29C, 168–176 (1990).

[7] A. Gyárfás, Z. Kiraly, J. Lehel. *On-line graph coloring and finite basis problems.* Combinatorics: Paul Erdos is Eighty Volume 1., 207–214 (1993).

[8] M. M. Halldórsson. *Parallel and on-line graph coloring.* J. Algorithms, 23, 265–280 (1997).

[9] M. M. Halldórsson. *Online Coloring Known Graphs.* The Electronic Journal of Combinatorics, 7(1), R7 (2000).

[10] M. M. Halldórsson, M. Szegedy. *Lower bounds for on-line graph coloring.* Theoretical Computer Science 130.1: 163-174 (1994).

[11] H. Kierstad. *On-line coloring k-colorable graphs.* Israel Journal of Mathematics 105, 93-104 (1998).

[12] C. Kudahl. *On-line Graph Coloring.* Master's thesis, University of Southern Denmark (2013).

[13] C. Kudahl. *Deciding the On-line Chromatic Number of a Graph with Pre-Coloring is PSPACE-Complete.* In proceedings of *9th International Conference on Algorithms and Complexity* (CIAC 2015). LNCS 9079, 313–324 (2015). Also arXiv:1406.1623.

[14] L. Lovász, M. Saks, W. T. Trotter. *An on-line graph coloring algorithm with sublinear performance ratio.* Annals of Discrete Mathematics 43: 319-325 (1989).