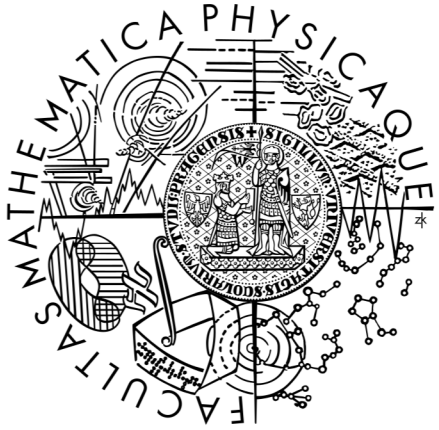
Charles University in Prague
Faculty of Mathematics and Physics

MASTER THESIS



Pavel Dvořák

Online Ramsey Theory

Computer Science Institute of Charles University

Supervisor of the master thesis: RNDr. Tomáš Valla, Ph.D.
Study programme: Computer Science
Study branch: Discrete Models and Algorithms

Prague 2015

I declare that I carried out this master thesis independently, and only with the cited sources, literature and other professional sources.

I understand that my work relates to the rights and obligations under the Act No. 121/2000 Sb., the Copyright Act, as amended, in particular the fact that the Charles University in Prague has the right to conclude a license agreement on the use of this work as a school work pursuant to Section 60 subsection 1 of the Copyright Act.

In Prague 29.4.2015 Pavel Dvořák

Title: Online Ramsey Theory

Author: Pavel Dvořák

Institute: Computer Science Institute of Charles University

Supervisor: RNDr. Tomáš Valla, Ph.D., Computer Science Institute of Charles University

Abstract: An online Ramsey game is a game between Builder and Painter, alternating in turns. In each round Builder draws an edge and Painter colors it either red or blue. Builder wins if after some round there is a monochromatic copy of the fixed graph H, otherwise Painter is the winner. In this thesis we investigate the computational complexity of the related decision problem and we show that it is PSPACE-complete. Moreover, we study a version of the game when Builder can draw only planar graphs and a generalization of the game for hypergraphs. We found a new class of graphs unavoidable on planar graphs. We provide a result showing that Builder wins the online Ramsey game on 3-uniform hyperforests when the goal graph H is 1-degenerate.

Keywords: online Ramsey game complexity strategies

I would like to thank my supervisor Tomáš Valla for introducing me to combinatorial game theory, for helping me with this thesis and for fruitful discussion about online Ramsey theory. I would also like to thank Martin Böhm, who came up with the idea to study the complexity of the online Ramsey game, and Karel Král for checking the part of this thesis, despite his lack of time. Last but not least, I would like to thank my family for supporting me during my studies.

Contents

1. Introduction

The online Ramsey game is a two-player game with perfect information. It is played on the *background graph* $G = (V, E)$. At the beginning of the game the edges in E are either uncolored or some of them are colored. In each round the first player, usually addressed as *Builder*, draw an uncolored edge $e \in E$. The second player, usually addressed as *Painter*, colors the recently drawn edge e by red or blue. The goal of Builder is to force Painter to color a monochromatic copy of the fixed *goal graph* H. If Builder does so he wins the game. Otherwise, if all edges in E are colored and the graph G does not contain a monochromatic copy of H as a subgraph then Painter wins. In the case the graph G is infinite Painter wins if Builder can not win in finite time.

The online Ramsey game was introduced independently by Beck [2] and Friedgut et al. [6]. Since then several results have been published. In Section 1.2 we present some known results about this topic and point out differences between online and offline Ramsey theory. We introduce in Section 1.1 some basic notions which we use in this thesis. The summary of our results is presented in Section 1.3.

1.1 Preliminaries and Notions

In this section we presented some notions of the graph theory, logic and complexity. It should serve as a brief and a little bit informal introduction to these fields. We try to explain here only some basic notions. If the reader is familiar with the basics of these fields this section can be certainly skipped, except Subsection 1.1.4 where we present some our notions and some notions which are not so widely known.

1.1.1 Graph Theory

A *graph* G is a pair (V, E) where V is a set of vertices and $E \subseteq \binom{V}{2}$ is a set of edges, thus E contains pairs of the vertices. A *degree of a vertex* $v \in V$ is a number of edges in E which are incident with v. Thus, the degree of v, $\deg(v) = |\{e | e \in E, v \in e\}|$. The set of vertices (or edges) of G is also denoted as $V(G)$ (or $E(G)$). We say two vertices $u, v \in V(G)$ are *neighbors* if $\{u, v\} \in E(G)$. A *size* of a graph $G = (V, E)$ is the size of V and E. A *path* P_n is a graph (V, E) such that

$$V = \{v_0, \ldots, v_n\},$$
$$E = \big\{\{v_{i-1}, v_i\} | i \in [n]\big\}.$$

By $[n]$ we mean the set $\{1, \ldots, n\}$. Thus, the path P_n is a path with n edges and $n + 1$ vertices. By *length* of a path P we always mean the number of edges of P. A *cycle* C_n is a graph (V, E) such that

$$V = \{v_0, \ldots, v_n\},$$
$$E = \big\{\{v_{i-1}, v_i\} | i \in [n]\big\} \cup \big\{\{v_0, v_n\}\big\}.$$

Thus, the cycle C_n arises from the path P_n by connecting the endpoints of P_n. A *complete graph* $K_n = (V, E)$ on n vertices contains all possible edges, i.e. $E = \binom{V}{2}$. A *subgraph* $H = (U, F)$ of $G = (V, E)$ is a graph such that $U \subseteq V$ and $F \subseteq E$. A graph G is *k-degenerate* if every subgraph of G contains a vertex of degree at most k. A subgraph of a graph $G = (V, E)$ *induced* by a vertex set $U \subseteq V$ is a graph $H = (U, F)$ such that F contains all edges whose endpoints are in U, formally

$$F = \big\{\{u, v\} | \{u, v\} \in E, u, v \in U\big\}.$$

A graph $G = (V, E)$ is *connected* if for all $u, v \in V$ it holds that there is a path between u and v as a subgraph of G. A *connected component* (or just *component*) of a graph $G = (V, E)$ is a maximal connected subgraph $H = (U, F)$. Thus, H is connected and there is no graph H_1 such that H is a subgraph of H_1 and H_1 is connected. A *cut vertex* is a vertex in a component C such that if we remove it the component C split to more components.

A *forest* is a graph which does not contains any cycle as a subgraph. A *tree* is a connected forest. An important property of the trees (or forests) is that every tree T contains a vertex $v \in V(T)$ (usually called *leaf*) such that the degree of v is 1. Therefore, we can remove leaf v (with the incident edge) from a tree T and get another tree. This is a common method of proving some properties for trees by induction. A *rooted tree* T is a tree T with labeled vertex $v \in V(T)$—called the root.

A *planar* graph is a graph that can be drawn on the plane in such a way that its edges intersect only at their endpoints. Such drawing of the planar graph is called *embedding*. A *face* in some embedding of the planar graph is a part of the plane bounded by drawn edges. An *outer face* is the only unbounded part of the plane when a planar graph is embedded. An *outer planar* graph G is a planar graph such that there exists an embedding for G where all vertices in $V(G)$ are incident with the outer face.

A *two-terminal* graph is a graph G with two labeled vertices $u, v \in V(G)$. A *series operation* creates from two two-terminal graphs $T_1 = (G_1, u_1, v_1)$ and $T_2 = (G_2, u_2, v_2)$ a two-terminal graph $S(T_1, T_2) = (G, u, v)$ such that it arises from edge disjoint union of G_1 and G_2 by identifying the vertices v_1 and u_2 and $u = u_1$ and $v = v_2$. A *parallel operation* create from two two-terminal graphs $T_1 = (G_1, u_1, v_1)$ and $T_2 = (G_2, u_2, v_2)$ a two-terminal graph $P(T_1, T_2) = (G, u, v)$ such that it arises from edge disjoint union of G_1 and G_2 by identifying the vertices u_1 and u_2 into u and v_1 and v_2 into v. A *series-parallel* graph (G, u, v) is a two-terminal graph which arises by using inductively the series and parallel operation:

1. A 3-tuple (P_1, u, v) is a series-parallel graph.

2. If $T_1 = (G_1, u_1, v_1)$ and $T_2 = (G_2, u_2, v_2)$ are series-parallel graphs then $S(T_1, T_2)$ is a series-parallel graph.

3. If $T_1 = (G_1, u_1, v_1)$ and $T_2 = (G_2, u_2, v_2)$ are series-parallel graphs then $P(T_1, T_2)$ is a series-parallel graph.

An *independent set* of a graph G is a set $I \subseteq V(G)$ such that I induces a graph with no edges. A *clique* of a graph G is a set $C \subseteq V(G)$ such that C induces a

complete graph. A *vertex cover* of a graph G is a set $C \subseteq V(G)$ such that for every edge $\{u, v\} \in E(T)$ it holds that $u \in C$ or $v \in C$. A *bipartite graph* is a graph $G = (V, E) = (A \cup B, E)$ such that V can be partitioned $V = A \dot\cup B$ in such a way that sets A and B induce independent sets[1]. A *complete bipartite* graph $G = (A \cup B, E)$ is a bipartite graph which contains all possible edges, formally

$$E = \big\{\{u, v\} | u \in A, v \in B\big\}.$$

Two graphs G_1, G_2 are *isomorphic* via bijection $f : V(G_1) \to V(G_2)$ if for all $\{u, v\} \in \binom{V(G_1)}{2}$ it holds that $\{u, v\} \in E(G_1)$ if and only if $\{f(u), f(v)\} \in E(G_2)$.

A *hypergraph* H is a pair (V, F) where V is a set of vertices and $F \subseteq \mathcal{P}(V)$, where $\mathcal{P}(V)$ denote the set of all subsets of V. The elements of F are called hyperedges. A hypergraph $H = (V, F)$ is *k-uniform* if every hyperedge $f \in F$ has the size k, thus

$$F \subseteq \binom{V}{k}.$$

All meaningful notions like subhypergraph, degree and degeneracy of hypergraphs are defined by the same way as for the graphs.

1.1.2 Logic

A *formula* contains variables, negations and binary connective and it is defined inductively:

1. A variable x is a formula.
2. If φ is a formula then a negation $\neg\varphi$ is a formula.
3. If φ and ψ are formulas and $\cdot$ is a binary connective then $\varphi \cdot \psi$ are formulas.

A *literal* in a formula φ is a variable or its negation. A formula φ in *conjunctive normal form* (or CNF) contains variables, their negations and binary connectives $\wedge$ for *conjunction* and $\vee$ for *disjunction*. It is defined inductively:

1. A literal ℓ is a formula in CNF.
2. If $\ell_1, \ldots, \ell_n$ are literals then a *clause* $\ell_1 \vee \ell_2 \vee \cdots \vee \ell_n$ is a formula in CNF.
3. If $C_1, \ldots, C_m$ are clauses then a conjunction of the clauses $C_1 \wedge C_2 \wedge \cdots \wedge C_m$ is a formula in CNF.

Let φ be a formula in CNF and X be a set of variables of φ. An *evaluation* of φ is a function $e : X \to \{0, 1\}$ (or {false, true}). We denote the value of a formula φ in CNF with an evaluation e as $v(\varphi, e)$. It is computed inductively:

1. If φ is a literal x then $v(\varphi, e)$ is true if and only if $e(x) = 1$.
2. If φ is a literal $\neg x$ then $v(\varphi, e)$ is true if and only if $e(x) = 0$.
3. If φ is a clause $\ell_1 \vee \cdots \vee \ell_n$ then $v(\varphi, e)$ is true if and only if there exists a literal ℓ_i such that $v(\ell_i, e)$ is true.

[1]The partition $V = A \dot\cup B$ means that $A \cap B = \emptyset$ and $A \cup B = V$.

4. If φ is a conjunction of clauses $C_1 \wedge C_2 \wedge \cdots \wedge C_m$ then $v(\varphi, e)$ is true if and only if for every $i \in [m]$ it holds that $v(C_i, e)$ is true.

A formula φ is *satisfiable* if there exists an evaluation e such that a value of φ with the evaluation e is true. We mean by $e[x = c]$ an evaluation which evaluates the variable x by the value c.

A *quantified* formula φ in prenex conjunctive normal form is a formula which has two parts, the first one contains only quantifiers $\forall x$ or $\exists x$ for the variables of φ and the second part is a formula ψ in CNF. A value $v(\varphi, e)$ is computed inductively:

1. If φ does not contain any quantifier then $v(\varphi, e)$ is computed by the same way as in the previous case.

2. If φ has a form $\forall x \varphi'$ then φ is true if and only if $v(\varphi', e[x = 0])$ is true and $v(\varphi', e[x = 1])$ is true.

3. If φ has a form $\exists x \varphi'$ then φ is true if and only if $v(\varphi', e[x = 0])$ is true or $v(\varphi', e[x = 1])$ is true.

1.1.3 Computational Complexity

An *alphabet* Σ is a finite set of characters. A *word* over Σ is a finite sequence of the characters in Σ. A set of all words is usually denoted by Σ^*. A *decision problem* (or a language) is a subset of Σ^*.

A *deterministic Turing machine* (or DTM) is an abstract machine defined by a tuple $(Q, \Sigma, \varepsilon, \delta, q_0, F)$ and it consists of:

1. An eventually infinite *tape* divided into cells. Each cell contains one symbol from the alphabet Σ. Character $\varepsilon \in \Sigma$ is the blank symbol which is the only character in Σ allowed to be on the tape infinitely many times.

2. A *head* which can read and write characters on the tape and can move on the tape.

3. A *state register* that stores an actual state. The machine can be in any state in Q. The state $q_0 \in Q$ is the initial state. The subset $F \subseteq Q$ contains accepting states.

4. A *program* which is defined by the partial function

$$\delta : Q \times \Sigma \to Q \times \Sigma \times \{L, N, R\},$$

which is called the *transition function*.

Let $M = (Q, \Sigma, \varepsilon, \delta, q_0, F)$ be a DTM. A *configuration* of M consists of a state $q \in Q$ of M, a content of the tape of M and a position of the head of M. An *initial configuration* of M consists of the initial state q_0, an input word x over Σ written on the tape and an initial position of the head of M (usually at the beginning of x). Other cells of the tape (besides those where x is written) contains the blank symbol ε. A *step* of M is one use of the transition function. Let $c \in \Sigma$ be a character written in the cell under the head of M and q be a state of M. Then, M executes a step $\delta(q, c) = (q_1, c_1, m)$ (if defined) by:

1. Changing the actual state to q_1.

2. Writing the character c_1 in the cell under the head.

3. Moving the head according to m—one cell left if $m = L$, one cell right if $m = R$, do not move if $m = N$.

A *computation* of M is a sequence of the steps of M starting in the initial state of M. The DTM M stops if the transition function is not defined for actual state and the character under the head. An *accepting computation* of M is a computation which stops in some accepting state in F—then we say M accepts an input word x. A *refusing computation* of M is a computation which stops in a state which is not in F or which never stops—then we say M refuses an input word x. We can simulate all computers of our world and their computations by deterministic Turing machines. Actually, DTM is stronger than a computer because any computer has a finite memory.

A *nondeterministic Turing machine* (or NTM) M is defined analogously as DTM. However, the transition function δ is a partially function

$$Q \times \Sigma \to \mathcal{P}(Q \times \Sigma \times \{L, N, R\}).$$

Thus, for a state $q \in Q$ of M and a character $c \in \Sigma$, the result $\delta(q, c)$ is a set of 3-tuples which describe all possibilities how the machine M can continue in the computation. The NTM M *accepts* an input word x if there is an accepting computation.

A Turing machine M decides a problem L if M accepts an input words x if and only if $x \in L$. We measure efficiency of Turing machines by time and space complexity. The *time complexity* of an Turing machine M is a function $f(n)$ such that M make at most $f(n)$ steps on an arbitrary input of length n. The *space complexity* is analogous only the number of cells used by M is bounded. However, we do not need the exact number of steps (or cells). Therefore, the $\mathcal{O}$-notation was introduced.

Definition 1.1. Let $f(n) : \mathbb{N} \to \mathbb{N}$ and $g(n) : \mathbb{N} \to \mathbb{N}$. The function $f(n)$ is in $\mathcal{O}(g(n))$ if there exist constants c, n_0 such that for all $n \geq n_0$ it holds that $f(n) \leq cg(n)$. The function $f(n)$ is in $\Theta(g(n))$ if there exist constants c_1, c_2, n_0 such that for all $n \geq n_0$ it holds that $c_1 g(n) \leq f(n) \leq c_2 g(n)$.

There is also a notation to describe a function is substantially smaller than another function. We say a function $f(n)$ is in $o(g(n)$ if a fraction $\frac{f(n)}{g(n)}$ goes to 0 when n goes to infinity.

For our thesis there are three important classes of decision problems. The class P contains all decision problems which can be decided by a deterministic Turing machine in polynomial time. Informally, a problem is in P if there is an algorithm which solve the problem in polynomial time. The class PSPACE contains all decision problems which can be decided by a deterministic Turing machine in polynomial space. The class NP contains all decision problems which can be decided by a nondeterministic Turing machine in polynomial time. An equivalent definition of NP is that NP contains all decision problems which can be verified in polynomial time. More precisely, let $L \in \mathsf{NP}$. Then, there is a polynomial size certificate $c(x)$ for $x \in L$ and a deterministic Turing machine

which decides in polynomial time if $c(x)$ is valid for x (i.e. it proves that $x \in L$). For example, the following well-known problem is in NP.

PROBLEM:	INDEPENDENT SET
Input:	Graph G, $k \in \mathbb{N}$
Question:	Does G contains an independent set of size k?

It is easy to see that INDEPENDENT SET is in NP. The certificate can be an independent set I of the size k. It is obvious we can check in polynomial time there is no edge between any two vertices in I. We can represent INDEPENDENT SET as a language of such words (over a suitable alphabet) which represents graphs with an independent set of the size k.

We say a problem $L_1 \subseteq \Sigma_1^*$ can be reduced to a problem $L_2 \subseteq \Sigma_2^*$ if there is a function $f : \Sigma_1^* \to \Sigma_2^*$ such that for all $x \in L_1$ it holds that $x \in L_1$ if and only if $f(x) \in L_2$. There can be various requirements for the function f. However, in this thesis we use only a polynomial reduction, i.e. the output of the function f can be computed in polynomial time in the length of x. We say a problem L is NP-hard (or PSPACE-hard) if every problem $L' \in$ NP (or PSPACE) can be reduced to L (under polynomial reduction). We say a problem L is NP-complete (or PSPACE-complete) if L is NP-hard (or PSPACE-hard) and $L \in$ NP (or PSPACE). To prove the hardness of a problem L we take another problem L' which is complete for the appropriate class and make a reduction from L' to L. Thus, we need to find a function f (computable in polynomial time) such that for all $x \in L'$ it holds that $x \in L'$ if and only if $f(x) \in L$. To prove the completeness of a problem L we need also to prove that L belong to the appropriate class, which is usually easy as in the case of INDEPENDENT SET. Actually, INDEPENDENT SET is NP-complete. It is easy to see INDEPENDENT SET is as hard as the following problem.

PROBLEM:	CLIQUE
Input:	Graph G, $k \in \mathbb{N}$
Question:	Does G contains a clique of size k?

CLIQUE belongs to the famous Karp's list of NP-complete problems [9].

1.1.4 Notions of Online Ramsey Theory

A *drawn graph* D is a subgraph of the background graph $G = (V, E)$ induced by the edges drawn by Builder. Formally, let $\{e_1, \ldots, e_k\} \subseteq E$ be a set of all edges drawn by Builder. The drawn graph D is

$$\big(\{v | \exists e_i : v \in e_i\}, \{e_1, \ldots, e_k\}\big).$$

In the proofs we use the notions of winning strategies of the players. Informally, a *winning strategy* is a kind of instructions how to win the game. More formally, we can view the winning strategy for a player P as a function S from sets of the adversary moves to moves of the player P. Thus, let $m_1, \ldots, m_k$ be moves of the adversary. Then, $S(\{m_1, \ldots, m_k\})$ is the move of the player P after k adversary's moves. Since only the winning strategies concern us, we sometimes omit the word winning.

There is a similar game called a graph Ramsey game. It is a typical positional game of two players. In every round the first player colors an uncolored edge

by blue and the second player colors an uncolored edge by red. A winner is the player who first colors a fixed monochromatic goal graph. The computational complexity of some versions of the graph Ramsey game were studied by Slany [18]. However, this thesis is only about the online Ramsey game, which we sometimes abbreviate to the Ramsey game.

There are a lot of graph coloring—vertex coloring, edge coloring, vertex list coloring and so on. In this thesis (according to the rules of the online Ramsey game) we use only edge 2-coloring. Formally, an edge 2-coloring of a graph G is a function $c : E(G) \to \{red, blue, uncolored\}$, which we sometimes abbreviate to coloring. A *colored* edge of the graph G colored by c is an edge $e \in E(G)$ such that $c(e) \neq uncolored$.

A *t-star* is a complete bipartite graph $G = (A \cup B, E)$ where $|A| = 1$ and $|B| = t$. The *center* of the star is the only vertex of the partition A.

Let $H = (V, F)$ be a hypergraph. A graph $G = (V, E)$ is a *host graph* for H if every hyperedge $f \in F$ induces a connected subgraph in G. The hypergraph H is a *hypertree* (or *hyperforest*) if there exists a tree (or forest) which is the host graph for H.

1.2 Offline versus Online Ramsey Theory

The online Ramsey game is related to Ramsey theory. A *Ramsey number* $r(k)$ is the least number $n \in \mathbb{N}$ such that every edge 2-coloring of the graph K_n contains a monochromatic K_k as a subgraph. A basic result of Ramsey theory is that the number $r(k)$ exists and is finite for every $k \in \mathbb{N}$. Conlon [5] studied the differences between the offline theory and the online one. He studied the online Ramsey game where the background graph is eventually infinite complete graph and the goal graph is K_k. It is clear if Builder draws $K_{r(k)}$ he certainly wins. However, Conlon proved that it often suffices less edges than $E(K_{r(k)})$ to Builder to force K_k. An *online Ramsey number* $\widetilde{r}(G)$ is the least number n such that Builder uses n edges to force G.

Theorem 1.2 (Conlon [5])**.** *There is a constant $c > 1$ such that for infinitely many values of $t \in N$ holds that*

$$\widetilde{r}(K_t) \leq c^{-t}\binom{r(t)}{2}$$

We say the graph G is *Ramsey big* for H if any edge 2-coloring of G contains a monochromatic copy of H as a subgraph. The relation between the offline and the online theory is that if G is Ramsey big for H then Builder can force a monochromatic copy of H on G. However, the opposite implication does not hold as we see from the result of Grytczuk et al. [8]. They studied Builder's strategies when Builder has to draw a graph only from fixed classes.

Definition 1.3. Let $\mathcal{C}$ be a class of graphs. A graph G is *unavoidable* on $\mathcal{C}$ if Builder wins the online Ramsey game even if the drawn graph is in $\mathcal{C}$. A class of graph $\mathcal{D}$ is *unavoidable* on $\mathcal{C}$ if every $G \in \mathcal{D}$ is unavoidable on $\mathcal{C}$.

We say a class of graph $\mathcal{C}$ is *self-unavoidable* if $\mathcal{C}$ is unavoidable on $\mathcal{C}$. Grytczuk et al. [8] proved that the classes of forests and k-colorable graphs are self-unavoidable.

This is an interesting result because if we have a forest T we can color it such that it contains only monochromatic copies of stars. We can root every component C of T in some vertex $v \in V(C)$ and color C alternatively by layers. The edges incident with v are blue, edges in the distance 1 from v are red and so on. Hence, any monochromatic subgraph of T is a star. Therefore, any forest T is not Ramsey big for any forest T' such that T' contains a path P_3 as a subgraph. Grytczuk et al. [8] also proved the cycle C_3 is avoidable on outer planar graphs and cycles are unavoidable on planar graphs. Thus, the unavoidability is not a trivial property which has every graph class.

They also conjectured that the unavoidable class on planar graphs is exactly the class of outer planar graphs. This was disproved by Petříčková [12]. She proved outer planar graphs are unavoidable on planar graphs. Moreover, she found an infinite subclass $\mathcal{C}$ of planar graphs which is unavoidable on planar graphs and the graphs in $\mathcal{C}$ are not outer planar.

Definition 1.4. Let $\theta_{i,j,k}$ be a graph consisting of 2 vertices u, v and 3 internally disjoint paths P_i, P_j, P_k connecting u and v.

Theorem 1.5 (Petříčková [12])**.** *Let $j, k \in \mathbb{N}$ be even. The graph $\theta_{2,j,k}$ is unavoidable on planar graphs.*

A part of the unavoidability is a study of an online degree Ramsey number.

Definition 1.6. Let H be a goal graph. An *online degree Ramsey number* $\widetilde{r}_\Delta(H)$ is the least number k such that H is unavoidable on graphs with the maximum degree k.

Several results are known about online degree Ramsey numbers. Some bounds were found by Butterfield et al. [4]. They proved a lower bound for general graphs and they made some bounds for specific graph classes:

1. $\widetilde{r}_\Delta(H) \geq \Delta(H) + t - 1$ where
$$t = \max_{\{u,v\} \in E(H)} \min\{\deg(u), \deg(v)\}.$$
2. $\widetilde{r}_\Delta(H) \leq 3$ if and only if H is a disjoint union of paths or each component of H is a subgraph of 3-star.
3. $\widetilde{r}_\Delta(H) \leq d_1 + d_2 - 1$ for a tree H and d_1, d_2 are two largest vertex degrees.
4. $4 \leq \widetilde{r}_\Delta(C_n) \leq 5$, with $r_\Delta(C_n) = 4$ except finitely many values of n.
5. $\widetilde{r}_\Delta(H) \leq 6$ for H with the maximum degree at most 2.

Rolnick [15] completed the result about cycles and proved $r_\Delta(C_n) = 4$. In his previous work, Rolnick [14] described a class of trees with the online degree Ramsey number at most 4.

Definition 1.7. A tree T is a *maple* if the maximum degree of T is at most 3 and there are no two neighbors with degree 3.

Theorem 1.8 (Rolnick [14])**.** *Let T be a tree. Then, $\widetilde{r}_\Delta(T) \leq 4$ if and only if T is a maple or a 4-star.*

Another topic in online Ramsey theory is the optimality of strategies and computing the exact online Ramsey number for specific graphs. We say a Builder's strategy for a goal graph H is optimal if he can force H on a graph with k edges and he cannot force H on any graph with at most $k-1$ edges. However, it seems this topic is extremely hard and as far as we know there are not many results about it. Let $f(n)$ be the maximum value of $\widetilde{r}(T)$ over all trees T with n edges. Grytczuk et al. [7] proved that $f(n)$ is in $\Theta(n^2)$. Moreover, they proved $\widetilde{r}(P_n) \leq 4n-7$ and some other upper and lower bounds. Belfrage et al. [3] computed the optimal strategies for the paths of length 1 to 10 when Builder has to draw only forests. Prałat [13] computed the online Ramsey numbers for the paths P_7 and P_8 and made a bound for the path P_9. However, computers were used to search for the optimal strategy in both results about the paths.

1.3 Our Results

In this thesis we study two topics of online Ramsey theory. We study the computational complexity of the decision problem if Builder has a winning strategy on a given game. As we saw there is a lot of versions of the online Ramsey game. We ask a question how hard is to compute if Builder can win some version of the online Ramsey game on the background (partly precolored) graph G with the goal graph H. We formalize different versions of the game as rules for both players and we define the online Ramsey game as a decision problem.

PROBLEM:	Online Ramsey Game
Input:	Partly edge-colored background graph G, goal graph H.
Rules:	Rules for Builder and Painter—rules have to be independent on the input and a decision problem if a move is valid is in PSPACE (measure against the size of the graph G)
Question:	Can Builder force a monochromatic copy of H in G if both players make only valid moves?

We prove the problem Online Ramsey Game is PSPACE-complete even for the bipartite background graph of the maximum degree 3 and the goal graph as a tree. Moreover, we also found a version of the problem which is NP-complete. Our results about the complexity are in Chapter 2.

The second topic we study are strategies for Builder. We study three versions of the game. First one is a version where Builder can draw only planar graphs. We found a new class which is unavoidable on planar graphs.

Theorem 1.9. *Let $Q(k,\ell,u,v)$ be a graph consisting of ℓ internally disjoint paths of length k between the vertices u and v. The graph $Q(k,\ell,u,v)$ is unavoidable on planar graphs for arbitrary $\ell \in \mathbb{N}$ and $k \in \mathbb{N}$ odd.*

The second version is a generalization of the game where Builder can draw 3-uniform hyperforests. We prove 1-degenerate 3-uniform hyperforests are unavoidable on 3-uniform hyperforests. The third version is a game where Builder can draw only forests. We remark the strategy of Grytczuk et al. [8] is not opti-

mal and we present a slight improvement of the strategy. The results about the strategies are in Chapter 3.

2. Computational Complexity of Online Ramsey Game

In this chapter we study the computational complexity of Online Ramsey Game and other similar problems. Note that we can bound the input size of Online Ramsey Game by the size of graph G (up to a multiplicative constant 2). The goal graph H has to be smaller than the background graph G, otherwise Builder has no chance to win. Thus, the size of the input is fully determined by the size of G. Actually for many Builder's strategies, the goal graph is usually much smaller then the background graph.

It is easy to see that Online Ramsey Game is in PSPACE regardless of the rules. We may use a simple depth-first-search over the set of valid moves such that in each position the algorithm iterates over all valid moves and recursively searches each of the moves for a solution. The maximum depth of the recursion is linear in the size of G. The number of moves which the algorithm has to keep in memory in every step of the recursion is also linear in the size of G. Therefore, the algorithm needs only memory of the polynomial size of G.

In this chapter we show Online Ramsey Game is PSPACE-complete. It is not very surprising because many other decision problems about the existence of game strategies are PSPACE-complete. However, it is PSPACE-complete even for quite simple input. The proof is made by polynomial reduction from other PSPACE-complete problem. In the first section we present formula games which we use for the reduction. In Section 2.2 we show a hardness proof of a little bit relaxed problem where we demonstrate basic ideas of the proof. The final hardness proof of Online Ramsey Game is in Section 2.3. A version of Online Ramsey Game which is NP-complete is presented in Section 2.4.

2.1 Formula Games

For the reduction we use a type of a formula game. The standard formula game is played on a proposition formula φ in CNF with variables $x_1, y_1, x_2, y_2, \ldots, x_n, y_n$. In a round i the first player evaluates x_i by 0 or 1 and the second player evaluates y_i. The game ends when all variables are evaluated. The first player wins if the formula φ is true at the end of the game. The second player wins if φ is false. The first player wins if and only if there is an evaluation of x_1 such that for all evaluation of y_1 there is an evaluation of x_2 and so on to an evaluation of y_n such that formula φ is true after all these evaluations. Thus, the first player wins if and only if a formula $\exists x_1 \forall y_1 \exists x_2 \forall y_2 \ldots \exists x_n \forall y_n \varphi$ is true. Therefore, the question who win the formula game is only a reformulated decision problem if a quantified formula is true or false, which is a well-known PSPACE-complete problem [19].

We use a slightly different formula game, which is also played on a formula φ in CNF. However, the variables are paired and the first player decides which pair is evaluated at the beginning of each round.

PROBLEM: PAIRED FORMULA GAME
Input: Formula φ in CNF, pairs of variables

$$Z = \{(x_1, y_1), \ldots, (x_n, y_n)\}.$$

The formula φ contains only variables from the pairs in Z.
Rules: In a round j the first player picks a pair (x_i, y_i) and evaluates x_i and the second player evaluates y_i. The game ends when all variables are evaluated. The first player wins if and only if φ is true at the end of the game.
Question: Can the first player win this instance?

There are many type of the formula games which are PSPACE-complete. They were studied for example by Schaefer [17], whose proofs inspired us when we proved the completeness of PAIRED FORMULA GAME. However before we prove the completeness, we define an auxiliary game. Let J_ℓ be an equation system over $\mathbb{Z}_2$ consisting[1] of an equation $a_1 = b_1$ and for every $2 \leq i \leq \ell$ of an equation $b_i = b_{i-1} + a_i$. Let *equation game* be a game on the system J_ℓ of two players with the following rules. In each round the first player picks a variable which will be evaluated. If the picked variable is b_i for some $i \in [\ell]$ then the first player evaluates it by 0 or 1. Otherwise, the second player evaluates it. The game ends when all variables are evaluated. The first player wins if all equations in J_ℓ holds after the variable evaluation. Otherwise, the second player wins.

It is easy to see that the first player wins. He can pick first all variables a_i and then he evaluates all variables b_i such that all equations in J_ℓ holds. However, if he picks the variables in wrong order he can lose.

Lemma 2.1. *If there exist $j \leq i \in [\ell]$ such that the variable a_j is picked after the variable b_i then the second player wins the equation game on J_ℓ.*

Proof. We prove the lemma by induction on ℓ. The only equation of J_1 is $a_1 = b_1$. By the assumptions a_1 is picked after b_1. Thus, the second player evaluates a_1 by $1 - b_1$ and he wins the game.

Suppose the lemma holds for $\ell = k - 1$. We prove the lemma for the system J_k. If a variable a_j is picked after a variable b_i for $j \leq i < k$ then the second player wins by induction hypothesis. Let a_j be the first variable picked after b_k for $j \leq k$. If $j < k$ then the variable a_k have been already evaluated and the variable b_{k-1} has a determined value from the equation $b_k = b_{k-1} + a_k$. Therefore, we can suppose b_{k-1} has been already evaluated. The second player wins by induction hypothesis because b_{k-1} has been picked before a_j.

It remains the case a_k is picked after b_k. Suppose the variable b_{k-1} has a known value, i.e. it has been picked or b_{k-2} and a_{k-1} have known values. Hence, the second player sets a_k such that the equation $b_k = b_{k-1} + a_k$ does not hold. Let b_{k-1} has not a known value. Suppose there exists $i \in [k-1]$ such that a_i has not been picked. When the second player evaluates a_k then he determined the value of b_{k-1}. Therefore, he can win by induction hypothesis. Suppose all $a_1, \ldots, a_{k-1}$

[1] Variables has value 0 or 1 and an addition result is reduced by modulo 2—i.e. $1 + 1 = 0$.

have been picked. The second player determined all values of $b_1, \ldots, b_{k-1}$ by the evaluation of a_k. Thus, he evaluates a_k such that the equation $a_1 = b_1$ does not hold. □

We can view a variable of an equation over $\mathbb{Z}_2$ also as a formula variable. If x as the equation variable is equal to 1 than x as the formula variable is true and vice versa. Thus, the equation $E : x = y$ over $\mathbb{Z}_2$ can be view as an abbreviation for the formula $\varepsilon = (x \vee \neg y) \wedge (\neg x \vee y)$. The equation $S : x = y + z$ over $\mathbb{Z}_2$ can be viewed as an abbreviation for

$$\sigma = (x \vee y \vee \neg z) \wedge (x \vee \neg y \vee z) \wedge (\neg x \vee y \vee z) \wedge (\neg x \vee \neg y \vee \neg z).$$

Note that the equation E (or S) holds if and only if the formula ε (or σ) is true.

Theorem 2.2. Paired Formula Game *is PSPACE-complete.*

Proof. It is clear the problem is in PSPACE by the same argument as Online Ramsey Game is in PSPACE. For proving the hardness we make a reduction from the standard game. Let formula φ in CNF with variables $V = \{x_1, y_1, \ldots, x_n, y_n\}$ be a formula of the standard game. We create pairs with new variables

$$Z = \big\{(x_1, a_1), (b_1, y_1), \ldots, (x_n, a_n), (b_n, y_n), \big\}$$

and new formulas

$$\psi' = \bigwedge_{2 \le i \le n} (b_i = a_i + b_{i-1}) \wedge (b_1 = a_1)$$
$$\psi = \psi' \wedge \varphi.$$

The abbreviate formulas $x = y$ and $x = y + z$ are discussed above. The input for Paired Formula Game is the formula ψ with the variable pairs Z. Informally, the formula ψ' forces the first player to pick pairs with $x_1, \ldots, x_i$ before the pair with y_i for each i. It is clear the size of the formula ψ with the set Z is linear in the size of the formula φ with the set V. To finish the proof we need to prove the first player wins the standard game if and only if he wins the paired game.

Suppose the first player has a winning strategy S for the standard game on φ. We create the first player strategy for the paired game. The first player just picks pairs in the order $(x_1, a_1), (b_1, y_1), \ldots, (x_n, a_n), (b_n, y_n)$. In a round $2i-1$ the pair (x_i, a_i) is picked. The variables y_j for all $j < i$ have been already evaluated. Therefore, the first player can use the strategy S to evaluate x_i. In a round $2i$ the pair (b_i, y_i) is picked. If $i = 1$ then the first player evaluates b_1 by the value of a_1, which has been evaluated in the round 1. Otherwise, the first player evaluates b_i in a way that the subformula $b_i = a_i + b_{i-1}$ would be true, which he can do because a_i and b_{i-1} have been already evaluated. Since the variables x_i for every i have been evaluated according to S, the formula φ is true at the end of the game. Every clause in ψ' is also true. Therefore, the first player wins the paired game.

Now suppose the second player has a winning strategy S for the standard game. The first player plays soundly if he always picks a pair (b_i, y_i) after (x_j, a_j) for all $j \le i$. Suppose the first player plays soundly. If a pair (b_i, y_i) is picked then all variables x_j for every $j \le i$ have been evaluated. Therefore, the second

player can use S to evaluate all variables y_i for every i. The second player wins the game because at the end φ is false.

Suppose a pair (x_j, a_j) is picked after (b_i, y_i) for some $j \leq i$. Thus, the second player cannot use the strategy S for an evaluation of y_i. However, the second player can make the formula ψ' false by Lemma 2.1. □

Actually, Paired Formula game is still PSPACE-complete for more restricted input.

Definition 2.3. Let φ be a proposition formula. The formula φ is in *3-CNF* if it has at most 3 variables in each clause. The formula φ is in *3-bounded 3-CNF* if it is in 3-CNF and every variable has at most 3 occurrences in the formula φ.

In the proof we use the auxiliary formula

$$E(z_1, \ldots, z_k) = (z_k \vee \neg z_1) \wedge \bigwedge_{2 \leq j \leq k} (z_{i-1} \vee \neg z_i).$$

Observation 2.4. *The formula $\varepsilon = E(z_1, \ldots, z_k)$ is true if and only if all variables $z_1, \ldots, z_k$ have the same value.*

Proof. In every clause of the formula ε there is a positive and a negative literal. Therefore if all variables $z_1, \ldots, z_k$ are true or false every clause of ε has to be true. On the other hand, if there are $z_i \neq z_j$ then there are $z_\ell, z_{\ell+1}$ (in cyclic order, i.e. $z_{k+1} = z_1$) such that $z_\ell = 0$ and $z_{\ell+1} = 1$. Therefore, the clause $z_\ell \vee \neg z_{\ell+1}$ is false. □

Theorem 2.5. Paired Formula game *is PSPACE-complete even when it is played on a formula in 3-bounded 3-CNF.*

Proof. Actually, Stockmeyer [19] proved deciding if quantified formula in 3-CNF is true is PSPACE-complete. Our reduction in the proof of Theorem 2.2 preserves 3-CNF. Therefore, Paired Formula Game is PSPACE-complete on formulas in 3-CNF. It remains to prove the theorem for formulas in 3-bounded 3-CNF.

Let φ in 3-CNF and variable pairs $Z = \{(x_1, y_1), \ldots, (x_n, y_n)\}$ be an input for Paired Formula Game. We create a new formula ψ with new variable pairs Z' such that ψ would be in 3-bounded 3-CNF and the first player wins the paired formula game on φ if and only he wins the game on ψ. We replace every occurrence of each variable in φ by a new variable. Let $(x_i, y_i) \in Z$. Suppose x_i has k occurrences in φ and y_i has ℓ occurrences in φ. To Z' we add pairs

$$(x_i^1, y_i^1), (x_i^2, a_i^2) \ldots, (x_i^k, a_i^k), (y_i^2, b_i^2), \ldots, (y_i^\ell, b_i^\ell).$$

Let φ' be a formula obtained from φ by replacing the j-th occurrence of the variable x_i by the variable x_i^j (same with the second player variable y_i). We denote by $o(z)$ the number of the variable z occurrences in φ. We set new formulas

$$\psi' = \bigwedge_{1 \leq i \leq n} \left(E(x_i^1, \ldots, x_i^{o(x_i)}) \wedge \left(E(y_i^1, \ldots, y_i^{o(y_i)})\right)\right.$$
$$\psi = \varphi' \wedge \psi'.$$

It is clear ψ is in 3-bounded 3-CNF. For abbreviation, we denote by φ-game the original game on the formula φ with the variable pairs Z and by ψ-game the new constructed game on the formula ψ with the variable pairs Z'.

Suppose the first player has a winning strategy S in the φ-game. We construct a winning strategy S' for the ψ-game. The strategy S' has two phases. In the first phase the first player follows the strategy S. Suppose the first player picks (x_i, y_i) in a round j of the φ-game. In a round j of the ψ-game he picks (x_i^1, y_i^1). He evaluates x_i^1 by the same value as he evaluates the variable x_i in the φ-game. He continues to copy the strategy S according to the evaluation of y_i^1 as if the variable y_i is set to the same value in the φ-game.

The first phase ends when all pairs (x_i^1, y_i^1) have been picked. In the second phase the first player picks all remaining pairs in arbitrary order. Note that all second player variables which have occurrence in ψ have been evaluated in the first phase. Thus, in this phase the second player has no chance to change the outcome of the ψ-game. The first player evaluates x_i^j (or y_i^j) by the value of x_i^1 (or y_i^1).

For every φ-variable z and the new variables $z^1, \ldots, z^k$ which replace all occurrences of z in φ it holds that variables $z^1, \ldots, z^k$ have the same value. Thus, the formula ψ' is true by Observation 2.4. The strategy S is the winning strategy for the first player in φ-game. Therefore, φ' has to be true and the first player win the ψ-game.

Suppose the second player has a winning strategy S in the φ-game. The first player plays unsoundly if there exists i and $j \neq k$ such that $x_i^j \neq x_i^k$ or $y_i^j \neq y_i^k$. Suppose the first player plays unsoundly. Thus, the formula ψ' has to be false by Observation 2.4 and the second player wins the ψ-game. From now we can assume the first player plays soundly. This also implies the first player picks (x_i^1, y_i^1) before any pair (y_i^j, b_i^j). If a pair (x_i^1, y_i^1) is picked after (y_i^j, b_i^j) then the second player evaluates y_i^1 by the opposite value of y_i^j and makes the formula ψ' false.

If the first player picks a pair (x_i^j, a_i^j) (or (y_i^j, b_i^j)) the second player can evaluate the variable a_i^j (or b_i^j) arbitrarily because it does not affect the outcome of the ψ-game. Suppose the first player picks (x_i^1, y_i^1) and let X be a set of the first player variables in the ψ-game which have been already evaluated. Let

$$I = \{j | \exists k : x_j^k \in X\}.$$

The second player evaluates y_i^1 by the same value as he would evaluate y_i in the φ-game when the variables $x_j, j \in I$ have been already evaluated. Since S is the winning second player strategy, the formula φ has to be false. Hence, the formulas φ' and ψ have to be false as well. □

2.2 Multiple Ramsey Game

We define more general problem where Builder's goal is to force several monochromatic copies of the goal graph. However, these copies have not to be colored by the same color.

PROBLEM:	Multiple Online Ramsey Game
Input:	Partly edge-colored background graph G, goal graph H, repetition parameter $m \in \mathbb{N}$.
Rules:	Rules for Builder and Painter—rules have to be independent on input and decision problem if a move is valid is in PSPACE (measure against the size of the graph G).
Question:	Can Builder force m monochromatic copies of H in G if both players make only valid moves?

Online Ramsey Game is a version of Multiple Online Ramsey Game where the repetition parameter m equals to 1. Let us have a multiple game with the goal graph H and a standard game with the goal graph consisting of exactly m vertex disjoint copies of H. To win the standard game as Builder all copies of H must have the same color, which is not necessary in the multiple game.

In this section we prove Multiple Online Ramsey Game is PSPACE-complete. The problem is in PSPACE by the same reason as Online Ramsey Game, which is discussed at the beginning of this chapter. Therefore, it remains to prove the hardness of the problem. We prove it by reduction from Paired Formula Game. In all reductions we call the formula game players as the first and the second player and the Ramsey game players as Builder and Painter. We will use the following type of graphs.

Definition 2.6. Let (n, a, b)-*shackles* be a graph consisting of two disjoint cycles C_a and C_b connected by a path P_n. The path P_n is called a *chain*. If n is even we call the center vertex of P_n as the *center of shackles*. The edge of P_n connected to the cycle C_a (or C_b) is called a-connection (or b-connection) and the number a (or b) is a *parameter* of the connection.

In our proof copies of $(4, 4, 6)$-shackles represent occurrences of the formula variables. All these shackles are precolored in the same way. Edges of the cycles are precolored alternatively by red and blue and the connecting paths will be red, except the connections. The connection type edges have no color and they can be drawn by Builder. The cycle vertices which are closest to the connecting path are called *locks*. A color of the lock is determined by a color of the edge which connects the lock and the chain endpoint. For better understanding the definition of the shackles and how it is precolored see Figure 2.1.

Let x be a variable of formula φ and C be a clause. By $x \in_j C$ we mean the j-th occurrence of variable x is positive and it is in C. If the j-the occurrence of x is negative we denote it by $\neg x \in_j C$.

Definition 2.7. Let (n, a)-*lollipop* be a graph consisting of a cycle C_a with an attached path P_n.

In the proof, $(6, 8)$-lollipop is the goal graph H. We call (n, a)-lollipop almost complete if all its edges have the same color except the last edge of the lollipop path (the one which is the furthest from the lollipop cycle). We say the lollipop is completed when the last uncolored lollipop edge is colored by the color which have the rest of the lollipop edges.

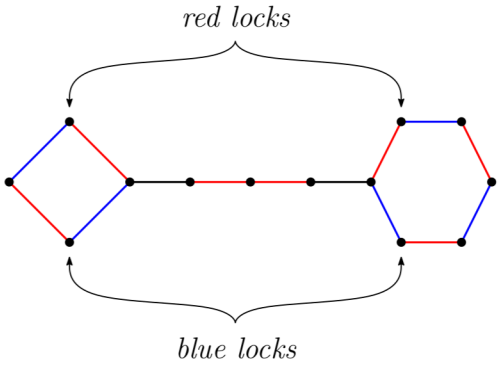


Figure 2.1: Figure of $(4, 4, 6)$-shackles with the black connections and marked locks.

2.2.1 Construction of the Background Graph

In this section we describe the background graph and the goal graph for the Ramsey game which we create from a given formula of the paired formula game. Let formula φ and pairs of variables $Z = \{(x_1, y_1), \ldots, (x_n, y_n)\}$ be an input for Paired Formula Game. We construct the background graph $G(\varphi, Z)$ containing gadgets for variables and clauses.

First, we describe a *variable gadget*. Let (x, y) be a variable pair in Z. For all occurrences $x^1, \ldots, x^k, y^1, \ldots, y^\ell$ of the variables (x, y) in φ we create $(4, 4, 6)$-shackles and precolor them as it is described above (see Figure 2.1). We denote the shackles for the j-th occurrence of a variable z by $S_j(z)$. We add one more vertex which is connected by blue edges to the centers of the shackles $S_1(x), \ldots, S_k(x), S_1(y), \ldots, S_\ell(y)$. See Figure 2.2 for an example of such a gadget. We denote the whole gadget for the variable pair (x, y) by $R(x, y)$.

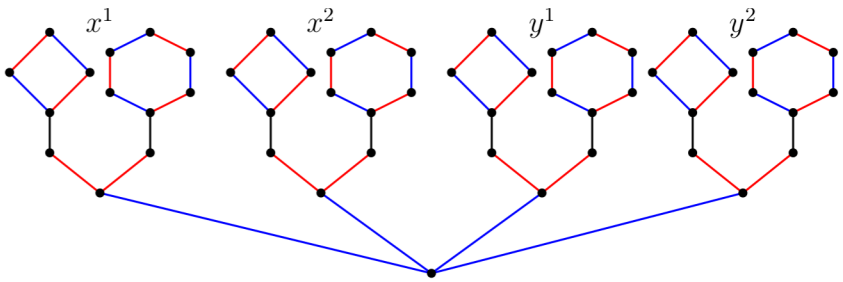


Figure 2.2: Example of a gadget $R(x, y)$ when both variables (x, y) have 2 occurrences. Almost all edges are precolored, only 4- and 6-connections can be drawn by Builder.

The idea of equivalence between strategies in the formula game and the Ramsey game is as follows. If the first player sets a variable x to 1 then in the corresponding gadget $R(x, y)$ only 4-connections are drawn by Builder. If $x = 0$ then only 6-connections are drawn. If the second player sets a variable y to 1 then all drawn edges in the gadget $R(x, y)$ are colored by blue, otherwise red.

For consistency of the equivalence we need to assure that in every variable gadget Builder draws only connections of one parameter and Painter colors all drawn edges by one color. We force it by two simple rules.

Rule 1 Builder has to draw all edges such that the drawn graph does not contain $(4,4,6)$-shackles or $(6,4,6)$-shackles as a subgraph.

Rule 2 Painter has to color the edges such the every $(6,4,4)$-shackles or $(6,6,6)$-shackles in the drawn graph contains an even number of red edges and an even number of blue edges.

Lemma 2.8. *Let variable gadget $R(x,y)$ for a pair (x,y) be a background graph for the Ramsey game restricted by Rules 1 and 2. Let Builder draws a connection e of the parameter p and Painter colors e by a color c in the first round of the game. Then for the rest of the game, Builder has to draw only connections of the parameter p and Painter has to color all drawn edges by the color c.*

Proof. Suppose Builder draws a 4-connection e and Painter colors it by blue in the first round. If Builder draws a 6-connection e_1 in any other round then $(4,4,6)$-shackles arises if e and e_1 are in the same $S_j(x)$ or $S_j(y)$ otherwise $(6,4,6)$-shackles arises. Every $(6,4,4)$-shackles and $(6,6,6)$-shackles in $R(x,y)$ has an even number of blue edges and an even number of red edges at the beginning of the game. Therefore, if Painter colors a 4-connection by red in any round i for $i > 1$ then $(6,4,4)$-shackles with an odd number of red edges and an odd number of blue edges arises. The other cases of moves are treated similarly. □

Observation 2.9. *A valid move can be checked in time polynomial in the size of the background graph.*

Proof. Rules forbid only structures up to 15 vertices. Therefore, we can check in polynomial time if there is a forbidden structure in any 15-tuple of the background graph vertices. □

Let C be a clause of the formula φ. We create a *clause gadget* $L(C)$ consisting of a red and a blue 8-cycle connected by one vertex $v(C)$. We connect $v(C)$ by paths of length 4 to the locks of the shackles which represents variables occurring in C. There is a $(4,4,6)$-shackles for every variable occurrence. Therefore, every $(4,4,6)$-shackles $S_j(z)$ is connected only to the one clause gadget. The connection clause and variable gadgets is quite technical, for better understanding see Figure 2.3.

As we stated before the 4-connections represent positive occurrences of the first player variables and the 6-connections represent negative ones. Let x be a first player variable such that $x \in_j C$. We connect $v(C)$ to the blue lock of 4-cycle in the variable shackles $S_j(x)$ by a blue path. We connect $v(C)$ to the red lock of 4-cycle in $S_j(x)$ by a red path. If $\neg x \in_j C$ then we connect $v(C)$ to the locks of the 6-cycle in $S_j(x)$ in the same way.

Blue color represents positive occurrences of the second player variables and red color represents negative ones. Let y be a second player variable such that $y \in_j C$. We connect $v(C)$ to the blue locks of both cycles in the variable shackles $S_j(y)$ by blue paths. If $\neg y \in_j C$ we connect $v(C)$ to the red locks of both cycles in $S_j(y)$ by red paths. This finished the construction of the background graph $G(\varphi, Z)$ for the Ramsey game. Note that all edges are precolored only the connection type edges in the variable gadgets can be drawn by Builder. We show some easy but important observations about the background graph and the rules.

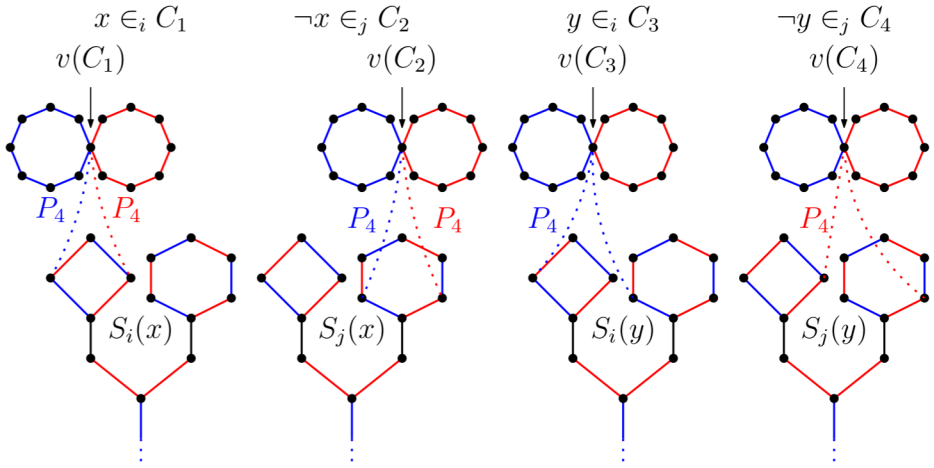


Figure 2.3: Connection of clause and variable gadgets. Almost all edges are precolored only 4- and 6-connections can be drawn by Builder. Every clause gadget is connected to the variable gadget by a path of length 4, which are depicted as dotted arcs and lines.

Observation 2.10. *The size of the graph $G(\varphi, Z)$ is linear in the size of the formula φ.*

Proof. Let n be the number of occurrences of all variables in φ and m be the number of clauses in φ. The size of all clause gadgets is linear in m and the size of all variable gadgets with connections to the clause gadgets is linear in n. □

Observation 2.11. *Let $R(x, y)$ be a variable gadget in the graph $G(\varphi, Z)$. Let e be the first Builder's move in $R(x, y)$ and Painter colors it by a color c. According to Rules 1 and 2:*

1. *Builder can draw e as a 4- or 6-connection and painter can color e by red and blue regardless of moves in the other variable gadgets in $G(\varphi, Z)$.*
2. *Builder can draw all other connections in $R(x, y)$ of the parameter of e.*
3. *Painter can color all drawn edges in $R(x, y)$ by the color c.*

Proof. Rules forbid only some type of shackles of the chain length up to 6. The shortest path between two variable gadgets has at least 8 edges. Thus, there cannot be some forbidden structure using vertices and edges of two variable gadgets. All 4-cycles and 6-cycles in $G(\varphi, Z)$ are only in variable gadgets. Thus, any forbidden structure can be only inside single variable gadget. Therefore, moves in $R(x, y)$ are not affected by any moves in the other variable gadgets. □

2.2.2 Hardness of Multiple Ramsey Game

In this section we show the hardness of Multiple Online Ramsey Game. We use the graph containing variable and clause gadgets described in the previous section.

Theorem 2.12. Multiple Online Ramsey Game *is PSPACE-hard.*

Proof. Let formula φ and variable pair $Z = \{(x_1, y_1), \ldots, (x_n, y_n)\}$ be an input for Paired Formula Game. We create the background graph $G(\varphi, Z)$ as it is described above and use Rules 1 and 2 for the Ramsey game. By Observation 2.9 and Observation 2.10 Rules 1 and 2 and the graph $G(\varphi, Z)$ are suitable input for Multiple Online Ramsey Game to make a reduction.

The goal graph H is a $(6, 8)$-lollipop. We set the repetition parameter m to the number of clauses in φ. There are a lot of almost complete $(6, 8)$-lollipops in the graph $G(\varphi, Z)$. Therefore, it is sufficient for Builder to complete m of these these lollipops to win the game. We claim the first player wins the formula game on the formula φ with variable pairs Z if and only if Builder wins the Ramsey Game on the constructed background graph $G(\varphi, Z)$ with the goal graph H and the repetition parameter m.

Suppose the first player has a winning strategy S for the formula game. We create a winning Builder's strategy S' for the Ramsey game. The interpretation of the Ramsey game moves to the formula game moves is as follows. If Builder draws a 4-connection in a variable gadget $R(x, y)$ then first player evaluates x by 1, otherwise by 0. If Painter colors a drawn edge in a variable gadget $R(x, y)$ by blue then the second player evaluates y by 1, otherwise by 0. The strategy S' has two phases.

The first phase is copying the strategy S in the Ramsey game. In a round i Builder interprets all moves of both players as it was described. Let (x, y) be a pair which have to be picked according to S in the round i of the formula game. Builder draws an edge e in the variable gadget $R(x, y)$. Note that e is the first drawn edge in $R(x, y)$. If $x = 1$ then Builder draws an arbitrary 4-connections in a variable gadget $R(x, y)$ otherwise he draws arbitrary 6-connections. The first phase ends after n rounds. Thus, in the moment when the formula game would end. There is exactly one connection type edge drawn (and colored) in every variable gadget.

The second phase of S' is just drawing remaining edges in all variable gadgets in a way to not violate the rules. Thus, if in a variable gadget $R(x, y)$ is drawn a 4-connection from the first phase then Builder draws all 4-connections in the gadget $R(x, y)$ and vice versa with 6-connections. By Lemma 2.8, moves in this phase are fully determined by moves in the first phase and players have no possible choice (up to the order of drawn edges by Builder, which is not important in this phase). All Builder's moves are valid by Observation 2.11.

Since the strategy S is a Builder's winning strategy, every clause of φ has a literal which is true at the end of the formula game. Let C be an arbitrary clause of φ and ℓ be a literal which is true.

Suppose ℓ is the first player variable x. Let $x \in_j C$. Hence, x is set to 1 and only the 4-connections are drawn in a variable gadget $R(x, y)$ where y is the second pair variable paired with x. It does not matter if the drawn edges in $R(x, y)$ are blue or red because red and blue 8-cycles from the clause gadget $L(C)$ are connected to the 4-cycle locks in the variable shackles $S_j(x)$. Therefore, the drawn 4-connection e from $S_j(x)$ completed a red or blue lollipop whose cycle is in $L(C)$. The case when $\neg x \in_j C$ is similar, only the 6-connection is used.

Suppose ℓ is the second player variable. Let $y \in_j C$. Hence, the connection type edges are colored by blue in a variable gadget $R(x, y)$. It does not matter if the drawn edges in $R(x, y)$ are 4-connections or 6-connections because blue

8-cycle from $L(C)$ is connected with the blue locks of the variable shackles $S_j(y)$. Therefore, the drawn blue edge e completed a blue lollipop. The case when $\neg y \in_j C$ is similar only red color is used. Builder forces one monochromatic copy of H for every clause of φ and each of these copies are disjoint. Therefore, he wins the Ramsey game.

Suppose the second player has a winning strategy S in the formula game. We construct a winning strategy for Painter. Let Builder draws an edge e in a variable gadget $R(x, y)$. If Builder has already drawn another edge e' in $R(x, y)$ then Painter colors e by the same color as he has colored the edge e'. Otherwise, he would violate the rules. If the edge e is the first drawn edge in $R(x, y)$ then he colors it according to the strategy S. He interprets all his and Builder's moves to the moves of the formula game as it is described above. Strategy S described how to evaluate the variable y. Painter colors e by blue if $y = 1$, otherwise by red. By Observation 2.11 all Painter's moves are valid.

Since the strategy S is a winning for the second player, there exists a clause C which is false. Thus, all literals of C are false. We claim there is no complete $(6, 8)$-lollipop containing any 8-cycle from the clause gadget $L(C)$. Let (x, y) be a variable pair. Suppose $x \in_j C$. Thus, x is set to 0 and only 6-connections are drawn in the variable gadget $R(x, y)$. Since the clause gadget $L(C)$ is connected to the 4-cycle of $S_j(x)$ there is no complete $(6, 8)$-lollipop containing any 8-cycle from $L(C)$ and the 6-connection in $S_j(x)$. The case $\neg x \in_j C$ is similar.

Suppose $y \in_j C$. Thus, y is set to 0 and all drawn edges in the variable gadget $R(x, y)$ are red. However, both 8-paths from $L(C)$ to $S_j(y)$ are blue. Therefore, there is no complete $(6, 8)$-lollipop containing any 8-cycle from $L(C)$ and the red connection type edge in $S_j(y)$. The case $\neg y \in_j C$ is similar. Hence, the 8-cycles from $L(C)$ cannot be in any complete $(6, 8)$-lollipop.

There are $m - 1$ other clause gadgets. Since both 8-cycles in every clause gadget are connected there cannot be two vertex disjoint $(6, 8)$-lollipops whose 8-cycles are from the same clause gadget. Note that 8-cycles in the background graph G are only inside the clause gadgets. Therefore, Builder can force only at most $m - 1$ monochromatic vertex disjoint $(6, 8)$-lollipops and Painter wins the Ramsey game. □

2.3 Online Ramsey Game

In this section we prove Online Ramsey Game is PSPACE-complete for quite simple background and goal graphs and when Painter has no rule. The main task is to remove the repetition parameter m from Multiple Online Ramsey Game. The main idea is to double the background graph and switch the color in one copy. We need to force Painter that he colors corresponding edges in both copies by different colors. In speak of a reduction, if Painter is forced to color k blue goal graphs and $m - k$ red goal graphs in the original game then after doubling Painter is forced to color k blue and $m - k$ red goal graphs in the first copy and $m - k$ blue and k red goal graphs in the second one. Hence, Painter is forced to color m blue and m red copies of the goal graph. We can set the new goal graph as m vertex disjoint copies of the original goal graph. Therefore, we can remove the repetition parameter.

Let e_1, e_2 be edges such that we want Painter colors these edges by different

colors. Suppose the goal graph H has two leaves v_1, v_2. Let u_1, u_2 be the only neighbors of v_1, v_2 and H' be a copy of H without the vertices v_1, v_2. We attach the red and blue copy of H' by u_1, u_2 to the endpoints of e_1 and e_2. Therefore, if Painter colors edges e_1 and e_2 by the same color he loses immediately. For better understanding the idea see Figure 2.4. As we state later by this construction we also remove rules for the Painter. We formalize these ideas in the next section.

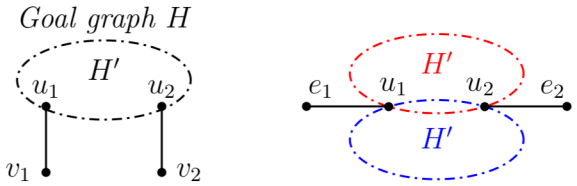


Figure 2.4: If Painter colors the edges e_1 and e_2 by the same color he loses immediately because a monochromatic copy of the goal graph H arises.

2.3.1 Construction of the Restricted Background Graph

We again use a polynomial reduction from Paired Formula Game to prove the hardness of Online Ramsey Game. The gadgets would by similar with gadgets for Multiple Ramsey Game. However, we want to restrict the background and the goal graph. We construct the bipartite background graph of the maximum degree 3 and the goal graph as a tree.

Let formula φ and variable pairs Z be an input for Paired Formula Game. By Theorem 2.5, we can assume the formula φ is in 3-bounded 3-CNF.

First, we describe the goal graph $H = H(\varphi, Z)$ because part of it will appear in variable gadgets. Let m be a number of clauses in φ. The goal graph H consists of a main path P' of length $2m - 2$ (i.e. it has $2m - 1$ vertices) and there is a path P_{12} attached to every odd vertex of P'. Thus, there are m paths of length 12 attached to P'. Let $H' = H'(\varphi, Z)$ be a graph obtain in the same way as H, however there are paths of length 10 attached to the endpoints of P' instead of P_{12}. The endpoints of P_{10} in H' are called gadget vertices. By the gadget vertices the graph H' will be attached to variable gadgets to force Painter to color some edges by different colors. Graphs H and H' are depicted in Figure 2.5.

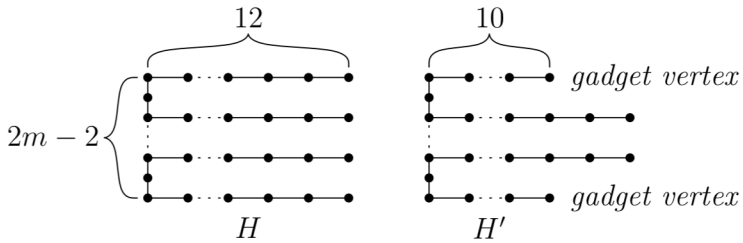


Figure 2.5: The goal graph H and the graph H' which will be used in variable gadgets.

A variable gadget is similar to the gadget from the proof of Theorem 2.12. However, every occurrence of each variable is represented by two shackles. For the j-th occurrence of the variables z in φ we create two $(6, 4, 6)$-shackles. Cycles

of these shackles are precolored in the same way as in the proof of Theorem 2.12, connections are also uncolored, chain edges connected to the connections are red and other two chain edges are blue. Moreover, we connect another blue edges e_1, e_2 to both connections in the shackles. Vertices connected by the edges e_1, e_2 are called blue goal vertices and vertices in the chain connected by red edges to the connections are called red goal vertices. To the goal vertices we will connect copies of the graph H' by the gadget vertices. The *occurrence gadget* for a variable occurrence is depicted in Figure 2.6. The goal vertices near to 4-cycle (or 6-cycle) are called 4-goal or (6-goal) vertices. We call two occurrence gadgets for the j-th occurrence of a variable z as original and inverse occurrence gadget and denote them by $S_j(z)$ and $S'_j(z)$. Our goal is to force Painter to color drawn edges in original and inverse occurrence gadget by different colors.

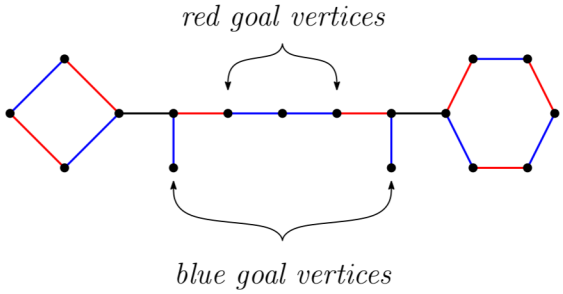


Figure 2.6: For every occurrence of each variable in φ we create 2 copies of this gadget.

We again connect all occurrence gadgets created for a variable pair. However, it is a little bit more complicated because every vertex must have the degree at most 3. Since every variable has at most 3 occurrences there are at most 12 occurrence gadgets created for a variable pair. Let T be a rooted binary tree such that T has 12 leafs and all leafs are in the depth 4. To avoid monochromatic long path in the precolored graph we colored edges of T alternatively by layers. Thus, root incidence edges are blue, edges in distance 1 from the root are red and so on. We attach one occurrence gadget by its shackles center to every leaf of T. Note that edges in T incident with leafs are red and edges incident with shackles centers in the occurrence gadgets are blue. For better understanding how occurrence gadgets are connected into a variable gadget see Figure 2.7

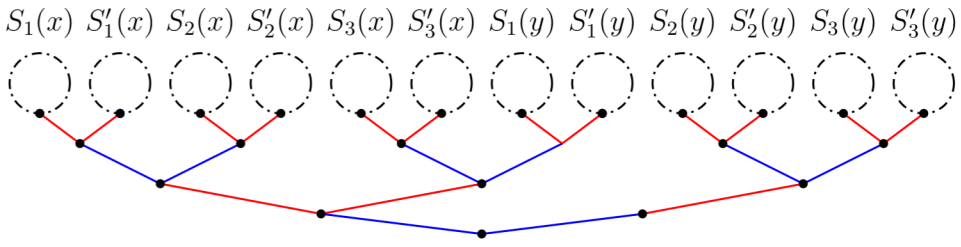


Figure 2.7: For simplicity occurrence gadgets are depicted only as dash dotted circles. They are connected by centers of shackles to the tree.

To finish the *variable gadget* $R(x, y)$ we attach copies of the graph H' to the goal vertices. We want to force Painter to color drawn edges in $S_i(z)$ and $S'_i(z)$

for any variable z by a different color. We fix a cyclic order O of the occurrence gadgets for a variable pair (x, y): $S_1(x), S'_1(x), \ldots, S'_3(x), S_1(y), \ldots, S'_3(y)$. The same order is in Figure 2.7. Let R_1, R_2 be two consecutive occurrence gadgets[2] in the order O. Suppose R_1 is an original occurrence gadget and R_2 is an inverse occurrence gadget—i.e. $R_1 = S_i(z)$ and $R_2 = S'_i(z)$ for some $i \in [3]$ and $z \in \{x, y\}$. We attach a blue copy of H' by the gadget vertices to the blue 4-goal vertices of the occurrence gadgets R_1 and R_2 and another blue copy of H' to the blue 6-goal vertices of gadgets R_1 and R_2. If R_1 is an inverse occurrence gadget and R_2 is an original occurrence gadget the attaching of the copies of H' is analogous. However, red copies of H' are attached to the red goal vertices in R_1 and R_2. We connect all pairs of consecutive gadgets with copies of H' in the same way as R_1 and R_2. A sketch of the connecting occurrence gadgets by copies of the graph H' is depicted in Figure 2.8.

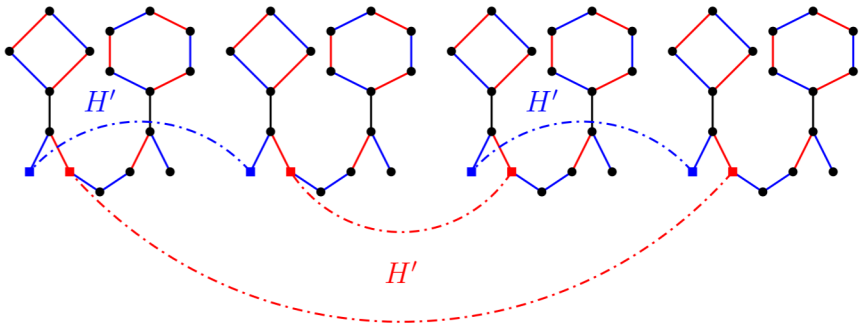


Figure 2.8: For clearness, connecting of the occurrence gadgets by the copies of H' are depicted only for four occurrence gadgets. The copies of H' are depicted as dash dotted arcs. Only connecting of 4-goal vertices is depicted, 6-goal vertices are connected by the copies of H' in the same way. Copies of H' are connected to the occurrence gadgets by identifying the gadget vertices and the goal vertices (marked vertices as squares).

We create a variable gadget for every variable pair from Z. To finish the background graph $G(\varphi, Z)$ we need to create clause gadgets. We create a path Q of length $2m - 2$. Every odd vertex on the path Q represents a clause. The other $m - 1$ vertices on Q are there only to create the background graph $G(\varphi, Z)$ bipartite. Let $v(C)$ be a vertex on Q representing a clause C. We need to connect $v(C)$ to the occurrence gadgets of variables in C. However, if we attach the paths from the occurrence gadgets directly to the vertex $v(C)$ then the degree of $v(C)$ can be bigger than 3. Therefore, we attach by an edge a rooted binary tree $T(C)$ with 3 leafs in the depth 2 to the vertex $v(C)$. Let p, r, q be variables of the clause C. We denote the 3 leafs of $T(C)$ as $\ell(p), \ell(q)$ and $\ell(r)$. If some clause has less then 3 variables then some leafs of the corresponding tree remain unused. The path Q with attached copy of $T(C)$ for every clause C of the formula φ creates the *clause gadget*.

We create red and blue copy of the clause gadget. We connect clause gadgets and variable gadgets in a similar way as in the proof of Theorem 2.12. However, it is more complicated for better understanding how the clause gadgets look like

[2]Note that O is cyclic, thus R_1 can be $S'_3(y)$ and R_2 can be $S_1(x)$.

and how they are connected to the first player variable occurrence gadgets see Figure 2.9 and how they are connected to the second player variable occurrence gadgets see Figure 2.10. All connections between clause gadgets and occurrence gadgets are made by red and blue paths of length 7. Let $\ell(x, L, c)$ be the leaf $\ell(x)$ from the copy of $T(L)$ of the color c (b for blue and r for red). Let x be a first player variable and $x \in_i C$. We connect $\ell(x, C, b)$ to the blue locks in the 4-cycle of $S_i(x)$ and $S'_i(x)$ by blue paths. We connect $\ell(x, C, r)$ to the red locks in the 4-cycle of $S_i(x)$ and $S'_i(x)$ by red paths. If $\neg x \in_j C$ we connect $\ell(x)$ to 6-cycles in occurrence gadgets. We connect $\ell(x, C, b)$ to the blue locks in the 6-cycle of $S_j(x)$ and $S'_j(x)$ by blue paths. We connect $\ell(x, C, r)$ to the red locks in the 6-cycle of $S_j(x)$ and $S'_j(x)$ by red paths.

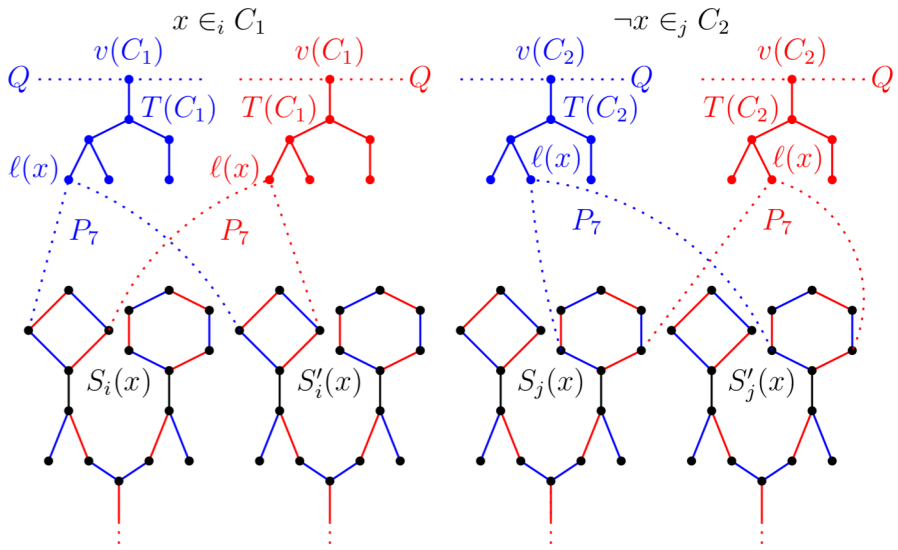


Figure 2.9: Example how a clause gadget and a variable gadget are connected for the first player variable. The paths P_7 connecting variable and clause gadgets are depicted as dotted lines and arcs.

Let y be a second player variable and $y \in_i C$. We connect $\ell(y, C, b)$ to the blue locks of the 4-cycle and the 6-cycle in $S_i(y)$ by blue paths. We connect $\ell(y, C, r)$ to the red locks of the 4-cycle and the 6-cycle in $S'_i(y)$ by red paths. If $\neg y \in_j C$ the colors switch. We connect $\ell(y, C, r)$ to the red locks of the 4-cycle and the 6-cycle in $S_j(y)$ by red paths. We connect $\ell(y, C, b)$ to the blue locks of the 4-cycle and the 6-cycle in $S'_j(y)$ by blue paths. This finished the construction of the background graph $G(\varphi, Z)$.

Observation 2.13. *The graph $G = G(\varphi, Z)$ is polynomial in the size of φ and Z. Moreover, the graph G is bipartite and its maximum degree is 3.*

Proof. Let n be a number of occurrences of all variables in φ and m be a number of clauses of φ. Every variable gadget contains constant number of copies of the graph H'. The size of H' is in $\mathcal{O}(m)$. Other parts of a variable gadget have constant size. Therefore, every variable gadget has size in $\mathcal{O}(m)$ and there are $|Z|$ of such gadgets. The clause gadget has size in $\mathcal{O}(m)$ as well. Therefore, the size of the graph G is polynomial in the size of the formula φ and the size of the set Z.

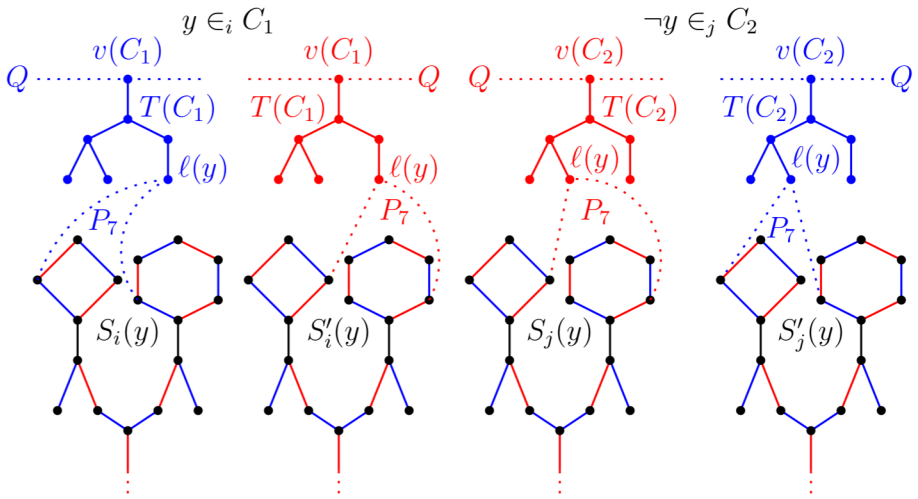


Figure 2.10: Example how a clause gadget and a variable gadget are connected for the second player variable. The paths P_7 connecting variable and clause gadgets are depicted as dotted lines and arcs.

It is clear the maximal degree of the graph G is 3. Every constructed gadget is bipartite and they are connected in such way it does not violate the bipartiteness. See Figure 2.11 how the graph G can be partitioned.

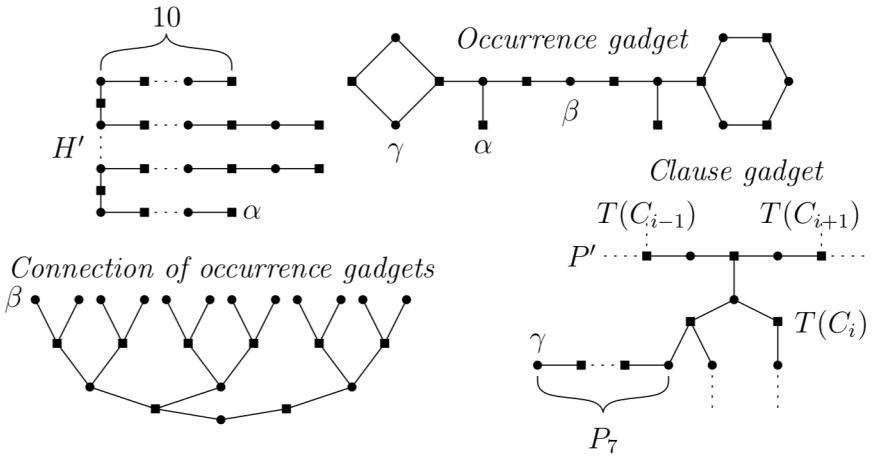


Figure 2.11: Example how vertices of the background graph $G(\varphi, Z)$ can be divided into two partitions. The vertices of one partition are depicted as disks and the vertices of the other partition are depicted as squares. How individual gadgets are connected together is indicated by Greek letters.

□

The equivalence between the evaluation of variables in the formula game and the drawing and coloring edges in the Ramsey game is similar to the equivalence in the previous reduction. Let (x, y) be a variable pair. The variable x is set to 1 if and only if the 4-connections are drawn in the variable gadget $R(x, y)$. The variable y is set to 1 if and only if the connections from the original occurrence

gadgets in $R(x,y)$ are colored by blue and the connections from the inverse occurrence gadgets in $R(x,y)$ are colored by red. That Painter will color the edges in original and inverse occurrence gadgets differently is treated by the copies of graph H' in variable gadgets—we prove it later. However, we need a rule to force Builder to draw only connections of one parameter.

Rule B Builder has to draw all edges such that the drawn graph does not contain shackles with the following parameters:

$$(6,4,6),(8,4,6),(10,4,6),(12,4,6),(14,4,6)$$

Rule B is similar to Rule 1 from the previous section. We obtain analogous observations for Rule B.

Observation 2.14. *Valid Builder's move can be checked in polynomial time in the size of the graph $G(\varphi, Z)$.*

Observation 2.15. *Let $R(x,y)$ be a variable gadgets in the graph $G(\varphi, Z)$. Let e be the first Builder's move in $R(x,y)$. According to Rule B:*

1. *Builder can draw e as a 4- or 6-connection.*
2. *Builder can draw all other connection in $R(x,y)$ of the parameter of the connection e.*

Lemma 2.16. *Let $G(\varphi, Z)$ be a background graph for the online Ramsey game with the goal graph $H(\varphi, Z)$ and Builder is restricted by Rule B. Let $R(x,y)$ be a variable gadget in the graph $G(\varphi, Z)$ such that all connections of one parameter have been already drawn. If Painter have colored some drawn connections in original and inverse occurrence gadget in $R(x,y)$ by the same color then Builder wins.*

Proof. Let $S_1, \ldots, S_{12}$ is the cyclic order of the occurrence gadgets in $R(x,y)$ in which they are connected by the copies of H'. Without loss of generality $S_1, S_3, \ldots, S_{11}$ are the original occurrence gadgets and $S_2, S_4, \ldots, S_{12}$ are the inverse ones. Let e_i be a drawn connection in S_i. Suppose there are two connections e_i, e_j colored by the same color such that e_i is in the original occurrence gadget S_i and e_j is in the inverse occurrence gadget S_j. Since i is odd and j is even, the number of the connections $e_{i+1}, \ldots, e_{j-1}$ between e_i and e_j is also even (it can be 0). Therefore, there must exists a pair e_k, e_{k+1} among $e_i, \ldots, e_j$ such that e_k and e_{k+1} have the same color, without loss of generality blue. If k is odd then S_k is an original occurrence gadget and there is a blue copy of H' between S_k and S_{k+1}. Therefore, a blue copy of H arises and Builder wins.

If k is even we find another pair $e_\ell, e_{\ell+1}$ such that both connections have the same color. The pair $e_\ell, e_{\ell+1}$ surely exists because there is an even number of the connections $e_{k+2}, \ldots, e_{k-1}$ between e_{k+1} and e_k (in the cyclic order). Let the pair $e_\ell, e_{\ell+1}$ be the closest possible pair to the pair e_k, e_{k+1}—it also includes the case $\ell = k+1$. Thus, connections $e_{k+1}, \ldots, e_\ell$ are colored alternatively. If ℓ is even then e_ℓ and $e_{\ell+1}$ have to be red. However, S_ℓ is an inverse occurrence gadget and it is connected by a red copy of H' with $S_{\ell+1}$. Thus, a red copy of H arises. If ℓ is odd then e_ℓ and $e_{\ell+1}$ have to be blue and a blue copy of H arises. □

2.3.2 Hardness of Online Ramsey Game

In this section we prove Online Ramsey Game is PSPACE-complete even for the restricted background graph and the goal graph.

Theorem 2.17. Online Ramsey Game *is PSPACE-complete even with the following restrictions:*

1. *The background graph G is bipartite and its maximum degree is 3.*
2. *The goal graph H is a tree.*
3. *Painter has no rule.*
4. *Valid Builder's move can be checked in polynomial time.*

Proof. We discuss that Online Ramsey Game is in PSPACE at the beginning of this chapter. It remains to prove the hardness. We prove it again by reduction from Paired Formula Game. Let formula φ and variable pairs $Z = \{(x_1, y_1), \ldots, (x_n, y_n)\}$ be an input for Paired Formula Game. We construct the background graph $G(\varphi, Z)$ and the goal graph $H(\varphi, Z)$ as it is described above. The only rule of the game is Rule B. By Observation 2.13 and Observation 2.14 the graph $G(\varphi, Z)$ and $H(\varphi, Z)$ with Rule B is a suitable input for Online Ramsey Game to make a reduction.

We claim the first player wins the formula game on φ with the variable pairs Z if and only if Builder wins the Ramsey game on the background graph $G(\varphi, Z)$ with the goal graph $H(\varphi, Z)$ and Rule B.

Recall that for every clause C of φ there are two copies of the vertex $v(C)$ in a red and a blue copy of the path Q. We denote the copy of $v(C)$ in the red copy of Q by $v^r(C)$ and in the blue copy of Q by $v^b(C)$. Let $\mathbb{P}^b(C)$ (or $\mathbb{P}^r(C)$) be a set of blue (or red) paths of length 11 connecting $v^b(C)$ (or $v^r(C)$) and connection type edges in the variable gadgets. Note that the paths in $\mathbb{P}^b(C)$ (or $\mathbb{P}^r(C)$) are not edge disjoint they have common edges from a copy of the tree $T(C)$. Thus, the Builder goal is to extend at least one path in sets $\mathbb{P}^b(C)$ (or $\mathbb{P}^r(C)$) for every clause C by an blue (or red) edge. On the other hand, the Painter goal is to play in such a way there exists clause C_1, C_2 such that any path in $\mathbb{P}^b(C_1)$ and $\mathbb{P}^r(C_2)$ is not extended by an edge of the same color.

In fact, if the first player has a winning strategy in the formula game then Builder can extend at least one path from the sets $\mathbb{P}^b(C)$ and $\mathbb{P}^r(C)$ by an edge of an appropriate color. On the other hand, if the second player has a winning strategy for the formula game then Painter can color the drawn edges in such a way there exists a clause C such that any path in $\mathbb{P}^b(C) \cup \mathbb{P}^r(C)$ is not extended by an edge of an appropriate color. We denote by $P^d_j(x, i)$ a path in $\mathbb{P}^d(C)$ connecting $v^d(C)$ and j-connection in $S_i(x)$ and by $Q^d_j(x, i)$ a path in $\mathbb{P}^d(C)$ connecting $v^d(C)$ and j-connection in $S'_i(x)$. For example, $Q^r_4(x, i) \in \mathbb{P}^r(C)$ is the red path connecting $v^r(C)$ and the 4-connection in $S'_i(x)$, if such a path exists.

Suppose the first player has a winning strategy S for the formula game. We construct a winning strategy S' for Builder. It is described above how to interpret Builder's and Painter's moves into the moves of the first and the second player. Strategy S' has again 2 phases. In the first phase Builder follows the strategy S. Let the first player pick a variable pair (x, y) in a round i. Builder draws

an edge in $S_1(x)$ in the gadget $R(x, y)$. If the first player evaluates x by 1 then Builder draws the 4-connection in $S_1(x)$. Otherwise, if $x = 0$ Builder draws the 6-connection. When the first phase ends there is exactly one drawn edge in every variable gadget.

In the second phase Builder just draws all connection edges in a way to not violate the rule. Thus, if there is a drawn 4-connection in the variable gadget $R(x, y)$ from the first phase then Builder draws all 4-connections in $R(x, y)$ and vice versa with the 6-connections. We can suppose Painter colors edges in the original and the inverse occurrence gadgets by different colors by Lemma 2.16. Therefore, moves of both players in the second phase are fully determined by the moves in the first phase. All Builder's moves are valid by Observation 2.15.

Let C be a clause of φ. Since the strategy S is winning for the first player, there must be a true literal ℓ in C. Suppose ℓ is the first player variable x and $x \in_j C$. Thus, the 4-connections are drawn in the gadget $R(x, y)$. If the 4-connection in $S_j(x)$ is blue then the 4-connection in $S'_j(x)$ is red and paths $P_4^b(x, j) \in \mathbb{P}^b(C)$ and $Q_4^r(x, j) \in \mathbb{P}^r(C)$ are extended. If the 4-connection in $S_j(x)$ is red then $Q_4^b(x, j) \in \mathbb{P}^b(C)$ and $P_4^r(x, j) \in \mathbb{P}^r(C)$ are extended.

If $\neg x \in_j C$ then paths from $v^b(C)$ and $v^r(C)$ are extended by the 6-connections from $S_j(x)$ and $S'_j(x)$. This case is similar to the previous one. Therefore, if C is true due to the first player variable then there are a blue path P_{12} connected to the vertex $v^b(C)$ and a red path P_{12} connected to the vertex $v^r(C)$.

Suppose ℓ is the second player variable y and $y \in_j C$. Thus, drawn edges in a gadget $R(x, y)$ are blue in the original occurrence gadgets in $R(x, y)$ and red in the inverse ones. If the drawn edges in $R(x, y)$ are the 4-connections than $P_4^b(y, j) \in \mathbb{P}^b(C)$ and $Q_4^r(y, j) \in \mathbb{P}^r(C)$ are extended. On the other hand, if the drawn edges in $R(x, y)$ are the 6-connections than $P_6^b(y, j) \in \mathbb{P}^b(C)$ and $Q_6^r(y, j) \in \mathbb{P}^r(C)$ are extended. If $\neg y \in_j C$ the case is similar only colors are switched. Therefore, if the clause C is true due to the second player variable then there is a path P_{12} of the appropriate color connected to $v^b(C)$ and $v^r(C)$ as well.

There is a path P_{12} of the appropriate color connected to the vertices $v^b(C)$ and $v^r(C)$ for every clause C of φ. These paths together with red and blue copies of the path Q create red and blue copy of the goal graph $H(\varphi, Z)$. Thus, Builder wins the Ramsey game.

Suppose the second player has a winning strategy S in the formula game. Painter uses the strategy S to win the Ramsey game. Suppose Builder draws an edge e in $R(x, y)$. If some edge is already drawn in $R(x, y)$ Painter colors e in such way to not lose immediately. Thus, he colors the edges in the original and the inverse occurrence gadgets by different colors. If the edge e is the first drawn edge in $R(x, y)$ then Painter use S to determine the color of e. If y is set to 1 according to S then Painter colors the edge e by blue if it is in the original occurrence gadget and by red if it is in the inverse occurrence gadget. Otherwise, if $y = 0$ then Painter colors the edge e by red if it is in the original occurrence gadget and by blue if it is in the inverse occurrence gadget.

Since the strategy S is winning for the second player, there is a false clause C in φ at the end of the formula game. Thus, every literal in C is false. Suppose $x \in_j C$ and x is the first player variable. Thus, only the 6-connections are drawn in the variable gadget $R(x, y)$. However, there are no path in $\mathbb{P}^b(C)$ (or $\mathbb{P}^r(C)$) connecting $v^b(C)$ (or $v^r(C)$) and the 6-connections in $R(x, y)$. If $\neg x \in_j C$ then

only the 4-connections are drawn in $R(x, y)$ and no path in $\mathbb{P}^b(C)$ (or $\mathbb{P}^r(C)$) connects $v^b(C)$ (or $v^r(C)$) and the 4-connections in $R(x, y)$. Therefore, there cannot be a path P_{12} of the appropriate color attached to $v^b(C)$ or $v^r(C)$ due to the occurrence of the first player variable in C.

Suppose $y \in_j C$ and y is the second player variable. Thus, drawn edges are red in the original occurrence gadgets in $R(x, y)$ and blue in the inverse ones. However, there is no path in $\mathbb{P}^r(C)$ connecting $v^r(C)$ and the connection type edges in $S_j(y)$ and no path in $\mathbb{P}^b(C)$ connecting $v^b(C)$ and the connection type edges in $S'_j(y)$. The case $\neg y \in_j C$ is similar, only colors are switched. Therefore, there is no path P_{12} of the appropriate color attached to $v^b(C)$ or $v^r(C)$.

As we saw there cannot be a monochromatic copy of H containing the whole copy of the path Q. There are a lot of monochromatic copies of the graph H'. Note that at the beginning of the game there are attached only blue edges to the red copies of H' in the variable gadgets to connect all occurrence gadgets. There are no attached edges to the blue copies of H' besides the edges on the goal vertices. Painter colored edges in the original and the inverse occurrence gadgets by different colors. Therefore, there cannot be a monochromatic copy of H containing the whole main path of the copy of H' connecting two shackles in variable gadgets.

However, what if a monochromatic copy of H arises somewhere else? Suppose there is a monochromatic copy H^m of the graph H in the background graph G at the end of the game. The goal graph H contains paths of length 12. Any such long paths, which can be used in H^m, are only in the copies of H' in the variable gadgets and as the connecting path between the variable and the clause gadgets. We know H^m have to use some edges of a copy Q^m of the path Q and some edges of a copy H_1^m of H' in a variable gadget. Thus, the main path of H^m has to use edges of the path between Q^m and H_1^m. Every second vertex on the main path of H has the degree 3. However, Q^m is connected with H_m^1 by a long path where almost all vertices have the degree 2—see Figure 2.12.

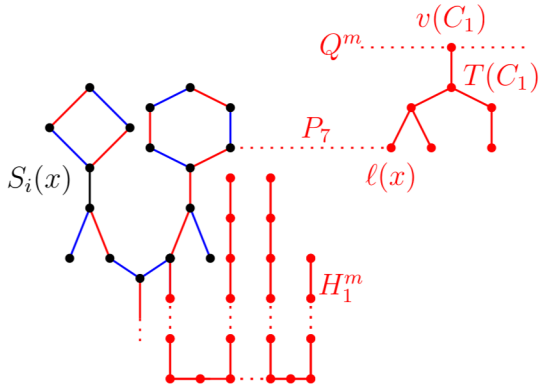


Figure 2.12: Sketch that the background graph does not contain any unwanted subgraph in the case when the 6-connection in $S_i(x)$ is colored red. Note that Q^m and H_1^m are connected by a long path where almost all vertices have the degree 2.

Therefore, a monochromatic copy of the goal graph H cannot arise on a set of edges using some edges of the copies of H' and some edges of the copies of Q. Hence, the second player wins the Ramsey game. □

Note that Theorem 2.17 is optimal in some sense. If the background graph G has the maximal degree 2 then G is a disjoint union of cycles and paths. Therefore, Builder can force a monochromatic path of length 2 if and only if he can build an odd cycle. Otherwise, he can force only a monochromatic edge.

Rules for Builder are also important. If Builder has no rules he can eventually draw all edges of the background graph. Therefore, he can win if and only if every edge 2-coloring of the background graph G contains a monochromatic copy of the goal graph H. However, this is problem of the classic Ramsey theory, which lies in Π_2 class of the polynomial hierarchy[3] and some version of the problem is Π_2-complete [16]. Therefore, it is unlikely Online Ramsey Game would be PSPACE-complete when Builder has no rules.

2.4 Star Ramsey Game

Even Online Ramsey Game is PSPACE-complete for quite simple graphs we found a type of the problem which is NP-complete.

PROBLEM:	Star Online Ramsey Game
Input:	Background graph G, $s, t \in \mathbb{N}$.
Rules:	Builder can build only vertex disjoint union of stars.
Question:	Can builder force s copies of t-stars of the same color?

We call the parameter t of a t-star S the *diameter* of S. The key result about Star Online Ramsey Game describes the following theorem.

Theorem 2.18. *Builder wins* Star Online Ramsey Game *if and only if the background graph G contains $2s-1$ vertex disjoint $(2t-1)$-stars.*

Proof. Let G be a graph containing $2s-1$ vertex disjoint $(2t-1)$-stars. Builder eventually draws edges of all $(2t-1)$-stars. There are at least t edges of the same color in every colored $(2t-1)$-star. These t-tuples of edges form monochromatic t-stars. There are $2s-1$ vertex disjoint monochromatic t-stars. Therefore, there are at least s vertex disjoint t-stars of the same color.

Suppose $G = (V, E)$ contains at most $2s-2$ vertex disjoint $(2t-1)$-stars. Due to the rules every component of the drawn graph has to be a star. We say a colored star S is maximal if there is no colored edge in $E \setminus E(S)$ connected to S.

Painter colors edges of small stars alternatively (up to the diameter $2t-2$). Thus, every maximal $(2t-2)$-star has $t-1$ blue edges and $t-1$ red edges. In the moment when a drawn edge e is connected to some maximal $(2t-2)$-star S Painter is forced to draw a monochromatic t-star. However, he can decide if the t-star will be blue or red. He counts the number of blue and red t-stars he has already colored. If there are more red t-stars then he colors the edge e by blue, otherwise red. If the drawn edge e is connected to a maximal k-star S where $k > 2t-2$ then S has to contains a monochromatic t-star S' as a subgraph. In

[3] If you are not familiar with the polynomial hierarchy see some book about complexity, for example Arora and Barak [1]

that case Painter colors the edge e by the color of the star S'. Therefore, any maximal k-star for $k > 2t-2$ contains only red or blue t-star as a subgraph but not both of them. Note that the numbers of red and blue t-stars differ by at most 1 in any moment of the game. Since the graph G contains at most $2s-2$ vertex disjoint $(2t-1)$-stars Builder can force at most $s-1$ vertex disjoint t-stars of the same color. □

By Theorem 2.18 we obtain Star Online Ramsey Game is as hard as the following problem.

PROBLEM:	Star Subgraphs
Input:	Graph G, $s, t \in \mathbb{N}$
Question:	Does the graph G contain s vertex disjoint t-stars as subgraphs?

Theorem 2.19. Star Subgraphs *is* NP*-complete.*

Proof. It is clear that Star Subgraph is in NP. We can use s vertex disjoint t-stars as a certificate. To show the hardness of the problem we use a reduction from Independent Set.

Let graph $G = (V, E)$ and $k \in \mathbb{N}$ be an input for Independent Set. Let d be a maximum degree of G. We can assume $d > 2$ because Independent Set for graphs with the maximum degree 2 is actually in P. We construct a graph G' as follows. For every $v \in V$ we create a d-star S_v. We label leafs of every such star S_v by neighbors of the vertex v. If v has less then d neighbors then some leafs of S_v remain unlabeled. For every $\{u, v\} \in E$ we identify the leaf of S_u labeled by v with the leaf of S_v labeled by u. Thus, $\{u, v\} \notin E$ if and only if stars S_u and S_v are vertex disjoint.

Note that only vertices in $V(G')$ of degree bigger then two are centers of stars S_v. Thus, for every t-star S for $t > 2$ which is a subgraph of G' there must exist $v \in V$ such that S is a subgraph of S_v. Therefore, G has an independent set of the size k if and only if G' contains k vertex disjoint d-stars. □

As a corollary of Theorem 2.18 and Theorem 2.19 we obtain the following theorem.

Theorem 2.20. Star Online Ramsey Game *is* NP*-complete.*

3. Builder's Strategies

The online Ramsey theory study mainly strategies for Builder and Painter. In this chapter we study Builder's strategies for the games played on an infinite complete graph. However, Builder can draw graphs only from a fixed class, usually planar graphs. In our strategies we often use known strategies for forcing several copies of some graph of the same color. We say copies of a graph are *strongly monochromatic* if all these copies have the same color.

We construct our strategies by induction. In the induction step we use a step operation. In the step operation we apply a known strategy on vertices which are already in a monochromatic copies of some graph. Suppose we have a strategy S for forcing a graph G. Let k be a number of vertices needed by the strategy S to force the monochromatic copy of G. Let H, I be graphs such that $V(H) \cap V(I) = \{w\}$. Suppose we obtain a Builder's strategy Q to force a graph pair (H, I) where both graphs are monochromatic. However, H and I does not have to be colored by the same color. By *step operation* we mean using the strategy Q at most $2k-1$ to force k disjoint copies $C_1, \ldots, C_k$ of pair (H, I) such that all copies of H are strongly monochromatic and all copies of I are strongly monochromatic as well. Let w_i be a copy of w in C_i. On the vertices $w_1, \ldots, w_k$ we use the strategy S to force a monochromatic copy of G. We use also a step operation where I is empty graph. In that case the vertex w is a fixed vertex in $V(H)$. After the step operation there is a monochromatic copy of G such that strongly monochromatic copies of H and strongly monochromatic copies of I are attached to every vertex of G by the copies of the vertex w. Specific use of the step operation is presented in proofs in this chapter.

In this chapter we also study properties of drawn graphs. Let Q be a Builder's strategy. A drawn graph which Builder draws using the strategy Q is denoted by $D(Q)$. If the strategy Q is used n times on the different sets of vertices the drawn graph is denoted by $D(Q)^n$.

Grytczuk et al. [8] and Petříčková [12] conjectured the graph K_4 is avoidable on planar graphs. We suspect the class of series-parallel graph is exactly the unavoidable class on planar graphs. We have not found the proof of this proposition. However, we present some techniques and results which might be useful for proving the conjecture. Our strategies were inspired by the result of Grytczuk et al. [8] that forests are unavoidable on forests. Thus, in Section 3.1 we present a slight improvement of the strategy for forests for better understanding the other strategies. Strategies on planar graphs are presented in Section 3.2. We also study strategies for hypergraphs. Our result about hyperforests is in Section 3.3.

3.1 Optimality of Strategies

Result of Grytczuk et al. [8] that forests are unavoidable on forests is interesting because an analogous result does not hold in the classic Ramsey theory. However, we found their strategy is not optimal. In this section we describe an easy improvement of the strategy for forests by Grytczuk et al. [8] which decreases the number of vertices needed by Builder. Usually to describe optimality of a strategy the number of the drawn edges is used because it is also the number

of the Ramsey game rounds. However, we use the number of vertices used by Builder because it is easier for computation and if $T = (V, E)$ is a forest without single vertices then $|V| \in \mathcal{O}(|E|)$.

Let T_n be a tree on n vertices and T' be a tree obtained from T_n by removing one leaf. The original strategy for forcing needs n strongly monochromatic copies of T' to force T_n. In the worst case Builder has to force $(2n-1)$ monochromatic copies of T' to force n copies of T' of the same color. Let G_n be a maximum number of vertices needed by Builder for forcing T_n. Thus, we obtain the following recurrence formula for G_n:

$$\begin{aligned} G_2 &= 2 \\ G_n &= (2n-1)G_{n-1} \end{aligned}$$

The solution for $n > 2$ of the recurrence is as follows:

$$\begin{aligned} G_n &= 2 \cdot \prod_{i=3}^{n}(2i-1) = 2 \cdot 5 \cdot 7 \cdots (2n-1) \\ 3 \cdot G_n \cdot 2 \cdot 4 \cdots (2n-2) &= 2(2n-1)! \\ G_n &= \frac{(2n-1)!}{3(n-1)!} \end{aligned}$$

Proposition 3.1. *Forests are unavoidable in the online Ramsey game on forests. Moreover, Builder needs at most $n!$ vertices where n is the number of vertices of the goal forest.*

Proof. Let $T = (V, E)$ be the goal graph. We can assume the goal graph is a tree. If the goal graph T is a forest we can connect the components of T by edges to obtain a tree. We prove the proposition by induction on $n = |V|$. If $n = 2$ Builder needs only one edge, i.e. two vertices, to force T and the proposition holds.

Suppose $n > 2$. Let T' be a tree obtained from T by removing a leaf ℓ. Let u be the only neighbor of ℓ. Builder has a strategy for forcing T' by induction hypothesis. In the first phase Builder forces some copies of T'. Now it comes our improvement. In the original strategy Builder forces n strongly monochromatic copies of T'. Let $C = \{v_1, \ldots, v_k\}$ be a minimal vertex cover of T. In our strategy Builder forces only k monochromatic copies $T_1, \ldots, T_k$ of T'. Let u_i be a copy of u in T_i. If there are two copies T_i, T_j of different colors Builder connects u_i and u_j. The color of $\{u_i, u_j\}$ does not matter, a monochromatic copy of T arises.

In the other case all copies $T_1, \ldots, T_k$ have the same color, say blue. Let $V \setminus C = \{v_{k+1}, v_n\}$. For every $v_i \in V \setminus C$ we take a vertex u_i of the background graph such that u_i is not in a drawn graph after the first phase. In the second phase Builder draws an edge $\{u_i, u_j\}$ if and only if $\{v_i, v_j\} \in E$. Since C is a vertex cover of T, there is attached a blue copy of T' to at least on vertex of each drawn edges in the second phase. Therefore, all edges drawn in the second phase have to be red, otherwise Painter loses immediately. All edges drawn in the second phase create a red copy of T, therefore Builder wins. An example how the strategy works for binary tree of the depth 2 is depicted in Figure 3.1.

Let I_n be a number of vertices needed by Builder when he uses the improved strategy for forcing a tree T_n on n vertices. Since vertex cover of T_n has at most

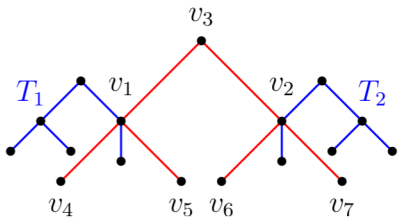


Figure 3.1: Example of the improved strategy on binary tree with a vertex cover $C = \{v_1, v_2\}$.

$\lfloor \frac{n}{2} \rfloor$ vertices, in the first phase of the n-th step Builder needs at most $\lfloor \frac{n}{2} \rfloor I_{n-1}$ vertices. In the second phase of the n-th step Builder needs at most n vertices. Therefore, the sequence I_n is upper bounded by J_n with the following recurrence formula:

$$J_2 = 2$$
$$J_n = \left\lfloor \frac{n}{2} \right\rfloor J_{n-1} + n$$

It can be easily proved by induction that $J_n \leq n!$. □

The bound of the number of vertices needed by Builder in the proof of Proposition 3.1 is quite inaccurate. However, $n! \in o(G_n)$. Thus, our improvement really decreases the number of vertices needed by Builder.

3.2 Strategies on Planar Graphs

In this section we present two technical lemmas about Builder's strategies on planar graphs. We hope these lemmas would be useful for the future work on this topic. As a corollary we found a new class of graphs which is unavoidable on planar graphs. The first lemma is about reusing existing strategy in a tree-like structure.

We define a tree-like structure where each edge of a tree is replaced by some graph. Let $T = (U, F)$ be a tree and $\mathcal{G} = \{G_f | f \in F\}$ be a set of graphs. We call a function $r : U \to \mathcal{P}(V(\mathcal{G}))$ by a *tree function*[1] if r has the following properties for every $f \in F$:

1. $|r(v) \cap V(G_f)| \leq 1$
2. $r(v) \cap V(G_f) \neq \emptyset \Leftrightarrow v \in f$

Observation 3.2. *For a tree function r and an edge $f \in F$ there are exactly 2 vertices $u, v \in U$ such that $r(u) \cap V(G_f)$ and $r(v) \cap V(G_f)$ are not empty.*

Proof. The size of a set $\{v | v \in U : r(v) \cap V(G_f) \neq \emptyset\}$ is exactly the size of the edge f, which is 2. □

[1] By $V(\mathcal{G})$ we mean a disjoint union of vertices of all graphs in $\mathcal{G}$. By $\mathcal{P}$ we mean a set of all subsets.

We define an *extended tree* $T(\mathcal{G}, r)$ for a tree T as an edge disjoint union of all graphs in $\mathcal{G}$ and for all $v \in U$ we identify all vertices in $r(v)$ into the *connecting vertex* r_v.

Informally, the extended tree $T(\mathcal{G}, r)$ arises by replacing every edge $f \in F$ by a graph $G_f \in \mathcal{G}$. The tree function r determined how the graphs in $\mathcal{G}$ are connected. An example of an extended tree is in Figure 3.2. An *extended forest* $T(\mathcal{G}, r)$ is defined in the same way but the graph T can be a forest. Let $T(\mathcal{G}, r)$ be an extended tree and R be a subtree of T. For simplicity, we use notation $R(\mathcal{G}, r)$ for extended tree such that the graph family $\mathcal{G}$ and the tree function r are restricted to the edges and the vertices of R.

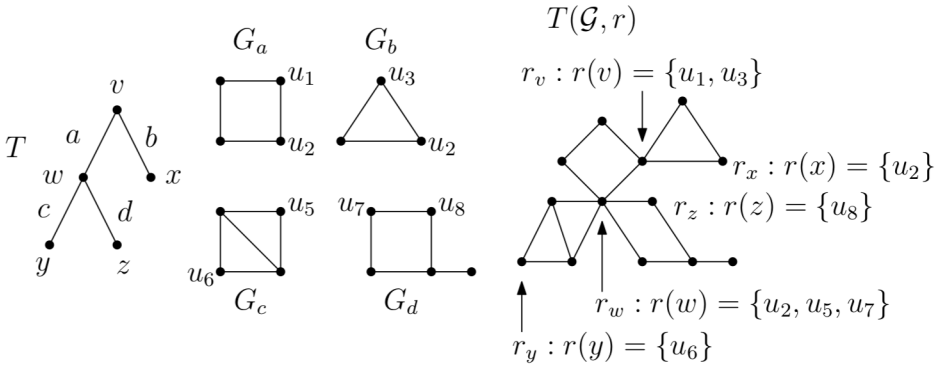


Figure 3.2: Example of an extended tree $T(\{G_a, G_b, G_c, G_d\}, r)$.

Definition 3.3. Let $G = (V, E)$ be a connected graph and $u, v \in V$. An *uniform* extended tree $T(G, u, v, r)$ is an extended tree $T(\mathcal{G}, r)$ such that every $G_i \in \mathcal{G}$ is isomorphic to G via isomorphism $f_i : V(G_i) \to V(G)$ and

$$\{f_i(v) | v \in V(T) : r(v) \cap V(G_i) \neq \emptyset\} = \{u, v\}.$$

Informally, the uniform extended tree arises by replacing every edge of a tree T by a copy of the fixed graph G such that all copies of G are connected by copies of two fixed vertices in $V(G)$. An example of an uniform extended tree is depicted in Figure 3.3.

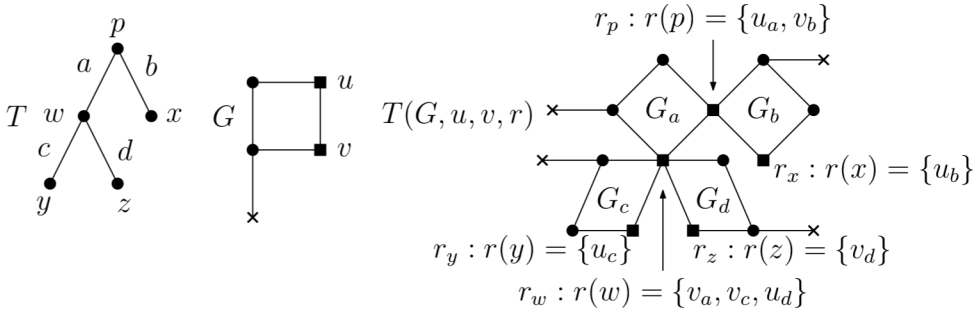


Figure 3.3: Example of an uniform extended tree $T(G, u, v, r)$.

Let $\tau_1 = T_1(\mathcal{G}_1, r_1)$ be an extended forest for a forest $T_1 = (U_1, F_1)$ and $\tau_2 = R(\mathcal{G}_2, r_2)$ be an extended forest for a forest $T_2 = (U_2, F_2)$. We say the extended forest τ_1 is a *proper subforest* of τ_2 if:

1. The forest T_1 is a subgraph of T_2
2. For every $f \in F_1$ it holds that $G^1_f \in \mathcal{G}_1$ is a subgraph of $G^2_f \in \mathcal{G}_2$.
3. For every $u \in U_1$ it holds that $r_1(u) = r_2(u)$.

Lemma 3.4. *Let $G = (V, E)$ be a graph, $u, v \in V$ and $T = (U, F)$ be a tree. Let Builder has a strategy S such that he can force a monochromatic copy of G on planar graphs. Then, there exists a Builder's strategy S' such that Builder force a monochromatic copy of an uniform extended tree $\tau_1 = T(G, u, v, r)$ such that the drawn graph $D(S')$ is a planar extended forest $\tau_2 = T(\mathcal{G}, r')$ and τ_1 is a proper subtree of τ_2. Moreover, if G is forced by S such that u and v are always in the same face of the drawn graph $D(S)$ then all copies of u and v are in the same face of a drawn graph $D(S')$.*

Proof. We prove the lemma by induction on the set of tree edges F. If $|F| = 1$ then $\tau_1 = G$ and the strategy S' is exactly the strategy S.

Suppose $|F| > 1$. Let T' be a tree obtained from T by removing one leaf ℓ. We denote the only neighbor of ℓ by z and the edge $\{\ell, z\} \in F$ by e. By induction hypothesis we obtain a Builder's strategy S_0 which forces a graph $\tau = T'(G, u, v, r)$ on a graph with the required properties. Builder uses a step operation with the strategy S_0. Let k be a number of vertices needed to force G by the strategy S. Builder forces k disjoint strongly monochromatic copies $T_1, \ldots, T_k$ of τ, without loss of generality blue.

Let z_i be a copy of the connecting vertex r_z in T_i. Builder applies the strategy S on the vertices $z_1, \ldots, z_k$ to force a monochromatic copy G_e of the graph G. There is attached a blue copy of τ to every vertex used for S in this step. Thus, if the graph G_e is blue then a blue copy of $T(G, u, v, r)$ arises and Builder wins because in particular there is blue copy of τ attached to the vertex $r_z \in V(G_e)$. An Example of the case when G_e is blue is depicted in Figure 3.4.

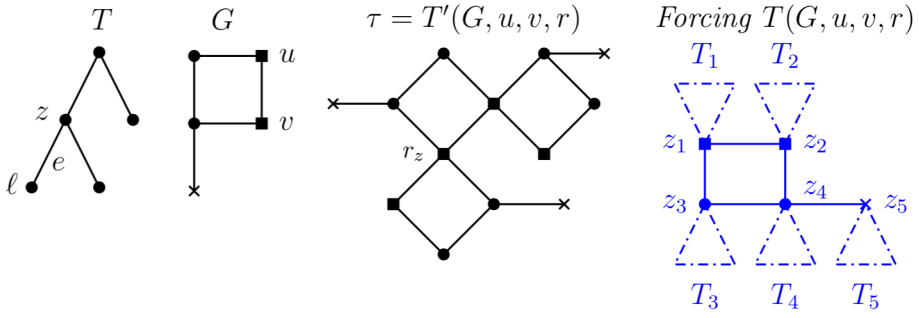


Figure 3.4: Example of how to force an extended tree $T(G, u, v, r)$. In the last graph copies of τ are depicted as dash dotted triangles.

However, the graph G_e can be red. In this case Builder can build a red copy of the graph $T(G, u, v, r)$. Let $t(H)$ be a colored graph consisting of a monochromatic copy H^m of a graph H and there is a monochromatic copy of τ of the other color than H^m attached to every vertex of H^m. We already obtain a strategy for forcing $t(G)$. Let R be a subtree of T and R' be a graph obtained from R by removing a leaf o. We denote the only neighbor of o by y and the edge $\{o, y\} \in E(R)$ by f. Let $\rho = R(G, u, v, r)$ and $\rho' = R'(G, u, v, r)$. By

induction hypothesis we have a strategy S_1 for forcing $t(\rho')$. We need to attach a monochromatic copy of G to the copy of $t(\rho')$.

Builder again uses a step operation with the strategy S_1. He forces k copies $R_1, \ldots, R_k$ of $t(\rho')$ such that all copies are colored in the same way. Without loss of generality the copies of ρ' in $R_1, \ldots, R_k$ are red and the copies of τ are blue. Let y_i be a copy of the connecting vertex r_y in R_i. Builder uses the strategy S on the vertices $y_1, \ldots, y_k$ to force a monochromatic copy G_f of the graph G. If G_f is blue then Builder wins immediately because a blue copy of $T(G, u, v, r)$ arises. If G_m is red then a copy of $t(\rho)$ arises such that copy ρ^r of ρ is red and there is a blue copy of τ attached to every vertex of ρ^r. By repeating this procedure $|F|$-times Builder eventually forces a red copy of $T(G, u, v, r)$.

Now we show the properties of the drawn graph $D(S')$. We show it by induction on steps of strategy S'. If $|F| = 1$ then S' is exactly the strategy S and the drawn graph has the required properties by assumptions.

Suppose $|F| > 1$. The strategy S' uses only the step operation. Therefore, it suffices to prove the one application of the step operation preserves the required properties. In the step operation O we have a strategy Q to force an extended tree $\sigma_1 = S(G, u, v, s)$ such that for every edge $e \in S$ the graph G_e is monochromatic. Let $\sigma_3 = S(G, u, v, s_3)$ be an extended tree containing a monochromatic copy G^m of the graph G such that there is a copy of σ_1 attached to every vertex of G^m and all these copies of σ_1 are colored in the same way. The goal of the step operation O is to create σ_3. By induction hypothesis a drawn graph $D(Q)$ is a planar extended forest $\sigma_2 = S(\mathcal{G}', s')$ and σ_1 is a proper subtree of σ_2.

If Builder uses $2k-1$ times the strategy Q on disjoint sets of the background graph vertices then a drawn graph $D(Q)^{2k-1}$ is still planar extended forest and there are $2k-1$ components of $D(Q)^{2k-1}$ which have a copy of σ_1 as a proper subtree. Let K be a set of k vertices used by Builder for the strategy S in the step operation O. Vertices in K is in different components $C_1, \ldots C_k$ of the drawn graph $D(Q)^{2k-1}$ before the strategy S is applied during the operation O. The graph $D(S)$ contains a monochromatic copy G^m of the graph G. For every i it holds that $|V(C_i) \cap V(G^m)| \leq |V(C_i) \cap V(D(S))| = 1$. Therefore, the drawn graph after the operation O is a planar extended forest and contains σ_3 as a proper subforest. The sketch of the drawn graph is in Figure 3.5.

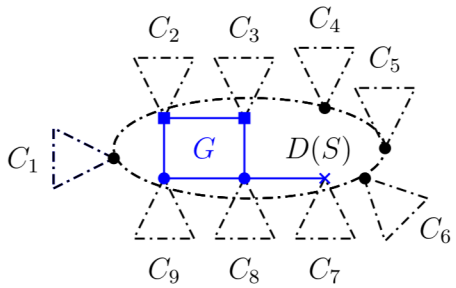


Figure 3.5: Sketch of the drawn graph after a step operation. The dash dotted black triangles represent drawn graphs of the strategy Q. The dash dotted ellipse represents a drawn graph $D(S)$. Each C_i contains a copy of σ_1 as a proper subtree.

Suppose u and v are always in the same face in the drawn graph $D(S)$. We can embed the drawn graph $D(S')$ into the plane in such a way the all copies of u

and v in $T(G, u, v, r)$ are in the outer face of $D(S')$. We construct the embedding by induction. If T is only one edge then $T(G, u, v, r) = G$ and $S' = S$ and the lemma holds. Let R be a tree obtained from T by removing a leaf ℓ. We denote the only neighbor of ℓ by x and an edge $\{\ell, x\}$ by e. A monochromatic G_e is connected by the vertex r_x to a monochromatic copy ρ^m of $R(G, u, v, r)$ to create the monochromatic graph T^m. Since the graph T^m is a proper subforest of the graph $D(S')$, the vertex r_x is a cut vertex in the whole graph $D(S')$.

Let D_1 and D_2 be parts of $D(S')$ when $D(S')$ is cut by r_x. Suppose the graph ρ^m is a subgraph of D_1. By induction hypothesis the graph D_1 can be embedded in the plane such that all copies of u and v in the graph ρ^m are in the outer face of $D(S')$. Note that the vertex r_x is among these copies of u and v. The strategy S has been used for forcing the monochromatic graph G_e. Therefore, the copies of u and v (denoted as u_e and v_e) in the graph G_e are in the same face of D_2. We can simply embed the graph D_2 (which is actually $D(S)$) such that vertices u_e and v_e are in the outer face of D_2. Actually, the vertex r_x is one of the vertices u_e and v_e. Therefore, we can join both embedded parts D_1 and D_2 by the vertex r_x and obtain the required embedding of the graph $D(S')$. □

If a drawn graph has to be planar then Builder has to draw only edges between two vertices in the same face of the drawn graph. Trivially, one edge is always monochromatic. Thus, we ask a question if there are two vertices u and v in the same face of a drawn graph, can Builder force a monochromatic path between u and v? The next lemma shows our result.

Lemma 3.5. *Let $G = (V, E)$ be a graph and $u, v \in V$. Let S is a Builder's strategy for forcing a monochromatic copy of G such that u and v are always in the same face of the drawn graph. Then, there exists a Builder's strategy S' which forces a monochromatic copy of G and a monochromatic path P_k of the odd length connecting u and v. Moreover, u and v are in the same face of a graph drawn by Builder when using the strategy S'.*

Proof. Since P_1 is only an edge, we suppose $k \geq 3$. We know forests are unavoidable on forests by Grytczuk et al. [8]. Let α be a number of vertices needed to force a path of the length $z = \frac{(k-1)^2}{2} + k - 1$ on forests by a strategy Y_α. Let β be a number of vertices needed to force a path of the length $k-1$ on forests by a strategy Y_β. Let T be a tree containing an α-star S called the major star rooted by the center c of S. There are β-stars $S_1, \ldots, S_\alpha$ called the minor stars attached by their center vertex to the leafs of S. The tree T is depicted in Figure 3.6.

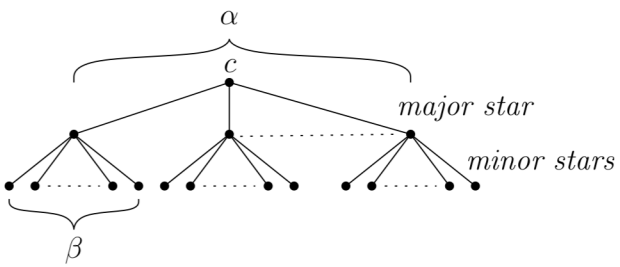


Figure 3.6: Tree T consisting of one α-star and α β-stars.

By Lemma 3.4 there exists a strategy Y for forcing graph $\tau = T(G, u, v, r)$ for any tree function r on planar extended forests. Let G_e (subgraph of τ) be a

copy of G corresponding to $e \in E(T)$. We denote the copy of u and v in G_e by u_e and v_e. We define the function r as follows:

- For the root c of T we define $r(c) = \{u_e | e \in E(S)\}$.
- For every leaf ℓ of the major star S and the minor star S' rooted in ℓ we define
$$r(\ell) = \{v_e | e = \{c, \ell\} \in E(S)\} \cup \{u_e | e \in E(S')\}.$$
- For every leaf m of the tree T we define $r(m) = \{v_e\}$ where e is the only edge in $E(T)$ incident to m.

We know all copies of u and v in $V(\tau)$ are in the same face of a drawn graph $D(Y)$. Without loss of generality, the face is outer. Let $\sigma = S(G, u, v, r)$ be a subgraph of τ. Builder forces a path P of the length z by the strategy Y_α on the copies of v in the graph σ. Let

$$\begin{aligned} V(P) &= \{p_0, \ldots, p_z\} \\ E(P) &= \big\{\{p_i, p_{i+1}\} | 0 \leq i \leq z-1\big\}. \end{aligned}$$

Let W be a set of every $k-1$-th vertex of P beside the vertices p_0 and p_z. Formally,

$$\begin{aligned} W \subset V(P) &= \{p_i | 1 \leq i < z, i \bmod k-1 = 0\} \\ &= \Big\{w_i | w_i = p_{i(k-1)}, 1 \leq i \leq \frac{k-1}{2}\Big\}. \end{aligned}$$

Let $q = \frac{k-1}{2}$, thus $|W| = q$. Let $S^1, \ldots, S^q$ be strongly monochromatic extended minor stars rooted in $w_1, \ldots, w_q \in W$. Builder forces paths $Q_1, \ldots, Q_q$ on the copies of v of every S^i by repeating the strategy Y_β. A part of τ with the monochromatic paths $P, Q_1, \ldots, Q_q$ is depicted in Figure 3.7.

Let $\mathcal{G} = \{G_e | e \in E(S), V(G_e) \cap V(P) \neq \emptyset\}$. Informally, the set $\mathcal{G}$ contains the copies G_e of G such that v_e in $V(G_e)$ is also in $V(P)$. We order the copies of G in $\mathcal{G}$ in the order of the vertices of P and embed them into the plane in that order. We do not need other copies of G in the extended major star σ beside of $\mathcal{G}$. Therefore, we can embed them into inner faces according to $D(Y_\alpha)$ to not violate the planarity. In such embedding of τ with $D(Y_\alpha)$ the copies of u and v of all extended minor stars S^i rooted in the vertices of P are still in the outer face of the drawn graph. The embedding of the extended minor stars $S^1, \ldots, S^q$ with a forest $D(Y_\beta)^q$ drawn for forcing $Q_1, \ldots, Q_q$ can be done by the similar way. Therefore, all vertices of the monochromatic paths $P, Q_1, \ldots, Q_q$ are in the outer face of a drawn graph. A sketch of the drawn graph embedding into the plane is in Figure 3.8.

We distinguish two cases when all paths $Q_1, \ldots, Q_q$ have the same color as P and when there exists a path Q_i with different color as P. Suppose all paths $P, Q_1, \ldots, Q_q$ have the same color, say blue. Let d_i be an endpoint of Q_i, it does not matter which one. Builder draws k edges

$$F = \big\{\{r_c, w_1\}, \{d_q, p_z\}\big\} \cup \big\{\{w_i, d_i\}, \{d_i, w_{i+1}\} | 1 \leq i \leq q-1\big\}.$$

If any edge in F is blue a monochromatic copy G_e of G with a blue copy of P_k connecting the vertices u_e, v_e arises. Otherwise, if all edges of F are red then the

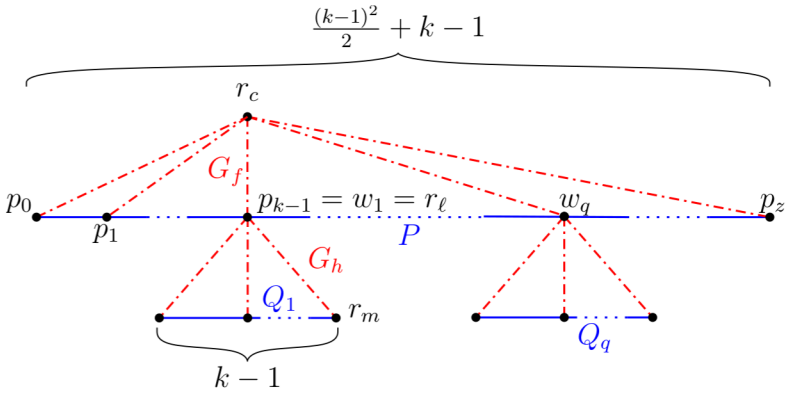


Figure 3.7: Part of τ with monochromatic paths which Builder needs for connecting the vertices u, v by a monochromatic path. The copies of G are depicted as red dash dotted lines and paths P and Q_i as blue lines. The marked vertices are copies of u and v. The vertex r_c arises by identifying the vertices u_e in $V(G_e)$ for all e in the major star S. The vertex r_ℓ arises by identifying the vertex v_f and the vertices u_e in $V(G_e)$ for all e in the minor star rooted in $\ell \in V(T)$. The vertex r_m is the vertex v_h.

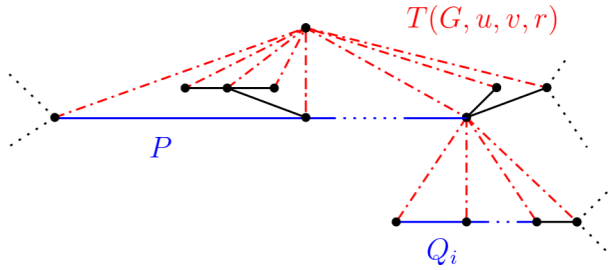


Figure 3.8: Sketch how we can embed the drawn graph when Builder forces a monochromatic τ with the monochromatic paths $P, Q_1, \ldots, Q_q$. The copies of G in τ are depicted as dash dotted red lines, the paths P and Q_i as blue lines and other edges of graphs $D(Y_\alpha)$ and $D(Y_\beta)^{q+1}$ as black lines.

copies of u and v are connected by red copy of P_k. For better understanding see Figure 3.9. Vertices of edges in F are still in the outer face of a drawn graph. A monochromatic path between copy of u and v arises always between vertices on the paths $P, Q_1, \ldots, Q_q$, no matter if all edges in F are red or one edges in F is blue. Therefore, the copies of u and v connected by a monochromatic copy of P_k remains in the outer face of a drawn graph.

Now suppose P is blue and there is a path Q_i which is red. Let c_1, c_2 be endpoints of Q_i and c_3 be a vertex of Q_i in the distance 2 from c_1. Let b_1, b_2 be vertices of P in the distance $k-1$ from w_i. Builder draws 3 edges

$$J = \big\{\{b_1, c_3\}, \{c_2, b_2\}, \{b_2, r_c\}\big\}.$$

If any edge $e \in J$ is blue then the edge e with a part of P create a blue path between the copies of u and v. If all edges in J are red then part of Q_i with edges in J create a red path between the copies of u and v. Again both endpoints of the monochromatic path P_k remains in the outer face of the drawn graph. The sketch of this case is in Figure 3.10

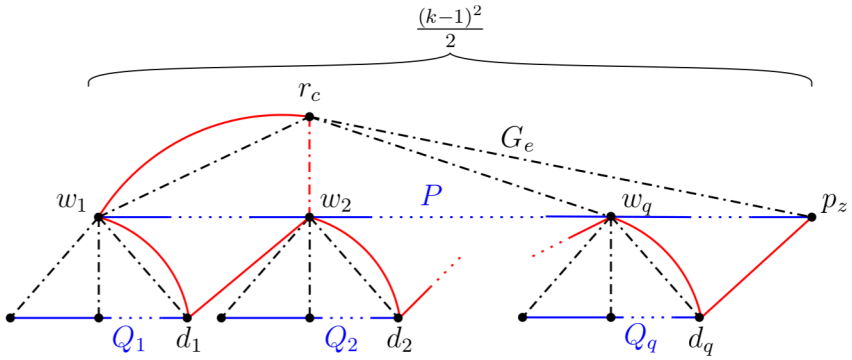


Figure 3.9: The strongly monochromatic copies of G are depicted as black dash dotted lines and the paths $P, Q_0, \ldots, Q_q$ as blue lines. The edges in F are depicted as solid red arcs and lines. If any of these edges in F is blue then a monochromatic copy G_m of G with a blue path P_k between u_m and v_m arises. If all edges in F are red then the monochromatic copy G_e of G with a red path P_k between $r_c = u_e$ and $p_z = v_e$ arises.

□

Theorem 1.9 is only a corollary of Lemma 3.5.

Corollary 3.6. *Let $Q(k, \ell, u, v)$ be a graph consisting ℓ internally disjoint paths of the length k between the vertices u and v. The graph $Q(k, \ell, u, v)$ is unavoidable on planar graphs for arbitrary $\ell \in \mathbb{N}$ and $k \in \mathbb{N}$ odd.*

Proof. Builder uses $2\ell-1$ times a strategy given by Lemma 3.5 to connect vertices u and v by a monochromatic path of the length k. At least ℓ of these paths must have the same color. □

3.3 Unavoidability of Hypertrees

Natural generalization of the online Ramsey game is to allow Builder drawing hyperedges. I.e. in each round Builder draws a set of vertices (instead of a vertex pair) and Painter colors the set by red or blue. As far as we know, the online Ramsey theory for hypegraphs has not been studied very much. Kierstead and Konjevod [10] studied Builder's strategies for complete hypergraphs. In this section we study Builder's strategy for hyptertrees. We can restrict the study only to uniform hypergraphs.

Observation 3.7. *Builder can not force any non-uniform hypergraph.*

Proof. Let $H = (V, F)$ be a non-uniform hypergraph. Therefore, there are two hyperedges $f_1, f_2 \in F$ with different sizes. If Builder draws a hyperedge of the size $|f_1|$ then Painter colors it by blue. Painter colors hyperedges of other sizes by red. Therefore, Builder can not force a monochromatic copy of H. □

By Observation 3.7 we can assume the goal hypergraph in the online Ramsey game is k-uniform. Hence, we can assume Builder draws only hyperedges of the size k because there is no use for edges of different size. We wanted to

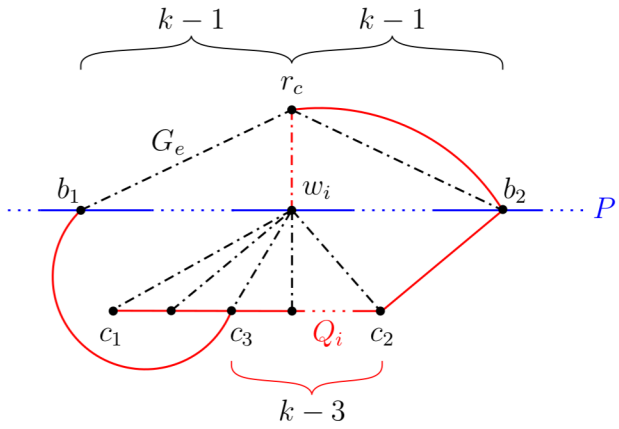


Figure 3.10: The strongly monochromatic copies of G are depicted as black dash dotted lines and the path P as a blue line and the path Q_i as a red line. The edges in J are depicted as solid red arcs and lines. If any of these edges in J is blue then a monochromatic copy G_m of G with a blue path P_k between u_m and v_m arises. If all edges in J are red then the monochromatic copy G_e of G with a red path P_k between $r_c = u_e$ and $b_1 = v_e$ arises.

generalize result of Grytczuk et al. [8] for hyperforests. In this section we present a hypergraph class which is unavoidable on 3-uniform hyperforests.

Let H be a 3-uniform hypertree and O be a host tree for H. We denote a subgraph of a O induced by a hyperedge $f \in F$ by $O(f)$. Let H' be a copy of subgraph of H and $m : V(H') \to V(H)$ be a mapping such that for all $\{u, v, w\} \in \binom{V(H')}{3}$ it holds that $\{u, v, w\} \in E(H')$ if and only if $\{m(u), m(v), m(w)\} \in E(H)$. We say a host tree O' of H' is *based* on O if for all $\{u, v\} \in \binom{V(H')}{2}$ it holds that $\{u, v\} \in E(O')$ if and only if $\{m(u), m(v)\} \in E(O)$. We used the notion also when H' is a subgraph of H, then the mapping m is the identity on the vertices of H'. First we show some properties of 1-degenerate 3-uniform hypertrees. We say a hyperedge $f \in F$ is a *leaf hyperedge* if f contains a vertex of degree 1. A leaf hyperedge f is *proper* if H without f is connected.

Proposition 3.8. *Every 1-degenerate 3-uniform hypertree H with at least two edges has at least two proper leaf hyperedges.*

Proof. Let H be a minimal counterexample and $f = \{v, u, w\}$ be a leaf hyperedge of H. Let $v \in f$ be a vertex of degree 1 and H' be a hypertree arising from H by removing f.

If f is a proper leaf hyperedge then H' is connected and has 2 proper leaf hyperedges g, i. However, at least one hyperedge from g, i is a proper leaf hyperedge also in H, which is a contradiction.

Suppose H has no proper leaf hyperedge and H' has two components C_1 and C_2 such that $u \in V(C_1)$ and $w \in V(C_2)$. Since H is a minimal counterexample, C_1 must have two proper leaf hyperedges g, i. However, one of the hyperedges g, i has to be disjoint with f and thus be proper also in H. By the same argument we find another proper hyperedge of H in C_2, which is a contradiction. □

Definition 3.9. Let H be a 1-degenerate 3-uniform hypertree and O be a host tree for H. Let S be a strategy which forces H on 3-uniform hyperforests. A host

forest O' for a drawn graph $D(S)$ *properly contains* O if a subtree of O' induced by the vertices of the monochromatic copy of H is a copy of O. The strategy S *respects* the tree O if the drawn graph $D(S)$ has a host tree O' such that O' properly contains O.

We use a structure similar to the uniform extended forest. Let $T = (U, F)$ be a forest and $\mathcal{H} = \{H_f | f \in F\}$ be a set of hypertrees such that every $H_f \in \mathcal{H}$ is isomorphic to a hypertree $H = (V, E)$. We define a *hypertree function* $g : U \to \mathcal{P}(V)^F$ with the following properties[2] for every $f \in F$ and $v \in U$:

1. $|g(v)_f \cap V(H_f)| \leq 1$
2. $g(v)_f \cap V(H_f) \neq \emptyset \Leftrightarrow v \in f$

An *extended hyperforest* $T(H, g)$ for the tree T arises as $|F|$ edge disjoint copy of H and for every $v \in U$ we identify all vertices in $g(v)$ into the connecting vertex g_v. Let O be a host tree for H. An *extended host tree* $T(O, g)$ for $T(H, g)$ arises in the same way as $T(H, g)$. Note that the definition of an extended host tree is correct because a hypertree and its host tree have the same set of vertices. It is easy to see the extended host tree $T(O, g)$ is a host tree for $T(H, g)$. First we prove a technical lemma, which is similar to Lemma 3.4.

Lemma 3.10. *Let $H = (V, F)$ be a 3-uniform hypertree and $T = (U, E)$ be a tree. If Builder has a strategy S for the hypertree H on 3-uniform hyperforests then Builder has a strategy S' for an extended hypertree $T(H, g)$ on 3-uniform hyperforests for any hypertree function g. Moreover, if S respects a host tree O for H then S' respects the extended host tree $T(O, g)$ for $T(H, g)$.*

Proof. We prove the lemma by induction on $|E|$. For $|E| = 1$ the lemma is trivial. Suppose $|E| > 1$. We again uses a step operation like in proof of Lemma 3.4, now using for hypergraphs.

Let R be a tree obtained from T by removing a leaf ℓ and $e = \{\ell, v\}$ be the only edge incident with ℓ. Builder has a strategy for forcing $\rho = R(H, g)$ by induction hypothesis. Let k be a number of vertices needed to force H by the strategy S. Builder forces k strongly monochromatic copies $R_1, \ldots, R_k$ of ρ, say blue. Let v_i be a copy of the connecting vertex g_v in T_i. Builder applies the strategy S on the vertices $v_1, \ldots, v_k$ and a monochromatic copy H^m of H arises. If H^m is blue then Builder wins because there is a blue copy of ρ attached to every vertex of H^m, in particular to the copy of g_v.

Suppose H^m is red. In this case Builder forces a red copy of $T(H, g)$. We construct the strategy again by induction. Let $t(I)$ be a colored hypergraph consisting a monochromatic copy I^m of a hypergraph I and there is a monochromatic copy of ρ of the other color then I^m attached to every vertex of I^m. We already have a strategy S_0 for forcing $t(H)$. Let $R_1 = (U_1, F_1)$ be a subtree of T and R'_1 be a tree obtained from R by removing a leaf o. Let $e = \{o, w\} \in E(R)$ be the only edge incident with o.

By induction hypothesis we obtain a strategy S_1 for forcing $t(R'_1(H, g))$ on 3-uniform hypertrees. Builder uses S_1 to force k copies $t_1, \ldots, t_k$ of $t(R'(H, g))$

[2]The function g maps a vertex $v \in U$ to a set S of subsets of V such that each hyperedge $f \in E$ has a set $V_f \subseteq V$ in S.

such that all copies are colored in the same way. Without loss of generality copies of $R'_1(H,g)$ in $t_1,\dots,t_k$ are red and copies of ρ are blue. Let w_i be a copy of the connecting vertex g_w in t_i. Builder forces a monochromatic copy H^m of H on the vertices $w_1,\dots,w_k$. If H^m is blue then Builder wins immediately because there is a blue copy of ρ attached to every w_i. Therefore, a blue copy of $T(H,g)$ would arise. If H^m is red then Builder forces a copy of $t(R(H,g))$ arises because also a red copy of $R'(H,g)$ is attached to every w_i. By this procedure Builder can eventually force a red copy of $t(T(H,g))$ and wins the game.

Builder uses only step operation. The strategy S forces the hypertree H on 3-uniform hyperforests and respects the host tree O. If the strategy S is used on some vertices of a drawn graph these vertices are always in different components $C_1,\dots,C_k$ of the drawn graph. After applying the strategy S in a step operation the components $C_1,\dots,C_k$ are connected by 3-uniform hypertree. Moreover, for every $i \in [k]$ it holds that intersection of the drawn graph $D(S)$ and the component C_i consists only one vertex. Since $C_1,\dots,C_k$ are 3-uniform hypertrees by induction hypothesis, the whole drawn graph after the step operation is a 3-uniform hyperforest as well.

Let $M_1,\dots,M_\ell$ be all components from $C_1,\dots,C_k$ such that the component M_i is incident with the monochromatic copy H^m of H forced during the step operation. Every component M_i contains a copy T_i of some extended tree $T'(H,r')$. Moreover, M_i has a host tree which properly contains $T'(O,r')$ by induction hypothesis. By operation step Builder forces a copy H^m of H with a copy of $T'(H,r')$ (somehow colored) attached to the every vertex of H^m which forms some extended hypertree $R_1(H,r_1)$. The drawn graph $D(S)$ (containing H^m) has a host tree which properly contains O. Since intersection of H^m and any component M_i consists only one vertex, the drawn graph after the step operation has a host tree which properly contains $R_1(O,r_1)$. Thus, the strategy S' respects $T(O,r)$.

□

Theorem 3.11. *The 1-degenerate 3-uniform hyperforests are unavoidable on 3-uniform hyperforests.*

Proof. Let $H=(V,F)$ be a 1-degenerate 3-uniform hyperforest. Let C_1,C_2 be components of H and $v_1 \in V(C_1), v_2 \in V(C_2)$. We can create a hypertree H' by connecting the components C_1,C_2 by a new edge $\{v_1,v_2,u\}$ such that $u \notin V$. The hypergraph H' has one component less then H and is still 1-degenerate 3-uniform hyperforest. Thus, we can connect the original hypergraph H into the connected hypergraph H'. Since H is a subgraph of H', Builder can force H' and wins. From now, we assume the goal hypergraph H is connected.

We construct a Builder's strategy S for H by induction on $|F|$. Actually, we prove a stronger proposition. Let O be a host tree for H. We construct a strategy S such that S respects the host tree O. If $|F|=1$ the strategy is trivial. Suppose $F=\{f_1,\dots,f_m\}$ for $m>1$. Let H' be a hypergraph arising from H by removing a proper leaf hyperedge f. Let O' be a host tree for H' based on O. We obtain a Builder's strategy S' for H' by induction hypothesis. Moreover, the strategy S' respects the host tree O'.

The strategy S has always two phases. The first one is forcing strongly monochromatic copies $H_1,\dots H_\ell$ of H', say blue. The second phase is a reconstruction of H on vertices of H_i. In the second phase Builder draws copies of the

edges $f_1, \ldots, f_m$. The edges are drawn in such a way if an hyperedge f' drawn in the second phase is blue then it creates with some copy H_i a blue copy of H. Otherwise, if all edges drawn in the second phase are red then it create a red copy of H.

Since f is a leaf hyperedge, the intersection of the edge f and $V(H')$ consists at most two vertices. Let $V = \{v_1, \ldots, v_n\}$. Suppose $f \cap V(H') = \{u\}$. The strategy for this case is the same as the strategy for trees by Grytczuk et al. [8]. In the first phase Builder uses the strategy S' for forcing n strongly monochromatic copies $H_1, \ldots, H_n$ of H', say blue. Let u_i be a copy of u in the blue hypertree H_i. In the second phase Builder draws hyperedges $h_1, \ldots, h_m$ which are copies of the hyperedges $f_1, \ldots, f_m$. If $f_\ell = \{v_i, v_j, v_k\}$ then Builder draws $h_\ell = \{u_i, u_j, u_k\}$. If a hyperedge h_i is blue then with some copy H_j creates a blue copy of H and the game ends. If all edges $h_1, \ldots, h_m$ are red then a red copy of H arises.

Let $M_1, \ldots, M_\ell$ be components of the drawn graph before the second phase and R_i be a host tree of the component M_i. Since an intersection of any component M_i and all the edges drawn in the second phase contains at most one vertex, it is clear the drawn graph $D(S)$ is a 3-uniform hyperforest. Suppose the hyperedges $h_1, \ldots, h_m$ form a red copy H^r of H. Let R be a host tree of H^r based on O, thus R is a copy of O. Since an intersection of any tree R_i and R contains at most one vertex, there exists a host tree for $D(S)$ which properly contains O.

Now suppose a hyperedge $h_\ell = \{u_i, u_j, u_k\}$ is blue. There are the blue copies H_i, H_j, H_k of H' attached to the vertices u_i, u_j, u_k. Thus, the copies H_i, H_j, H_k with the hyperedge h_ℓ form 3 blue copies of H. The hyperedge h_ℓ act as a copy of $f = \{u, v, w\}$ in the blue copy of H. Recall the vertex u is the only vertex in $f \cap V(H')$. Let I be a hypergraph induced by $h_1, \ldots, h_\ell$. Let B be a host tree for I based on O. Without loss of generality $\{u_i, u_j\}$ and $\{u_j, u_k\}$ are edges of B. Suppose O contains edges $\{u, v\}$ and $\{u, w\}$. Thus, we take H_j with h_ℓ as the blue copy of H and the drawn graph $D(S)$ properly contains the host tree O. If there are $\{v, u\}$ and $\{v, w\}$ (or $\{w, u\}$ and $\{w, v\}$) edges of T then we take H_i with h_ℓ as the blue copy of H. Therefore, if $|f \cap V(H')| = 1$ the strategy S respects the host tree O. A sketch how we choose the copy of H' is depicted in Figure 3.11.

Now suppose $f \cap V(H') = \{u, w\}$. We create a set which covers all hyperedges by vertex ordered pairs and each pair is an edge of O. Let $f = \{u, v, w\}$ such that $E(O(f)) = \big\{\{u, w\}, \{v, w\}\big\}$. Let $K = \big\{(x_1, y_1), \ldots, (x_m, y_m)\big\}$ be a set with the following properties:

1. For every $i \in [m]$ it holds that $f_i = \{x_i, y_i, z_i\}$ and
$$E(O(f_i)) = \big\{\{x_i, y_i\}, \{y_i, z_i\}\big\}.$$
2. For every $(x_i, y_i) \in K$ it holds there is no pair (y_i, x_i). Thus, there are no two pairs with the same vertices only in the opposite order.

The pairs in K have not to be disjoint, i.e. there can be a vertex x which is in several pairs. However, each intersection of two pairs $(x_i, y_i), (x_j, y_j)$ contains at most one vertex, thus $|\{x_i, y_i\} \cap \{x_j, y_j\}| \leq 1$ for every $i \neq j$. Therefore, the set K can be easily create by induction on $|F|$.

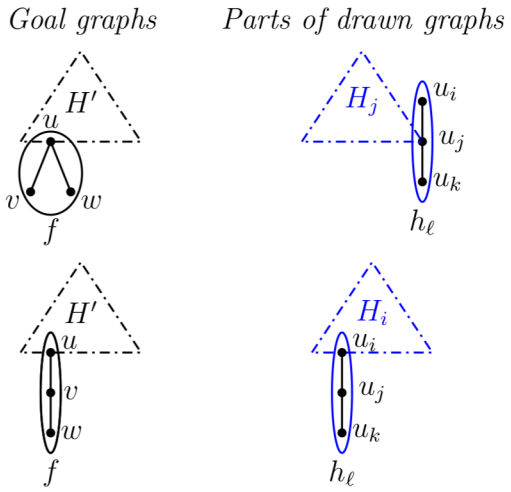


Figure 3.11: Sketch how a copy of H' is chosen for both cases of $O(f)$ that the strategy S respects the host tree O. The hypergraph H' and its copies H_i and H_j are depicted as dash dotted triangles, the hyperedges f and h_ℓ as ellipses and the edges of the host graphs as black lines among the marked vertices.

The idea of Builder's strategy is similar to the previous case. First, suppose all pairs in K are disjoint. In the first phase Builder forces m strongly monochromatic copies $H_1, \ldots, H_m$ of H', say blue. We denote the copies of u and w in the blue hypertree H_i by u_i, w_i. Let

$$Z = V(H) \setminus \bigcup_{1 \le i \le m} \{x_i, y_i\} = \{z_1, \ldots, z_\ell\}.$$

The set Z contains vertices of H which are not in any pair in K. For every $z_i \in Z$ we take a vertex z_i' of the background graph such that z_i' is not incident to any drawn hyperedge (before the second phase).

In the second phase Builder draws edges $h_1, \ldots, h_m$ as copies of the edges $f_1, \ldots, f_m$. Suppose Builder wants to draw a copy of $f_i = \{x_i, y_i, a_i\}$. If $a_i \in Z$ then $a_i = z_k$ for some k and Builder draws a hyperedge $h_i = \{u_i, w_i, z_k'\}$. Otherwise, $a_i \notin Z$ and a_i have to be in some pair in K. If $a_i = x_k$ for some k then Builder draws a hyperedge $h_i = \{u_i, w_i, u_k\}$. If $a_i = y_k$ for some k then Builder draws a hyperedge $h_i = \{u_i, w_i, w_k\}$. If h_i is blue then with H_i form a blue copy of H. Otherwise, if all edges $h_1, \ldots, h_m$ are red then a red copy of H arises.

Let M_i be a component of the drawn graph before the second phase which contains H_i. Let O_i be a host tree of H_i based on O. The component M_i is a 3-uniform hypertree and it has a host tree R_i which properly contains O_i by induction hypothesis. Thus, $\{u_i, w_i\}$ is an edge of R_i. Let I be a hypergraph induced by the edges drawn in the second phase and R be a host tree for I based on O. Suppose all hyperedges $h_1, \ldots, h_m$ are red. Hence, the hypergraph I is a monochromatic copy of H and R is a copy of O. By construction of K, for every $i \in [m]$ it holds that $\{u_i, w_i\}$ is an edge of R. Union of R and trees R_i (for all $i \in [m]$) is a host tree of a component of $D(S)$ containing a monochromatic copy of H. Therefore, the whole drawn graph $D(S)$ is a 3-uniform hypertree and the strategy S respects the tree O.

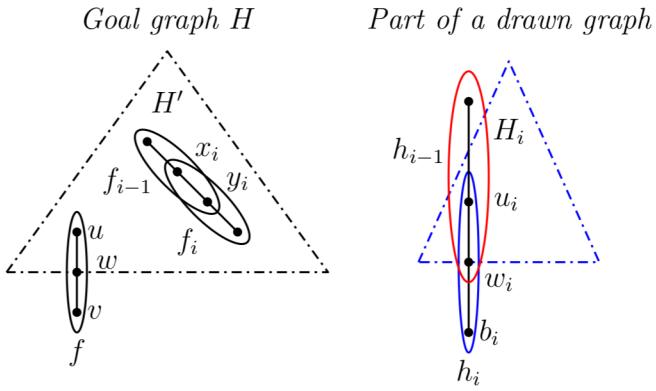


Figure 3.12: Sketch how a copy H_i of H' is connected to the hyperedge h_i by the vertex pair (u_i, w_i). The hypergraph H' and its copy H_i are depicted as dash dotted triangles, the hyperedges f and h_i as ellipses and the edges of the host graphs as black lines among the marked vertices.

Now suppose a hyperedge h_i is blue. The whole drawn graph $D(S)$ is a 3-uniform hypertree by the same argument as in the previous case. The hyperedge $h_i = \{u_i, w_i, b_i\}$ with the hypergraph H_i forms a blue copy H^b of H. The hyperedge h_i has a role of f in H^b. By construction of K, the edges of the graph $R(h_i)$ are $\{u_i, w_i\}$ and $\{w_i, b_i\}$. Recall $E(O(f)) = \big\{\{u, w\}, \{w, v\}\big\}$ and u_i is the copy of u and w_i is the copy of w_i. Therefore, the strategy S respects the host tree O.

However, the pairs in K do not have to be disjoint. Unfortunately, these hypertrees really exists—see Figure 3.13. Thus, the hypergraphs $H_1, \ldots, H_m$ do not have to be disjoint.

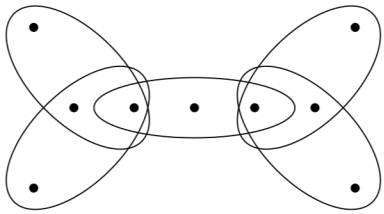


Figure 3.13: An example of a 3-uniform hypertree which can not be covered by disjoint pairs of vertices.

However, every intersection of two hypergraph H_i and H_j consists at most one vertex. To deal with this problem we use Lemma 3.10. Let J be a union of all H_i. Note that J is also 1-degenerate 3-uniform hyperforest. Moreover, for every component C of J there exists a tree T_C such that C is an extended hypertree $T_C(H', r_C)$ for some hypertree function r_C. We can join all components of J to obtain J' such that $J' = T(H', r)$ for some tree T and some hypertree function r. Since there is a Builder's strategy for forcing H', there is a strategy for forcing $T(H', r)$ by Lemma 3.10. Hence in the first phase of the strategy S, Builder forces m strongly monochromatic copies $T_1, \ldots, T_m$ of $T(H', r)$ instead of copies $H_1, \ldots, H_m$. For every i it holds that H_i is a subgraph of $T(H', r)$. Builder uses only the parts of $T(H', r)$ which he needs in the second phase. There is a host tree of $T(H', r)$ which properly contains the host tree $T(O', r)$. Therefore, the

whole drawn graph $D(S)$ is a 3-uniform hypertree and also S respects O by the analogous arguments to the previous case. □

There can be used an improvement similar to the improvement used in the proof of Theorem 3.1 in the strategies for 3-uniform hypertrees or extended trees. However, we omit it that the main ideas of the proofs stay clear.

Conclusion

In this thesis we studied two topics about the online Ramsey game. First one is the computational complexity of the decision problem Online Ramsey Game, derived from the online Ramsey game. We proved that deciding if Builder has a winning strategy on a given partly precolored background graph and a given goal graph is PSPACE-complete. Moreover, we proved the problem is PSPACE-complete even for the bipartite background graph with the maximum degree 3 and for the goal graph as a tree. We showed the problem is trivial for the background graph with the maximum degree 2. However, there are other interesting questions in this topic. For example, how hard is Online Ramsey Game when the background graph is not precolored? How hard is the problem when the background graph is planar?

Lichtenstein [11] proved the decision problem if a quantified boolean formula φ is true is PSPACE-complete even if the formula φ has a planar incident graph. The incident graph G_φ for the formula φ in CNF is a bipartite graph $(A \cup B, E)$ such that the vertices in A represent the variables of φ, the vertices in B represent the clauses of φ. There is an edge $\{x, C\} \in E(G_\varphi)$ if and only if the variable x occurs in the clause C. However, our reduction for Paired Formula Game does not preserves the planarity of the input formula incident graph. We have an idea how to treat the crossing of two path with different colors. If two paths of the background cross and they have different colors the crossing can be in some vertex of these paths—see Figure 3.14.

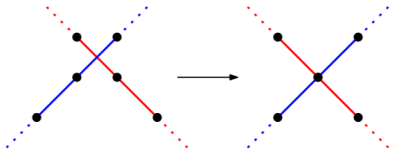


Figure 3.14: Sketch how to treated two crossing paths of different colors.

By this operation there can not arise a new unwanted copy of the goal graph, which is crucial in proving the completeness by a reduction. However, we do not know how to treat the crossing of two paths with the same color. Moreover, the maximum degree of the background graph increases to 4. Thus, another question is how hard is Online Ramsey Game with the planar background graph of the maximum degree 3.

Another topic we studied are strategies of Builder, especially when Builder has to draw only planar graphs. By previous and our work we made a conjecture.

Conjecture. *A graph G is unavoidable on planar graphs if and only if G is series-parallel.*

We did not manage to prove nor disprove the conjecture. However, we showed some results which we hope to be helpful for proving the conjecture. We developed some techniques how to reuse strategies and create new strategies. As a corollary we found a new class of graphs which is unavoidable on planar graphs. We showed 1-degenerate 3-uniform hypertrees are unavoidable on 3-uniform hypertrees. It would be interesting to generalize the result to k-uniform hypertrees.

Bibliography

[1] S. Arora and B. Barak, *Computational Complexity A Modern Approach*, Cambridge University Press, 1st ed., 2009, ch. 5.

[2] J. Beck, *Achievement games and the probabilistic method*, Combinatorics, Paul Erdős is Eighty, 1 (1993).

[3] M. Belfrage, T. Mütze, and R. Spöhel, *Online ramsey games involving trees*.

[4] J. Butterfield, T. Grauman, W. B. Kinnersley, K. G. Milans, C. Stocker, and D. B. West, *On-line ramsey theory for bounded degree graphs*, Electronic Journal of Combinatorics, 18 (2011).

[5] D. Conlon, *On-line ramsey numbers*, SIAM J. Discrete Math., 23 (2009), pp. 1954–1963.

[6] E. Friedgut, Y. Kohayakawa, V. Rödl, A. Ruciński, and P. Tetali, *Ramsey games against one-armed bandit*, Combinatorics, Probability and Computing, 12 (2003), pp. 515–545.

[7] J. Grychzuk, H. Kierstead, and P. Prałat, *On-line ramey numbers for paths and stars*, Discrete Mathematics and Theoretical Computer Science, 10 (2008).

[8] J. A. Grytczuk, M. Hałuszczak, and H. A. Kierstead, *On-line ramsey theory*, Electronic Journal of Combinatorics, 11 (2004).

[9] R. M. Karp, *Reducibility among combinatorial problems*, in Proceedings of a symposium on the Complexity of Computer Computations, held March 20-22, 1972, at the IBM Thomas J. Watson Research Center, Yorktown Heights, New York., 1972, pp. 85–103.

[10] H. A. Kierstead and K. Goran, *Coloring number and on-line ramsey theory for graphs and hypergraphs*, Combinatorica, 29 (2009), pp. 49–64.

[11] D. Lichtenstain, *Planar formulae and their uses*, SIAM Journal on Computing, 11 (1982), pp. 329–343.

[12] Š. Petříčková, *Online ramsey theory for planar graphs*, Electronic Journal of Combinatorics, 21 (2014).

[13] P. Prałat, *A note on small on-line ramsey numbers for paths and their generalization*, Australasian Journal of Combinatorics, 40 (2008), pp. 27–36.

[14] D. Rolnick, *Trees with an on-line degree ramsey number of four*, Electronic Journal of Combinatorics, 18 (2011).

[15] ——, *The on-line degree ramsey number of cycles*, Discrete Mathematics, 313 (2013), pp. 2084–2093.

[16] M. Shaefer, *Graph ramsey theory and the polynomial hierarchy*, Journal of Computer and System Sciences, 62 (2001), pp. 290–322.

[17] T. J. Shaefer, *On the complexity of some two-person perfect-information games*, Journal of Computer and System Sciences, (1978), pp. 185–225.

[18] W. Slany, *The complexity of graph ramsey games*, in Computers and Games, vol. 2063 of Lecture Notes in Computer Science, Springer Berlin Heidelberg, 2001, pp. 186–203.

[19] L. J. Stockmeyer, *Thy polynomial-time hierarchy*, Theoretical Computer Science, 3 (1976), pp. 1–22.

List of Figures

List of Abbreviations

- CNF—Conjunctive normal form
- DTM—Deterministic Turing machine
- NTM—Nondeterministic Turing machine