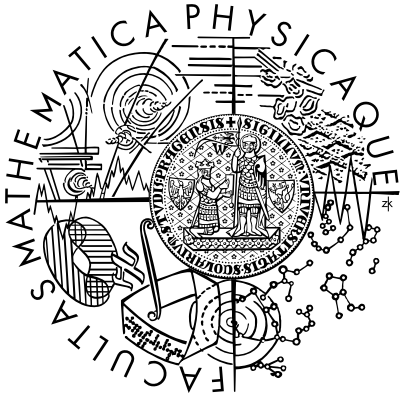
Charles University in Prague
Faculty of Mathematics and Physics

Master Thesis



Tomáš Gavenčiak

Hry na grafech ve vztahu k zdvihovým parametrům grafů

Games on graphs with respect to width parameters

Department of Applied Mathematics

Supervisor: Prof. RNDr. Jan Kratochvíl, CSc.
Study program: Computer Science
Field of study: Discrete Models and Algorithms

2009

I would like to thank my supervisor for supporting me both in my work and my studies. I enjoyed the discussions with Andrzej Proskurowski and Petr Golovach and I thank them for many helpful remarks.

I also thank Tereza Klimošová for an infinite amount of energy, love and support. *Thando!*

I hereby certify that I wrote my Master Thesis myself and using only the referenced sources. I agree with lending the thesis.

Prague, 3rd August 2009 Tomáš Gavenčiak

Název práce:	Hry na grafech ve vztahu k zdvihovým parametrům grafů
Autor:	Tomáš Gavenčiak
e-mail autora:	gavento@kam.mff.cuni.cz
Katedra:	Katedra aplikované matematiky
Vedoucí diplomové práce:	Prof. RNDr. Jan Kratochvíl, CSc.
e-mail vedoucího práce:	honza@kam.mff.cuni.cz

Abstrakt: V práci se zabýváme variantou tzv. *hry na četníky a zloděje* s nekonečně rychlým zlodějem a jejími vztahy s ostatními podobnými hrami. Srovnáváme tzv. vrtulníkovou hru charakterizující tree-width, klasickou verzi hry na četníky a zloděje a varianty s různě rychlými zloději.

U varianty s nekonečně rychlým zlodějem rozebíráme její složitost a charakterizujeme všechny grafy, kde vyhrává jeden četník. Jako hlavní výsledek ukazujeme polynomiální algoritmus rozhodující hru na intervalových grafech a zodpovídáme tak otevřenou otázku z článku Fomin a kol.: On tractability of Cops and Robbers game, *IFIP TCS 2008*, 171-185.

V důkazu polynomiality rozhodovacího problému zavádíme novou, ekvivalentní pomocnou hru na intervalové reprezentaci grafu, o jejímž rozhodování ukážeme, že je polynomiální. Ekvivalence her je dokázána technikou redukcí herních strategií.

Klíčová slova: kombinatorická teorie her, hry na četníky a zloděje, teorie grafů, tree-width, výpočetní složitost

Title:	Games on graphs with respect to width parameters
Author:	Tomáš Gavenčiak
Author's e-mail:	gavento@atrey.karlin.mff.cuni.cz
Department:	Department of Applied Mathematics
Supervisor:	Prof. RNDr. Jan Kratochvíl, CSc.
Supervisor's e-mail:	honza@kam.mff.cuni.cz

Abstract: We consider a variant of a *cop and robber game* with an infinitely fast robber and its relations to other similar games. We compare the helicopter game characterizing tree-width, the classical cop and robber game and its versions with various speeds of the robber.

We study the complexity of the infinitely fast robber variant and give an explicit characterization of all the graphs where one cop can win. As the main result, we show a polynomial time algorithm deciding the game on interval graphs. This answers a question from the paper Fomin et al.: On tractability of Cops and Robbers game, *IFIP TCS 2008*, 171-185.

To show the polynomiality of the game on interval graphs, we introduce a new auxiliary game on an interval representation of the graph and show the polynomiality of that game. Then we use game strategy reductions to show the equivalence of the two games.

Keywords: combinatorial game theory, cop and robber games, graph theory, tree-width, computational complexity

Contents

1 Introduction

Mathematical games are well-defined abstractions of real-world games or situations and allowing us to formalize and study their properties. Many of the mathematical games simulate economical processes or verify safety systems in hostile environments; these games frequently deal with randomness, secret information, and include continuous elements. Such games are generally hard to analyze exactly and the analysis of such games is often stochastic and asymptotic.

On the other hand, combinatorial games correspond to deterministic games in (usually) discrete environments, where players' decisions are the only things influencing the game and the rules and all the information about the current state are known to all the players. These games are slightly limited in their expressiveness, but many popular games still fall into this category (Chess, Go, Checkers, Nine Men's Morris, Nim, ...) and the stronger are the results that can be proved about them.

Most of the games are played on a certain board. The game board is usually fixed for the real-world games, but the mathematical abstractions often allow to use any graph (or a similar structure) as the game board.

One of the fundamental types of games on graphs are two player cop and robber games also called search games or pursuit games. These games usually simulate a fugitive (either hostile, or lost and confused) and a coordinated, systematically searching force. Cop and robber games have a wide potential in emergency and security applications and offer a common framework for all kinds of variants of the game and limitations on both players.

Another, closely related class of games is the class of graph-sweeping games. There, the cleaners move on the graph and clean up the edges they pass. The goal is to clean the graph of a contamination (for example an infection) that spreads infinitely fast and is blocked only by the cleaners themselves. The applications of these games include pursuing a group of invisible fugitives and contamination cleanup schemes.

In this thesis, we study the relations between cop and robber games, graph parameters and complexity of the corresponding decision problems. We start with the Helicopter game closely related to tree-width and compare it to the Cop and robber game in the number of required cops and the complexity of related decision problems.

Then we move on to a generalization of the Cop and robber game, namely by making the robber s times faster than the cops. There are several results concerning the complexity and properties of the generalized game. The case with $s = \infty$ gives rise to a game called the Fast robber game and we show that it lies between the Helicopter game and the Cop and robber game.

We can order the games with respect to the number of cops required to capture the robber on any fixed graph, from least cops to most:

Cop and robber game
Cop and robber game with $s \geq 2$
Fast robber game
Helicopter game

These relationships directly follow from the facts in Section 4.2 and Lemma 6.

The results for $s \geq 1$ do not naturally extend to $s = \infty$ and the complexity of the Fast robber game was an open problem. We solve this problem by showing a polynomial time algorithm deciding the problem on the class of interval graphs.

In the construction we introduce a new auxiliary Barrier game and prove its polynomiality and equivalence with the Fast robber game using the technique of strategy reductions.

The rest of the thesis is organized as follows:

Section 2 should give the reader a brief introduction to graph theory, game theory, complexity theory and the notation used in this thesis. We recommend the experienced reader to skip it and use it as a reference for less familiar topics.

The concept of the graph width parameter *tree-width* is briefly introduced in Section 3.

All the definitions of the considered games are given in **Section 4**, which also surveys the previous notable results, compares the games and gives a basic motivation.

Section 5 examines the complexity of the Fast robber game under various circumstances. Theorem 10 shows that the game is NP-complete on split-graphs (and therefore also on chordal graphs) and Theorem 12 strengthens this result to $W[2]$-hardness on split-graphs. Both these results are in fact older, but were proved independently by the author.

Section **5.3** remarks on general polynomial-time algorithms for cop&robber games with bounded number of cops. The results and proofs from Section 5 are included to stress the discovered complexity boundary, but are not used in the main proof. The main theorem about the Fast robber game on interval graphs is stated in Section **5.4**.

Section 6 introduces the Barrier game played on a representation of an interval graph and shows that it is decidable in polynomial time. The purpose of the definition is to show the equivalence between the Barrier game and the Fast robber game when played on the same graph. The two reductions are the core of **Section 7**.

A brief summary of our results with a sketch of a possibility to extend the polynomiality result to a super-class of interval graphs can be found in **Section 8**.

2 Preliminaries

This section gives an introduction to the graph theory, graph classes and intersection graphs, general combinatorial game theory and computational complexity. The author assumes a basic level of knowledge in discrete mathematics, combinatorial game theory and complexity theory.

A reader familiar with the semi-standard notation can skip most of this sections and move on to Section 3 and return later for the definitions of any unfamiliar concepts.

For more throughout introduction to graph theory, we recommend the book "Graph Theory" by Diestel [10] or the book "Modern Graph Theory" by Bollobás [6].

For an overview of the computational complexity, please see the book by Garey and Johnson [15] or by Papadimitriou [24]. As a source of information about the parametrized complexity, we recommend the book by Downey and Fellows [11].

A good introduction to the combinatorial game theory are the books "Lessons in play" [1] and "Winning Ways for your Mathematical Plays" [4]. The former is a good introduction to the theory and some algorithmic aspects and both the books provide countless examples and techniques for finding winning strategies.

2.1 Graph theory

The notion of undirected graph is central to the thesis both as a tool and a game board. Directed graphs capture the idea of the game state space and the possible moves.

An *undirected graph* G is a pair (V, E), where V is a set of *vertices* and $E \subseteq \binom{V}{2}$ is a set of unordered pairs of vertices called the *edges*. A *directed graph* or a *digraph* D is a pair (V, E), where V is a set of *vertices* and $E \subseteq V^2$ is a set of ordered pairs of vertices called the *arcs*. The arc is perceived as going from its first vertex to the second one.

Both directed and undirected edges are denoted as xy in this thesis, as is common in other literature. The order is not considered in the case of undirected edges. Some of the properties and notions are common for both graph types and then the distinction is not necessary and we use common notation where no confusion can arise.

The direction between directed and undirected graphs is usually clear from the context, but the graphs in this thesis are generally undirected unless indicated otherwise.

If $G = (V, E)$ is a graph, $V_G = V$ denotes its vertex set and $E_G = E$ denotes its edge set. The *order* of a graph G is the number of its vertices. In this thesis we deal only with finite graphs and therefore the graph order is always finite. It is also

common to use n as the order of a graph in the formulas and m is commonly used as the number of edges.

The the two vertices u and v are the *endpoints* of the edge uv. The vertices u and v are *adjacent* in G if uv is an edge of G. The *neighborhood* $N_G(v)$ of a vertex v in a graph G is the set of all the vertices adjacent to v. The number of the neighbors of a vertex v is called the *degree* of v and is denoted as $\deg_G(v)$. The *closed neighborhood* is the neighborhood of v including the vertex v and is denoted by $N_G[v]$.

We drop the subscript indicating the graph in question from the symbols if that is clear from the context.

Two graphs G and H are *isomorphic* if there is a bijection $\varphi : V_G \to V_H$ such that $uv \in E_G \Leftrightarrow \varphi(u)\varphi(v) \in E_H$.

In this thesis we consider all isomorphic graphs to be equal. This removes the information about vertex labels but simplifies the notation. We also consider all the graph classes to be closed under isomorphism.

A *disjoint union* of two graphs G and H with disjoint vertex sets is the graph with $V = V_G \cup V_H$ and $E = E_G \cup E_H$. Note that it is always possible to relabel the vertices of one of the graphs under isomorphism.

Graph H is a *subgraph* of graph G if $V_H \subseteq V_G$ and $E_H \subseteq E_G$. The subgraph is *induced* if H contains all possible edges, i.e. $E_H = E_G \cap \binom{V_H}{2}$ and similarly for digraphs.

In case v is a vertex of G, the graph $G - v$ denotes the graph G without vertex v and all the edges incident with in v. In case e is an edge of G, the graph $G - e$ denotes the graph G without that edge, but still containing both of its endpoints. The inverse operation is the addition of an edge and a vertex, denoted by $G + e$ and $G + v$ respectively. The same notation for both operations could be ambiguous in some cases, but the meaning is always clear from the context.

There are countless simple graph constructions of which we introduce and use only few basic ones.

- A *clique* or a *complete graph* on n vertices, also denoted K_n, is the graph with all the possible edges, i.e. $V_{K_n} = \{1, 2, \dots n\}$ and $E_{K_n} = \binom{V_{K_n}}{2}$.
- A *path* P_n of length n is a graph on $n + 1$ (distinct) vertices $v_0, v_1, \dots, v_n$ and with edges $v_i v_{i+1}$ for $0 \leq i < n$. The vertices v_0 and v_n are the *endpoints* of the path.
- A *cycle* C_n of length $n \leq 3$ is a path P_{n-1} with added edge $v_n v_0$.

A clique, path, or a cycle may also refer to a subgraph contained in a graph.

A graph is *acyclic* or a *forest* if it contains no cycle as a subgraph. A graph is *connected* if for every two vertices $u, v \in V$, there is a path ending in u and v. A connected acyclic graph is also called a *tree* and any disjoint union of trees is called a *forest*. An inclusion-maximal connected subgraph of graph G is a *connected component* of G.

A graph is *vertex k-connected* (or just k-connected) if it remains connected after a removal of any at most $k - 1$ vertices. Note that connectivity is equivalent to

1-connectivity. We also define *k-connected components* of G to be inclusion-maximal k-connected subgraphs.

The length of a shortest path (if there is any) between vertices u and v of an undirected graph G is the *distance* of u and v, denoted by $\text{dist}_G(u, v)$. The distance defines a metric on every undirected graph. The maximal distance among pairs of vertices of G is called the *diameter* of G.

Two well-studied vertex subset properties are independence and domination. A vertex subset $I \subseteq V$ is *independent* if there are no edges among the vertices of I. A vertex subset $D \subseteq V$ *dominates* a vertex subset $X \subseteq V$ if every vertex $v \in X$ has a neighbor in D. A set dominating the entire graph is called a *dominating set*.

2.2 Graph classes and properties

This thesis deals with some specific graph classes – namely chordal graphs, interval graphs and split-graphs – which we introduce in this section. Most of these graph classes can be defined both as intersection graphs and by their graph-theoretic description.

2.2.1 Chordal and split graphs

Graph G is *chordal* if every cycle C of length at least four has a *chord*, that is an edge between two non-consecutive vertices in cycle C. Equivalently, G has no cycle of length at least four as an induced subgraph.

Every trees is a chordal graph, but chordal graphs can be also very dense. In either case, all chordal graphs have a tree-like structure. This follows from the following inductive construction generating the entire class of chordal graphs:

Start with a single vertex. In every step choose a (possibly empty) clique in the current graph, add a new vertex and connect it to all vertices of the clique.

It is easy to see that this construction does not create long induced cycles. See Diestel's book [10] for details and more properties of the class.

A *split graph* G is a graph with its vertex set partitioned into two disjoint sets I and K. Any combination of edges between I and K is allowed, but the vertices of K must form a clique and the vertices of I must form an independent set.

Every split graph is chordal and has diameter at most 3. Although split graphs have very simple structure, they represent set systems and therefore some problems such as DOMINATING SET remain **NP**-hard even when restricted to split graphs. We discuss these properties in Section 5.

2.2.2 Intersection graphs

A wide family of graph classes arises from the *intersection graph* construction. Given a class of objects $\mathcal{C}$ and a binary "intersection" relation $\mathcal{R}$ over $\mathcal{C} \times \mathcal{C}$, any set of objects $C \subset \mathcal{C}$ gives rise to the *intersection graph* $(C, \mathcal{R}|_C)$ of C. This graph has the

elements of C as its vertices and there is an edge between every two elements of C in the relation $\mathcal{R}$. The set C is called a *model* of the resulting graph.

Examples of intersection graph classes are the intersection graphs of segments in a plane, balls or unit balls in $\mathbb{R}^k$, convex subsets of a plane or connected subgraphs of a given graph. For more information, see a book by McKee and McMorris [22] or an older classic by Golombic [19].

An example of an interval graph and its interval model is on Figure 1.

Every chordal graph is an intersection graphs of subtrees of some tree. A representation can be easily obtained from the inductive construction above.

Every split-graph is an intersection graph of subtrees of an arbitrarily large (or infinite) star graph. The subtrees containing the center c of the star correspond to the vertices of K and the subtrees not containing c (in fact only single vertices) correspond to the vertices of I.

The class of *interval graphs* is the class of intersection graphs of intervals in $\mathbb{R}$, or equivalently of subtrees of an infinite path. This is, again, a sub-class of the class of chordal graphs. There is also a pure graph-theoretic definition of the class, but it is of interest for this thesis and we direct an interested reader to the book by McKee and McMorris [22].

An interval graph can be recognized and its model can be constructed in linear time [8]. Also note that a representation of a graph on n vertices uses only $2n$ points of $\mathbb{R}$ and that it can be transformed into representation by intervals on a path of length $2n$ (and this is also the representation given by the recognition algorithm). In this thesis, we freely switch between the interval representations on a path of length $2n$ and on $\mathbb{R}$, assuming the former for effectiveness of the algorithms and the latter for simplicity.

For convenience, we refer to the lower numbers as the *left* side and to the higher numbers as the *right* side or $\mathbb{R}$. Visualizing the representation as intervals of line (with $-\infty$ on the left and $+\infty$ on the right side) should help understand the proofs and will be used in all the pictures. In some situations, it is convenient to include $-\infty$ and $+\infty$ to the domain and allow infinite intervals. This extension is only for convenience and does not increase the complexity of the arising graphs.

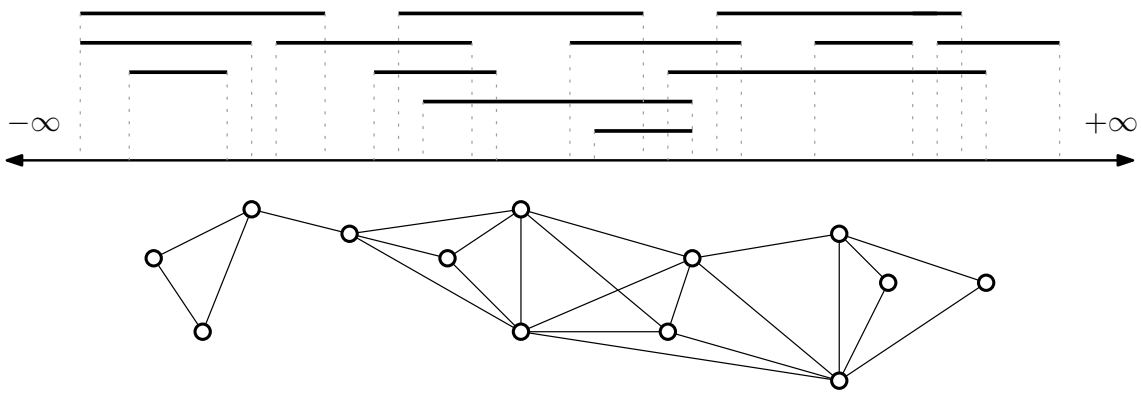


Figure 1: A family of intervals and the corresponding intersection graph.

2.3 Game theory

We use the following definition of a *combinatorial game:*

The game is played by two players. The *game state* is always fully determined and known to both players. Each state is assigned to one player — that player decides the next move[1]. The rules describe all the possible states of the game and allowed moves. The available moves depend only on the game state; never on chance nor game history[2]. Each player has a set of *winning states* — the player wins whenever the game reaches one of these states.

Some sources require the game to be finite, however, we may declare an infinite game either a *draw* or winning for one of the players. If we allow draws, there may be also some states marked as final draw states.

A *strategy* for one of the players is a dictionary, advising the next move for each situation with that player's turn. A strategy is *winning* (or *non-losing*) if a player using that strategy wins (wins or draws) against any strategy.

The *length* of a strategy $\mathcal{S}$ is the number of moves in the longest possible game played using $\mathcal{S}$ and is denoted $\text{len}(\mathcal{S})$.

An important theorem about strategies for combinatorial games states that for any game, one of the players has a non-losing strategy (for games with draw or infinite games) or a winning strategy (for finite games without draw). A very general lattice-theoretic proof of this theorem can be found in a paper by Banaschewski and Pultr [3]. Note that it is possible to extend the decision algorithm below to an elementary algorithmic proof of the theorem for games with a finite number of states.

Deciding a game is the problem of deciding the existence of a winning strategy for a given player. Generally, deciding many classes of combinatorial games with finite number of states is **PSPACE**-hard, as it is closely related to satisfying Boolean formulas with alternating quantifiers.

However, there is a simple *state-marking algorithm* running in time polynomial with the number of states[3]. With two players L and R, it marks the states either A-win or B-win. All the states start unmarked, only the winning states are marked as winning for the respective player.

Let S be a state with L on turn. If there is a move to a state already marked L-win, the algorithm marks S as L-win. If all the moves lead to states marked as R-win, then the algorithm marks S as R-win. The algorithm acts symmetrically on states with R on turn. The algorithm continues as long, as there are states that can be marked by the four rules.

[1] The players do not necessarily have to alternate in moving.

[2] However, the game history information may be part of the game state.

[3] The actual time complexity of the algorithm is linear in the number of possible moves.

Clearly, player L has a winning strategy for a game starting from a L-win state just following the L-win states and the same holds for the other player. If some states are left unmarked, then any player has a non-losing strategy just following the unmarked states.

We only sketch the algorithm and its proof, but an interested reader can find more information in the book "Lessons in play" [1] together with some implementation examples.

While not directly practical for most games, this method gives a polynomial algorithm for games having a polynomial size of the state space. We use this technique in Section 6.2 in the proof of polynomiality of the Barrier game since the game has $O(n^4)$ states.

One last concept we introduce is the *monotonicity* of a game. This can be defined in several ways, but usually it means that if one of the players has a winning strategy, then the player also has a winning strategy avoiding repetition of some pattern or property of the state in any game played using that strategy. This may refer to forbidding a repetition of a position of one of the players, forbidding expanding the area available for the other player, or to other concepts.

Note, that if a player has a winning strategy, then this strategy can be always made monotone in the sense that no state is repeated using that strategy.

2.4 Complexity

Complexity is a measurement of a hardness of a given decision problem. The complexity is usually classified in one of two ways, either as an asymptotic dependency of resources consumed by an algorithm solving the problem on the input size, or by assigning the problem to a *complexity class*.

We ask the inquiring reader to see the book by Papadimitriou [24] for the exact definitions and an overview of the complexity classes hierarchy as we introduce the complexity classes used in this thesis very briefly.

The first approach usually uses the well-known "big O" notation for time and memory used on a particular abstract machine model (usually RAM) and has –in some sense– a finer scale than the latter[4].

A function f is $O(g)$ if there is a constant c such that $f(x) \leq c\, g(x)$ for all x such that $f(x)$ is defined. In complexity, this captures the difference between algorithms running in a linear and an exponential time, but not between two linear ones. This saves a lot of extremely technical work counting the exact number of elementary instructions of our algorithms.

[4]Although it is possible to have a class of problems with algorithms running in time $O(f)$ for every function f.

There is an infinite hierarchy of complexity classes, but we introduce only the ones mentioned in the thesis:

- The class **P** is the class of all the problems solvable in time polynomial in the size of input on a deterministic Turing machine. The exponent of the polynomial can be arbitrary but must be fixed for every problem.
- The class **NP** is the class of all the problems solvable in a time polynomial in the size of input on a **non**-deterministic Turing machine.
- The class **EXPTIME** contains all the problems solvable in time exponential in the size of the input, that is in time $2^{poly(n)}$. The polynomial may be arbitrary, but must be fixed for the problem.
- The class **PSPACE** contains to all the problems solvable in space polynomial in the size of the input.

The *complexity class* approach groups the problems of a similar hardness by the possibility of *reducing* one problem to another one from the same class, i.e. solving it "fast" assuming we can solve the target problem "fast".

For any complexity class C, every problem such that any problem in C can be *reduced* to it, is called a *C-hard problem*. The class of *C-complete problems* consists of all the C-hard problems also contained in C.

A *reduction* is an algorithm transforming the input I_1 of the reduced problem P_1 in time polynomial in $|I_1|$ to an input I_2 of the target problem P_2 such that the answer to $P_1(I_1)$ is "YES" if and only if the answer to $P_2(I_2)$ is "YES".

Note that we have the following inclusions:

$$\mathbf{P} \subseteq \mathbf{NP} \subseteq \mathbf{PSPACE} \subseteq \mathbf{EXPTIME}.$$

The question of equalities remains open and it is an interesting task to separate **NP**-complete problems from similar problems in **P**. Different complexity results are especially interesting for closely related problems.

In the proofs of **NP**-hardness and **W[2]**-hardness in this thesis, we use reductions from the Dominating set problem, which is defined in the following way:

Dominating set
Input: A graph $G = (V, E)$ and a number $k \geq 0$.
Question: Is there a dominating set $D \subseteq V$ of G with $|D| \leq k$?

Dominating set is a well-known **NP**-complete problem. The parametrized version of Dominating set problem makes the k a parameter and is known to be **W[2]**-hard.

A generalized complexity measurement is the concept of *parametrized complexity*. The topic is introduced and well-described in the book "Parameterized Complexity" by Downey and Fellows [11]. We state only the basic definitions, as introducing the whole topic (especially the hardness part) it is out of scope of this thesis.

The concept gives us a way to split the input of a *parametrized version* of a problem in two parts – the data and the *parameter* – and analyze the complexity dependence on the data size and the parameter separately. We denote the size of the data part of the input n and the value of the parameter k. The parameter is usually a number, but it can be anything, for example a graph or a set.

Usually, parametrizing an **NP**-complete problem by the size of a certificate yields an algorithm running in time $O(n^{f(k)})$ for some f (usually a polynomial), but that does not really give us any new insight. It would be much more interesting if the exponent of n would be bounded, even if that would mean increasing the constant in the asymptotic complexity. This is captured in the definition of the (parametrized) class **FPT**, or fixed parameter tractable problems.

A problem is in **FPT** if there is an algorithm running in time $O(f(k)n^c)$ for some constant c and some (arbitrary) function f.

Famous examples of **FPT** problems include many problems parametrized by the tree-width of the graph[5] and other width parameters.

While **FPT** is the parametrized version of **P**, **W[i]**-hardness is the parametrized version of **NP**-hardness. The classes create a hierarchy of **W[i]**-hard problems for $i \geq 1$, but the most commonly used ones are **W[1]** and **W[2]**.

The classes **W[i]** are defined using reductions, but the details are quite technical and out of scope of this thesis.

[5]There is a famous result by Courcelle and Mosbah stating that every problem expressible in the monadic second-order logic (MSOL) admits a *linear* time algorithm with the multiplicative constant depending only on the tree-width of the graph and the formula [9].

3 Tree-width

The first and most famous of the many graph *width parameters* is *tree-width*, introduced by Robertson and Seymour in their Graph Minors series in 1986 [27]. Very roughly said, it indicates how much a graph resembles a tree in its structure.

There are several equivalent definitions of tree-width — a tree decomposition, a partial k-tree, an elimination sequence, brambles and a helicopter game. The helicopter game is defined and examined in Section 4.1.

A good introduction to tree-width can be found for example in the general graph-theoretic book "Graph Theory" by Diestel [10] and in the book "Treewidth, computations and approximations" by Kloks [20].

Tree-width has countless applications and many **NP**-complete problems admit a polynomial (often even linear) time algorithm on graphs of bounded tree-width. See the paper by Courcelle and Mosbah [9] for a description of a huge class of such problems.

We define tree-width in several ways and sketch the proofs of some of the equivalencies.

A *tree-decomposition* of a graph G is an auxiliary tree T with a subset T_t of V_G assigned to every node t of T satisfying the following conditions:

- For every edge xy of G, x and y appear together in some T_t.
- For every v of G, the subgraph of T induced by the set of nodes containing v is connected.

We call the vertices of T *nodes*[6] to distinguish them from the vertices of G.

The *width* of a tree decomposition is the size of the biggest node. Graph G has *tree-width at most* k if there is a tree decomposition of width at most $k+1$. The $+1$ in the definition assures that forests are exactly the graphs of tree-width 1.

See Figure 2 for an example of an optimal tree decomposition.

The second definition of tree-width uses the notion of *k-tree*, which is defined recursively:

- A complete graph K_k is a k-tree.
- If G is a k-tree and K a subgraph of G isomorphic to a K_k, then the graph G with a new vertex v adjacent only to K is a k-tree.

See Figure 3 for an example of a k-tree with its optimal decomposition tree.

[6] The vertices of T are also called *bags*.

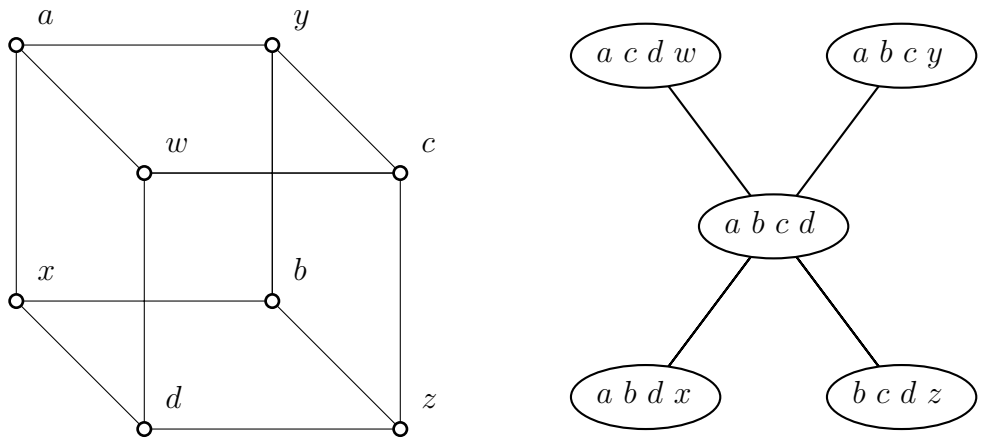


Figure 2: A tree decomposition of width 4 of the graph on the left proves that the graph has tree width at most 3.

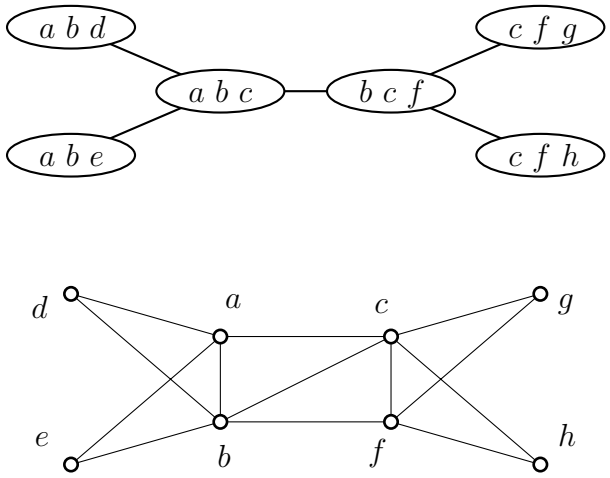


Figure 3: A 2-tree with an optimal tree decomposition of width 2. The 2-tree was constructed starting from $\{ab\}$ and the new vertices were added in alphabetic order.

A *partial k-tree* is a subgraph of some k-tree. The two definitions are equivalent by the following lemma:

Lemma 1. *Graph G has tree-width at most k if and only if it is a partial k-tree.*

The proof is constructive and straightforward. Given a tree-decomposition of width at most k, add all the possible edges (add xy whenever x and y occur in the same node) and check that this yields a k-tree.

On the other hand, it is easy to construct a tree-decomposition of width at most k for every k-tree by an inductive construction, adding one node containing K and v for every new vertex v getting connected to a clique K.

The definition by an *elimination ordering* is closely related to partial k-trees via the following lemma:

Lemma 2. *A graph G is a partial k-tree if and only if it can be reduced to a graph on at most $k+1$ vertices by the following operation:*

Take a vertex v of degree at most k and eliminate *it, that is: remove v and make all the vertices in $N(v)$ adjacent to each other.*

The proof is again quite straightforward using the correspondence between the two vertex-orderings. The reversed construction ordering of the k-tree can be used as an elimination ordering and vice versa.

While the equivalence between the definitions above is quite straightforward, the equivalence with the definition by *brambles* is more complicated. Brambles give a *dual* characterization in the sense that while all the definitions above describe a certificate that a graph has small tree-width, an existence of a certain bramble implies a high tree-width.

A *bramble* $\mathcal{B}$ is a family of subsets of V_G such that:

- Vertices of every subset $X \in \mathcal{B}$ form a connected subgraph of G.
- Every two subsets $X, Y \in \mathcal{B}$ are adjacent, that is either $X \cap Y \neq \emptyset$ or there is an edge xy in G with $x \in X$ and $y \in Y$.

The *order* of the bramble is the size of a smallest transversal of the set system, that is a set of vertices hitting every member of the bramble.

Lemma 3. *Graph G has tree-width at least k if and only if it has a bramble of order at least $k+1$.*

The complete proof of this equivalence is quite involved and out of scope of this thesis. The direction from a bramble of high order to a high tree-width follows from the properties of the Helicopter game. The Helicopter game is an alternative definition of tree-width introduced in Section 4.1.

This actually shows a duality between tree decompositions and brambles — the minimal width of a tree decomposition of a graph is equal to the maximal order of a bramble.

4 Games on graphs

Games on graphs form a very wide family of combinatorial games. There is no formal definition of a game on graph and the games may be of various kinds, but they all share some common properties. These games usually use a graph or a similar structure as the game board and the players place, remove or move tokens or avatars on the nodes of the graph.

In *cop and robber games* – also called *search games* or *pursuit games* – one player controls a number of cop (searcher) figures and second player controls one or more robber figures. The figures move in turns in a virtual map represented by a graph.

The goal of the cops is to capture all the robbers while the goal of the robber is to avoid capture (either indefinitely or by moving to a safe "haven"). There are countless variants on this concept varying in the positions, possible moves and the concept of capture. Most of the games are memory-less although some forbid the repetition of a game state.

In the following sections we describe three well-known cop and robber games, the Helicopter game, the Cop and robber game and some of its variants, and the Fast robber game. For a good survey on cop and robber graphs, see a survey by Alspach [2] and an overview by Fomin and Thilikos [14].

4.1 Helicopter game

The Helicopter game is an alternative definition of the famous graph parameter *tree-width*, defined in Section 3. There we show several equivalent definitions and the Helicopter game is yet another one, but quite different from the others and leading to different approaches in proofs. Both the tree-width and the Helicopter game were proposed by Robertson and Seymour [27].

The Helicopter game is played on the vertices of a given graph. The first player controls k cops, the second player controls one robber. The robber is visible at all times.

The cops choose arbitrary starting vertices; more cops may occupy one vertex at any time. Then the robber chooses a starting vertex. The players then alternate in moves (starting with the cops):

1. Some of (or all) the cops use helicopters to move to another vertices — these cops are removed from the graph and the destination vertex of every cop is announced to the robber.
2. The robber moves from the current position to any vertex that is reachable by a path avoiding all the cops on the graph.
3. The cops land on the announced destinations.

The cops win if any cop occupies the same vertex as the robber. The robber wins by avoiding the capture indefinitely.

As mentioned above, the game gives an alternative characterization of tree-width:

Lemma 4. *There is a winning strategy for k cops in the* Helicopter game *on G if and only if G has tree-width at most $k-1$.*

Proof. Given a tree decomposition T of width at most k, the cops start by occupying all the vertices of any of the nodes t. Then they move, one by one, to occupy the neighbor node t' in the direction of the robber. This is possible without letting the robber move to a different component of $T-t$, since the cops occupy all the vertices of $t \cap t'$ all the time. This way, robber's area only shrinks until the capture.

The other direction assumes the existence of a bramble $\mathcal{B}$ of order at least $k+1$. Then it is easy to construct a winning strategy for the robber: In every robber's turn, there is an $Y \in \mathcal{B}$ avoiding all the cops and all the announced destinations. If there was no such Y, then the cops' positions including the announced destination would form a transversal of $\mathcal{B}$, a contradiction with the order of $\mathcal{B}$.

The robber should always move to any vertex of such Y. Moving there from a (previously) cop-free subgraph X (possibly with some cops preparing to land in X) is always possible thanks to the conditions on connectivity of X and Y and the adjacency of X and Y. □

4.2 Cop and robber game and its variants

Another well-studied type of cop and robber games are so-called *pursuit games* with slow cops and a slow robber moving along edges of the graph. The original Cop and robber game was proposed in 1983 by Nowakowski and Winkler [23] with a direct motivation of formalizing the problem of finding a person lost in a cave system. The game was introduced independently in 1985 by Quilliot [25].

The Cop and robber game is played on the vertices of a given graph. The first player controls k cops, the second player controls one robber. The robber is visible at all times.

The cops choose arbitrary start vertices; more cops may occupy one vertex at any time. Then the robber chooses a starting vertex. The players then alternate in moves (starting with the cops):

1. Some of (or all) the cops move to vertices adjacent to their current locations.
2. The robber moves from the current position to any adjacent vertex.

The cops win if any cop occupies the same vertex as the robber. The robber wins by avoiding the capture indefinitely.

Various aspects of the game have been considered. The first focus was on the number of searchers, with notable results about planar graphs, Cayley graphs, bounded

genus graphs and others. Please refer to a survey by Alspach [2] for an overview of the results.

Another well-studied parameter was the maximum length of the game, called the *capture-time* of a graph. This was considered for example by Bonato et al. [7] with capture-time bounds for various classes. The matching bound on the capture-time for the one cop COP AND ROBBER GAME was found by the author in his bachelor thesis [16, 17].

Goldstein and Reingold [18] have shown that the COP AND ROBBER GAME is **EXPTIME**-complete on directed graphs (where both the cops and the robber have to obey the directions of the directed edges when moving). They also show that deciding the game in undirected graphs is **EXPTIME**-complete when the initial positions of the cops and the robber are part of the input.

There is a polynomial time algorithm for every fixed number of the cops, but it is exponential in the number of cops. Refer to Section 5.3 for a description and details of the algorithm.

There is a simple characterization of all the graphs where one cop wins the game. These graphs are usually called *cop-win* or *dismantlable*. The theorem uses the notion of a vertex u *dominating* a vertex v, this means that u is adjacent to v and every neighbor of v is also a neighbor u.

Theorem 5 (Nowakowski and Winkler [23])**.** *A graph is cop-win if and only if it can be reduced to one vertex by the following operation:*

Remove a vertex v such that there is a vertex u dominating $N[v]$.

The complexity of the game on undirected graphs (without the initial positions as a part of the input) has been surprisingly open for a quite long time, until Fomin et al. recently proved it to be at least **NP**-hard [12, 13].

A natural extension to a COP AND ROBBER GAME with the robber s-times faster than the cops is examined by Fomin et al. [12, 13]. Among other things, they have shown that the game is **NP**-hard to decide on split graphs for $s \leq 2$. This fact directly implies some of the results in Section 5.1.

Fomin et al. also show that this variant is decidable in polynomial time for a fixed number of cops on the class of interval graphs.

Increasing s and making the robber faster obviously does not decrease the number of cops required to capture the robber. The same holds for making the robber infinitely fast. This yields the infinite hierarchy of cop and robber games proposed in Section 1.

Fomin et al. [12, 13] ask a question about the complexity of the cop and robber game for $s = \infty$ (equivalent to the FAST ROBBER GAME) on interval graphs, which is answered in Section 5.4 using a very different approach from that of Fomin et al.

4.3 Fast robber game

The Fast robber game is a modification of the Helicopter game changing the way the cops move around the graph. Instead of temporarily leaving the graph with helicopters, the cops in their turn instantly and simultaneously move to the destination location which is limited to the neighbor vertices. We can also view the Fast robber game as the Cop and robber game with an infinitely fast robber.

The Fast robber game is defined as follows:

The cops choose arbitrary starting vertices; more cops may occupy one vertex at any time. Then the robber chooses a starting vertex. The players then alternate in moves (starting with the cops):

1. Some of (or all) the cops move to vertices adjacent to their current locations.
2. The robber moves from the current position to any vertex that is reachable by a path avoiding all the cops.

The cops win if any cop occupies the same vertex as the robber. The robber wins by avoiding the capture indefinitely.

The cops in this game, while limited to slow moves, are much more powerful than the cops in the Helicopter game. In fact, any winning strategy for k cops in the Helicopter game can be easily turned into a winning strategy for k cops in the Fast robber game on the same graph.

Lemma 6. *If k cops have a winning strategy in the* Helicopter game *on a connected graph G, so do k cops in the* Fast robber game *on G.*

Proof. Assuming we have a winning strategy for the Helicopter game, we just sketch the construction of a winning strategy for the Fast robber game.

The cops use the same starting position and generally follow the positions given by the strategy for the Helicopter game. Whenever the strategy for the Helicopter game commands the cops to fly to distant positions, the cops move (walk) there using shortest paths in at most n turns. While some of the cops are walking, they ignore the robber and the other cops stand still.

The resulting strategy may result in (accidentally) capturing the robber sooner, but always captures the robber in at most n-times more turns. □

This also shows that the tree-width of G plus one is the upper bound on the number of cops necessary to capture a robber in the Fast robber game on G. However, there are graphs with arbitrarily high tree-width where even one cop is sufficient to capture the robber in the Fast robber game, for example the complete graph K_n for arbitrarily high n.

Also note that if G is not connected and has many components (for example $k+1$), k cops can not win the Fast robber game on G no matter how small the tree-width of G is. The connectivity does not matter in the case of Helicopter game.

4.4 One cop Fast robber game

Here we give a simple characterization of all graphs where one cop can capture a fast robber.

Theorem 7. *One cop can win the* Fast robber game *on a graph G if and only if G can be reduced to a single vertex by repeating the following operation:*

Let C be a component of 2-connectivity of G on at least 2 vertices (C may be an edge), and let $v \in C$ be such that G is connected to the rest of G only by v (i.e. $G \setminus (C - v)$ is connected) and such that v dominates C. Then remove $C - v$ from G.

Proof. Let G denote the original graph and G' the graph after a single operation of removing $C - v$. We show that one cop has a winning strategy in G if and only if one cop has a winning strategy in G'.

Suppose the cop has a winning strategy on G'. Let the cop play on G using the same strategy except when the robber is in C; in that case play as if the robber was on v. This game will either end with the cop capturing the robber, or reach a state with the robber in C and the cop on v (by a move that would be winning in the original strategy on G'). From this state the cop can capture the robber with a single move.

A cop having a winning strategy for G can use the same strategy on G', moving to v whenever the strategy commands to move to some vertex in C. To see this properly, we can first modify the original strategy on G in the same way — letting the cop move to $C - v$ only if the robber is there. By staying on v whenever commanded to move to C, the cop threatens a superset of vertices. Finally notice that the modified strategy is directly usable on G'.

Every operation removes at least one vertex, so a finite graph requires only a finite number of operations. Note, that the operation preserves 1-connectivity of the graph.

Let G_0 be the subgraph of G that cannot be reduced by the described operation. If it is only a single vertex, the cop has a winning strategy on G_0 and therefore also on G. If it is disconnected, then G is as well and the robber can win by starting in a different component than the cop did.

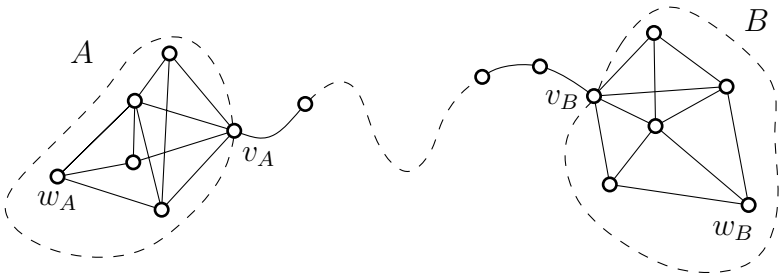


Figure 4: A situation with two components of 2-connectivity providing a pair of safe hideouts v_A and v_B for the robber.

If the graph has just one component of 2-connectivity, that component does not have a dominating vertex (otherwise it would be reducible) and the robber can avoid capture by always moving to a vertex not adjacent to the cop.

If the graph has more than one component of 2-connectivity, let A and B be two of those components, each connected to the rest of G_0 by only one vertex, v_A and v_B, respectively. There are at least two such components since the 2-connected components form a tree-like (acyclic) structure. Since neither A nor B can be reduced by the operation, there are vertices w_A and w_B in A and B not adjacent to v_A and v_B, respectively. The robber now has a winning strategy by starting in one of $w_{\{A,B\}}$ (whichever is safe) and always moving to the other one when the cop moves to the neighborhood of robber's current vertex. This is possible since the cop must be in $A\{v_A, w_A\}$ or in $B\{v_B, w_B\}$ and a free path exists for the robber.

See Figure 4 for an illustration of a G_0 winning for the robber. □

Note, that the ordering of the operations can be arbitrary and the operation includes removing leaves from the graph.

The characterization also gives a simple linear time algorithm deciding the one cop Fast robber game.

Corollary 8. *There is a linear time algorithm deciding the one cop* Fast robber game.

Proof. We give only a sketch of the algorithm: Given G, first check whether it is connected. The graph is robber-win if not. Second, compute the components of 2-connectivity. For each such component, compute and keep track of the *border* vertices — vertices having neighbors outside the component. Also, for each vertex keep track of 2-connected components it is part of (this is at most its degree).

Whenever a component of 2-connectivity C not marked as a blocker has exactly one border vertex v, check whether v dominates C. If so, remove $C - v$, update the information about v and continue. If not, mark C as a blocker.

Whenever a component of 2-connectivity C has no border vertices, check whether C has a dominating vertex. If so, then the cop has a winning strategy on G. If not, the robber has a winning strategy (just staying safe in C).

The algorithm can be implemented in time linear in the number of edges in G. We omit the detailed description and running time analysis, as both these parts are straightforward. The authors shall gladly provide further details upon request. □

Let us just note that it might be a possible to obtain a characterization of all the graphs where 2 cops can win the Fast robber game by eliminating certain components of 3-connectivity, but that would requires many technical conditions and several elimination rules. The characterization does not seem to give much insight and the resulting algorithm seems to run in quadratic time at best.

5 Fast robber game complexity

In this section we examine the complexity of the game decision problem. We show that the Fast robber game is **NP**-hard to decide on general graphs. Moreover, its parametrized version is **W[2]**-hard, so under some widely accepted complexity theory assumptions we may assume that an algorithm running in time $O(n^{O(k)})$ is close to the best possible. We mention such algorithm in Section 5.3.

Results of this section are generally older, but I proved these results independently and I include the proofs in the hope that the audience will find them interesting.

The game hardness theorems and proofs can be found in a paper by Fomin et al. [12,13]. Note that although the authors state the theorems for any bounded speed of the robber, speed 3 is sufficient in the case of split graphs.

Generally we use reductions to the Dominating set problem, which is **NP**-hard on general graphs (see a book by Garey and Johnson [15]).

To use the reduction, we alter the graph to ensure it has no small cut while preserving the size of the smallest dominating set. The following lemma allows us to sufficiently enlarge every cut of the graph.

Lemma 9. *Replacing a vertex v of a graph G by a clique K_v connecting every vertex of K_v to all vertices in $N(v)$ yields a graph G' with the same size of minimal dominating set.*

Proof. The graph G has a dominating set D of size at most k if and only if G' has a dominating set D' of size at most k.

We can get D' from D, just replace each vertex $v \in D$ by any vertex of K_v. The resulting D' dominates G'.

To get D from D', replace every vertex w of $D' \cap K_v$ with v if there is any such w. Note that in this case $|D| \leq |D'|$. □

5.1 NP-hardness on split-graphs

Note that hardness of the Fast robber game on split-graphs implies hardness on general chordal graphs. That makes the existence of polynomial algorithm for interval graphs particularly interesting.

Theorem 10. *Deciding the* Fast robber game *is **NP**-hard even when restricted to split-graphs.*

Proof. We use the fact that deciding the minimum size of a dominating set in split-graphs is **NP**-complete, as proved by Bertosi [5]. The reduction goes as follows:

The given instance of Dominating set is (G, k), where G is a split-graph with $V_G = K \cup I$, where vertices K form a clique and I is an independent set of vertices, and with $k \leq 0$.

First we must ensure that there are no cuts in our graph of size k or less. This can be done easily by successively replacing every original vertex v of K by a copy of K_{k+1} (called K_v) and connecting every vertex of K_v with $N(v)$. The new graph G' has $V_{G'} = K' \cup I$ and has no cut of size at most k while having a minimal dominating set of the same size as G by Lemma 9.

We claim that k cops can win the game on G' if and only if G has a dominating set of size at most k. If G has a dominating set D, then G' has a dominating set D' with $|D'| \leq |D|$. The cops start by occupying D' and then capture the robber in one move.

For the other direction, suppose that k cops have a strategy capturing the robber in G'. Consider a robber that in a situation with cops occupying the set C flees to any vertex not adjacent to a cop (if there is any). Since G' contains no cuts of size at most k, the robber can always move to any vertex not occupied by a cop.

Let D' be the position of the cops one turn before capture in any game against such robber. Since the robber could not flee to a vertex not adjacent to a cop, we have that D' dominates G' and $|D'| \leq k$. □

5.2 W[2]-hardness on split-graphs

The idea of simple recognition algorithms for a small fixed numbers of cops in Section 4.4 might suggest fixed parameter tractability of the game decision problem when parametrized by the number of cops k. However, this is not probably not the case since the parametrized version of the game is **W[2]**-hard even when restricted to split-graphs.

The following easy lemma is proved in the paper by Raman and Saurabh [26].

Lemma 11. *Parametrized* Dominating set *problem is* ***W[2]****-hard even when restricted to split-graphs.*

Proof. First note that Dominating set is **W[2]**-hard for general graphs. See the monograph by Downey and Fellows [11] for proof and related results. To show that it is also **W[2]**-hard for split graphs, we use the following reduction:

Given an instance (G, k) of Dominating set, create a graph G' with $V_{G'} = \{i_v, k_v \text{ for } v \in V_G\}$, two copies of G, where the vertices $\{k_v\}$ form a clique and the vertices $\{i_v\}$ form an independent set. For every $uv \in E_G$ add two edges $i_u k_v$ and $k_u i_v$ to $E_{G'}$. Graph G' is a split-graph and has the same size of the smallest dominating set as G.

To get a dominating set D for G from D' dominating for G', take all v such that $k_v \in D'$ or $i_v \in D'$.

To get a dominating set D' for G' from D dominating for G, take all k_v such that $v \in D$.

It is straightforward to verify that the constructed sets are indeed dominating and not larger than the original sets. It follows that (G', k) is an instance of Dominating set equivalent to (G, k). □

The theorem itself is an analogue of Theorem 10:

Theorem 12. FAST ROBBER GAME *parametrized by the number of cops is* ***W[2]****-hard even when restricted to split-graphs.*

Proof. Let the instance for the parametrized DOMINATING SET problem be (G, k) with G a split-graph. This problem is **W[2]**-hard by Lemma 11.

First we increase the size of the smallest cut in our graph above k by successively replacing every vertex of G by a copy of K_{k+1} as in Lemma 9. Let G' denote the new graph.

The rest of the proof is an analogue to the proof of Theorem 10: The split-graph G has a dominating set of size at most k if and only if k cops can catch the robber in graph G'. □

5.3 Universal algorithms

We feel obliged to mention an (almost) universal algorithm deciding the cop and robber games and some other games on graphs. The simple idea comes from the fact that for a majority of the memory-less games, the number of states of a game of k cops and l robbers on graph with n positions (usually vertices) is bounded by $O(n^{l+k})$ or similarly. For most "sensible" games it is also easy (polynomial) to decide the possibility of a move and decide whether a given state is a winning state. Under some additional technical conditions, the game can be then decided by the state-labeling algorithm sketched in Section 2.3.

This method gives an algorithm deciding both the HELICOPTER GAME and the FAST ROBBER GAME in time $O(n^{O(k)})$ on all graphs. The algorithms are practical for fixed small k but in general, both the HELICOPTER GAME and the FAST ROBBER GAME are **NP**-complete and therefore it is quite unlikely that there would be a polynomial-time algorithm.

5.4 Polynomiality on interval graphs

The main result of this thesis is a polynomial algorithm deciding the FAST ROBBER GAME on interval graphs. The proof is quite complicated and takes up rest of the thesis.

In order to show the polynomiality, we introduce the auxiliary BARRIER GAME and prove its polynomiality. The BARRIER GAME captures the idea that in any state of the FAST ROBBER GAME, the two closest barriers left and right of the robber held by the cops are the most important of the cops' position, while the other cops play a role mostly only in moving the two barriers. The BARRIER GAME is introduced in Section 6.

The next step is to show the equivalence of the two games. This is done in Section 7.

The main theorem can be stated as follows:

Theorem 13. *There is an algorithm that, given an interval graph G and $k \geq 0$, decides the k-cop* Fast robber game *on G in time polynomial in $|V_G|$.*

Proof. The theorem is a direct consequence of the results of Sections 6 and 7.

Lemmas 19 and 20 show that there is a winning strategy for k cops in the Fast robber game on G if and only if there is a winning strategy for k cops in the Barrier game on G.

Theorem 17 shows that k-cop Barrier game on G can be decided in polynomial time. □

The algorithm runs in time $O(|V_G|^6)$ and can output a winning strategy for the cops or for the robber for both the games.

We also get a bound on the length of both games from one of the reductions:

Corollary 14. *The length of a cops' shortest winning strategy is at most $|V_G|^2$ for the* Barrier game *and $|V_G|^3$ for the* Fast robber game.

Proof. The first part is directly Corollary 18, the second part is a combination of Corollary 18 with Lemma 20. □

6 Barrier game

In this section we define the Barrier game and prove its basic properties together with its polynomiality.

The game is defined in order to prove the polynomiality of the Fast robber game on interval graphs. The Barrier game is a two player combinatorial game played on an interval graph G and its fixed interval representation $\{I_v \subseteq \mathbb{R}\}_{v \in V_G}$. To avoid ambiguities, we assume all the endpoints of the intervals in the representation to be pairwise distinct.

The game is not a cop and robber game as such, since the states are not exactly cop and robber positions, but every state represents a set of cops' and robber's positions. We frequently refer to an "actual" position of cops on G when examining a state or a move of the Barrier game to express the relationship to the Fast robber game.

6.1 Definition

In order to introduce the Barrier game, we define the states and few concepts connecting it to the Fast robber game.

A *barrier* at point $B \in \mathbb{R} \cup \{-\infty, +\infty\}$ is the set of intervals of a representation (and therefore also vertices) containing the point B. The barriers in $-\infty$ and $+\infty$ are the empty barriers to the left and to the right of all the intervals.

Let $V(B)$ denote the vertices of the barrier at B and let $L(B)$ and $R(B)$ be all the intervals to the left and to the right of the barrier, respectively, not including the vertices in $V(B)$. Note that if both $L(B)$ and $R(B)$ are nonempty, then $B(B)$ is a (not necessarily minimal) cut between $L(B)$ and $R(B)$. If $B = -\infty$ then $L(B) = \emptyset$ and $R(B) = V$ and symmetrically for $B = +\infty$. Note that $L(B)$, $V(B)$ and $R(B)$ are always connected.

We view the cops' position to be a multiset C of vertices. The individual cops are generally indistinguishable, but sometimes we use them in a matching or for holding a barrier, and then we do distinguish between the cops sharing a vertex.

We say that the cops *threaten* a vertex set X if they can move to occupy entire X in one move. In another words, the cops in positions $C \subset V$ (a multiset) threaten X if and only if there is a perfect matching between C (again as a multiset) and X in G (with loops allowing some cops to pass).

The cops *hold* a vertex set X if there is at least one cop on every vertex of X. The cops *holding* a barrier B are some of the cops occupying $V(B)$, one per vertex of $V(B)$.

If a vertex is not threatened by a cop, we say it is *safe*. A set of vertices is *dominated* by a cop multiset C if every single vertex of the set is threatened by some cop of C. This is in fact the same domination as introduced in Section 2.1.

A *playground* is a pair (L, R) of two barriers, with $L \leq R$. The *area of the playground* $P = (L, R)$ denoted by $A(P)$ or $A(L, R)$ is the set of vertices $R(L) \cap L(R)$. $A[L, R]$ denotes $A(L, R) \cup L \cup R$.

Note that the borders of the playground may be the empty barriers outside the interval graph and that the playground may be empty even for $L \neq R$. A family of playgrounds $\mathcal{P}$ *covers* a vertex set X if $X \subseteq \bigcup_{P \in \mathcal{P}} P$

The Barrier game on an interval graph G with k cops is a game for two players (cop and robber). There are two kinds of game states:

- A *cop-state* is a pair (L, R) of barriers (a playground).
- A *robber-state* is a four-tuple of barriers (L, R', L', R) with $L \leq R' \leq R$ and $L \leq L' \leq R$.

The initial state of the barrier game is the cop-state $(-\infty, +\infty)$. All the cop-states (B, B) for B a barrier are winning states for the cops. There are no winning states for the robber — the robber wins by avoiding the states winning for the cops indefinitely.

There are three types of moves. The players do not necessarily alternate in turns as some of the cop moves lead to a cop-state and some of the robber-states may offer a losing state as one of the possibilities.

The possibility of every cop's move must be justified by the existence of a *witness* structure in the underlying graph G.

- *Cop's move type 1 (expanding the playground):* From a cop-state (L, R), the cop chooses a cop-state (L', R') such that $L' \leq L \leq R \leq R'$ and there is a vertex multiset (cops' positions) $C_{(L,R)\to(L',R')}$ containing both $V(L)$ and $V(R)$, threatening $V(R')$ and threatening $V(L')$ such that $|C_{(L,R)\to(L',R')}| \leq k$. The set $C_{(L,R)\to(L',R')}$ is the witness of the possibility of such move.
- *Cop's move type 2 (creating robber's choice):* From a cop-state (L, R), the cop chooses a robber-state (L, R', L', R) such that there is a vertex multiset (cops' positions) $C_{(L,R',L',R)}$ containing both $V(L)$ and $V(R)$, threatening $V(R')$ without moving cops on $V(L)$, threatening $V(L')$ without moving cops on $V(R)$ and dominating $A(R', L')$, such that $|C_{(L,R',L',R)}| \leq k$. The set $C_{(L,R',L',R)}$ is the witness of the possibility of such move.
  Note that the witness is just a multiset, but when appropriate, we fix the "threatening" matchings to L' and R' in the proofs below.
- *Robber's move:* From a robber-state (L, R', L', R), the robber chooses one of the playgrounds (L, R') and (L', R) as the next cop-state. Note that one or both the playgrounds may be empty (and therefore losing for the robber). You can see an example of a robber state with a witness of the previous cop's move on Figure 5.

In a game state with cops on vertices C and robber on a vertex v, let L be the rightmost barrier left of the robber's interval held by the cops and, symmetrically, let R be the leftmost barrier right of the robber held by the cops. The *playground*

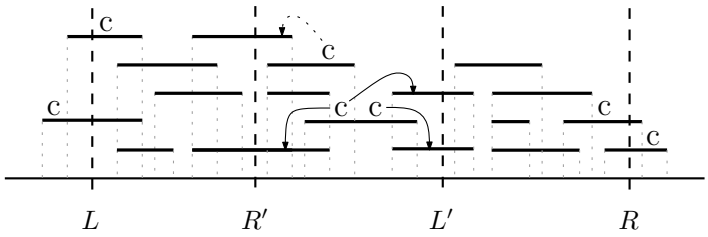


Figure 5: An example of a robber-state (L, R', L', R) with $R' \leq L'$ and a witness $C_{(L,R',L',R)}$ for the cop's move from (L, R). Also used as an example in the proof of Lemma 16.

corresponding to this game state is $P = (L, R)$ and describes the area in which the robber may freely move in his turn because $A(P) \setminus C$ is always connected.

The idea behind a robber-state and its certificate is that in the Fast robber game, the cops are positioned in such a way that they hold both the barriers L and R, threaten both the barriers R' and L' (not necessarily at once) and dominate $A(R', L')$. The robber then has to move either to playground (L, R') or (L', R) and the cops take barriers R' or L' respectively.

The details of the connection are discussed in Section 7.

6.2 Properties

Here we prove that there is a polynomial algorithm deciding the Barrier game. The algorithm has two parts: First we construct the game state digraph. The second step is to use a game state marking algorithm to decide whether the initial state (and therefore the whole game) is cop-win.

We start with the simpler move type:

Lemma 15. *The existence of a witness $C_{(L,R)\to(L',R')}$ for cop's move type 1 is decidable in polynomial time for $L' \leq L < R \leq R'$ and $L \cap R = \emptyset$.*

Proof. It is sufficient to solve the left and right sides separately since $L \cap R = \emptyset$. We compute the minimum required number of cops k_L and the corresponding cop multiset C_L for the left side; k_R and C_R for the right side is computed symmetrically.

Once we know the two numbers, the witness exists if and only if $k_L + k_R \leq k$ and then $C_{(L,R)\to(L',R')} = C_L \cup C_R$.

Let C_L be a smallest cop multiset C_L holding L, dominating $A = A(L', L)$ and threatening L' by a minimal subset of cops T. We may assume that in C_L a maximum number of vertices of L' is threatened by cops holding L. This does not require any change to C_L, just an adjustment to the matching between C_L and L'.

To get a smallest C_L, compute a maximum matching between L and L'. Then threaten all the unmatched vertices of L' by arbitrarily placed additional cops, one per vertex. Set C_L to be L and the additional cops and set $k_L = |C_L|$.

See Figure 6 for illustrations.

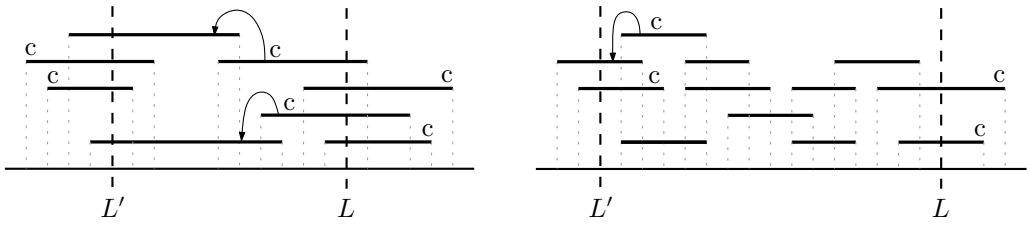


Figure 6: An example of type 1 move with L' adjacent to L **(a)** or not **(b)**. Only the left side of the playground is shown.

The assumptions we make about a smallest cop multiset guarantee that the created C_L has the minimal possible size. Note that the matching between L and L' used in the algorithm can be computed by a greedy algorithm. □

Lemma 16. *The existence of a witness $C_{(L,R',L',R)}$ for cop's move type 2 is decidable in polynomial time.*

Proof. This move type requires a bit more complicated reasoning, but the ideas are very similar to the previous proof. We distinguish two cases: $R' \leq L'$ and $L' < R'$.

I. First we consider the case $R' \leq L'$.

Assumption 1. Without loss of generality we assume that all the cops not holding L and R are in $A[R', L']$. We can do this because moving a cop $c \notin A[R', L']$ and not in $L \cup R$ to any vertex of L' (if $c > L'$) or R' (if $c < R'$) preserves the witness conditions.

Assumption 2. We use the observation from the proof of the previous lemma and assume that in any minimal witness, the maximal number of cops holding L are selected to take the (rightmost possible) vertices of L'. Symmetrically for R and R'. If that number is non-zero for either side, the $A(R', L')$ is already dominated by $L \cup R$.

Assumption 3L. We may assume that in any minimal witness, the cop assigned to threaten $l \in L'$ is either holding L or is positioned at the interval c with the leftmost left endpoint adjacent to l but still in $A(R', L')$ and therefore not entirely left of R'. That may seem complicated, but the goal is to place the cops such that they dominate the maximal possible area left of L' and threaten as much of R' as possible.

This can be done by a change to any minimal witness by moving some of the cops threatening L' to the left. This move of a cop not holding L preserves the holding of L and the threatening of l and the (possible) threatening of $r \in R'$. The domination of every vertex of $A(R', L')$ is also preserved.

Assumption 3R. The same applies to cops threatening $r \in R'$ and not holding R — they should be positioned at rightmost intervals of $A(R', L')$. Note that if in a minimal solution a cop threatens both $l \in L'$ and $r \in R'$, we may place the cop at any vertex in $N[r] \cap N[l]$ by the same reasoning.

An example of a witness with cops placed according to these rules is on Figure 5.

The algorithm constructing a smallest witness $C_{(L,R',L',R)}$ first determines a maximum matching between L and L' threatening the rightmost vertices of L', and between R and R' threatening the leftmost vertices of R'. Note that this can be done by a simple greedy algorithm.

The second step is to threaten each nonthreatened $l \in L'$ by a cop on the interval c with the leftmost left endpoint adjacent to l but still in $A(R', L')$. We have shown that we can assume this about any minimal witness. When placing c, assign it to threaten an nonthreatened vertex $r \in R'$ (if there is any) with the leftmost right endpoint. Again, we can do this modification to any smallest witness by a similar reasoning as in the beginning of the proof.

If there are any nonthreatened intervals in R', threaten each nonthreatened $r \in R'$ in a symmetric way. No threatening of L' is now necessary.

If, after the previous step, there are still some undominated vertices in $A(R', L')$, dominate them by a minimal dominating set of additional cops.

The minimality of the resulting witness follows from the fact that we can modify any minimal witness to the computed witness by moving individual cops and by changing threatening matchings without increasing the size of the witness size.

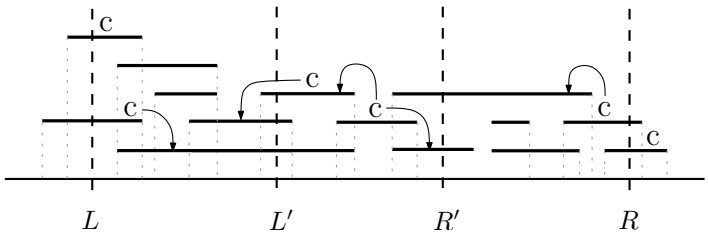


Figure 7: An example of a robber state (L, R', L', R) with $L' < R'$ and the minimal witness $C_{(L,R',L',R)}$ computed by the algorithm of Lemma 16 together with the matchings threatening L' and R'.

II. Second we consider the case $L' < R'$. We reuse most of the assumptions in the first part of the proof with minor changes. These reflect the fact that the interval (L', R') now lies at the other side of both R' and L' than before.

Assumption 1 is valid, we only need to exchange $<$ and $>$ in the proof.

Assumption 2 must be slightly adjusted — in any minimal witness, the maximum number of cops holding L is selected to take the (leftmost possible) vertices of L'. This switch between $l \in L$ threatening $a \in L'$ and a cop $c \notin L$ threatening $b \in L'$ with the (right endpoint of b) $<$ (right endpoint of a) can be done as $c \in A(R', L')$ adjacent to a is also adjacent to b. A symmetrical change is necessary for the case of R and R'.

Assumption 3L has to be adjusted in a similar way — in any minimal witness the cop assigned to threaten $l \in L'$ is either holding L or is positioned at the interval c with rightmost right endpoint adjacent to l but still in $A(R', L')$ and therefore not entirely right of R'. The reasoning is symmetrical to the first part of the proof. A similar (symmetric) modification has to be applied to *Assumption 3R*.

The algorithm constructing a smallest witness $C_{(L,R',L',R)}$ is a minor modification of the above algorithm.

The algorithm first determines a maximum matching between L and L' threatening the leftmost vertices of L', and between R and R' threatening the rightmost vertices of R'. Note that this can be done by a simple greedy algorithm.

The second step is to threaten each nonthreatened $l \in L'$ by a cop on the interval c with the rightmost right endpoint adjacent to l but still in $A(R', L')$. When placing c, assign it to threaten an nonthreatened vertex $r \in R'$ (if there is any) with the rightmost left endpoint.

If there are any nonthreatened intervals in R', threaten each nonthreatened $r \in R'$ in a symmetric way.

Note, that the algorithm does **not** dominate the remaining undominated vertices in $A(R', L')$, as this is not required by the witness.

The minimality of the resulting witness follows from the fact that we can modify any minimal witness to the computed witness by moving individual cops and by changing threatening matchings without increasing the size of the witness size.

Now it only remains to check the size of the computed witness against the total number of cops.

Without discussing the details, we note that the algorithm can be made to run in time $O(n^2)$ since all the parts can be implemented using greedy algorithms. □

Now the polynomiality follows from the polynomial size of the game state graph:

Theorem 17. *There is a polynomial time algorithm that, given an interval graph G and the number of cops k, decides the* Barrier game *on G.*

Proof. There are only $O(n^4)$ game states and the directed game state graph D has only $O(n^4)$ edges — $O(n^2)$ from each cop-state and two from each robber state. According to Lemmas 15 and 16, the existence of every edge can be decided in polynomial time and therefore the entire D can be constructed in polynomial time.

Since the initial and cop-win states are identified in constant time, it only remains to run the generic game deciding algorithm described in Section 2.3 that marks all the cop-win states in time $O(|D|) = O(n^4)$. After the algorithm finishes, it remains to check whether the initial state was marked as cop-win or not. If so, then the cops have a winning strategy by following the cop-win states. If not, then the robber has a strategy avoiding all the cop-win states. □

We also get a nice upper bound on the length of the game from the fact, that there are only n^2 cop-states and no shortest winning strategy can visit the same game state twice.

Corollary 18. *If the cops have a winning strategy for the* Barrier game *on a graph with n vertices, then the shortest strategy has length at most n^2.*

7 Game reductions

In this section we show the equivalence between the Fast robber game and the Barrier game by *game strategy reductions*. From cop's winning strategy for one game we construct a cop's winning strategy for the other game. In the following introduction, let A be the first game we know the winning strategy $\mathcal{S}$ for and B be the target game.

The core idea is to associate states of game B with states occurring in the tree of $\mathcal{S}$ if these two represent "similar" positions.

We have to ensure that the new strategy $\mathcal{S}'$ from any state S_B (corresponding to a state S_A of $\mathcal{S}$) either ends up in some state T_B corresponding to a state T_A reachable from S_A in $\mathcal{S}$, or captures the robber.

We also need to ensure that from the initial position, $\mathcal{S}'$ reaches a state S_B corresponding to a state S_A such that S_A occurs in $\mathcal{S}$; and that when $\mathcal{S}$ from S_A moves to a winning position, so will $\mathcal{S}'$ from every S_B corresponding to S_A.

These conditions are sufficient — by playing the cops using $\mathcal{S}'$, we repeatedly hit states of S_B^i corresponding to states S_A^i of $\mathcal{S}$. Since $\mathcal{S}$ has finite length, the length of $\mathcal{S}'$ is also finite and the cops eventually capture the robber.

Note that when both the length of the original strategy is bounded by x and the number of moves between hitting S_B^i is bounded by y, then the length of the resulting strategy is bounded by xy. This is the case for both the reductions we use.

To avoid some technicalities in the following discussion, we simplify the states of the Fast robber game: Since in a robber-state (C, r) it does not matter where *exactly* the robber started, just in which component of $G \setminus C$, we prefer to see the robber-states as tuples (C, M) of cop positions and the possible target vertices of robber's move. Note that M includes the vertices in the neighborhood of C.

We also suppose that the strategy for the Fast robber game captures the robber whenever the robber is in the neighborhood of a cop, both when the strategy is given and when we construct it. When appropriate, we may also assume that the robber avoids the neighborhood of every cop if possible.

The definition of corresponding states is the same for both reductions. Note that the definition of correspondence is symmetric.

The Barrier game cop-state corresponding to a the Fast robber game robber-state (C, M) is the state (L, R), where L is the first barrier left of M held by the cops in C and (symmetrically) R is the first barrier right of M held by the cops.

The Fast robber game robber-state corresponding to a Barrier game cop-state (L, R) is any state (C, M) with C holding both L and R and $M \subseteq G \setminus C$ maximal connected between L and R. The barriers L and R should be adjacent to M, but that is not always strictly necessary.

The decision to match the robber-states of one game with the cop-states of the other may seem confusing, but both these states in fact represent a simple playground with the robber anywhere inside.

7.1 From Barrier game to Fast robber game

We start with the (easier) case of reducing a Barrier game cop's strategy to a Fast robber game cops' strategy.

Lemma 19. *If there is a winning strategy $\mathcal{S}$ for k cops in the* Barrier game *on G, then there is a winning strategy $\mathcal{S}'$ for k cops in the* Fast robber game *on G. Also,* $\text{len}(\mathcal{S}') < |V_G| \, \text{len}(\mathcal{S})$.

Proof. Let $\mathcal{S}$ be a winning strategy for k cops in the Barrier game. We construct a strategy $\mathcal{S}'$ for k cops in the Fast robber game.

The cops start in a witness for the first move in the Barrier game that $\mathcal{S}$ makes from $(-\infty, +\infty)$.

Let (C, M) be the current state of the Fast robber game with a corresponding state (L, R) in $\mathcal{S}$. Strategy $\mathcal{S}'$ acts depending on the advised by $\mathcal{S}$:

Move type 1. When the strategy $\mathcal{S}$ commands to make a type 1 move to (L', R'), we let the cops move to the position $C' = C_{(L,R)\to(L',R')}$ in such a way that the cops holding L and R stay on their places. Then the cops threatening L' and R' in the witness move to L' and R' in one move. The resulting state corresponds to (L', R') because the robber could not escape from the playground.

Move type 2. When the strategy $\mathcal{S}$ commands to make a type 2 move to the robber-position (L, R', L', R), we let the cops move to the position $C' = C_{(L,R',L',R)}$ in such a way that the cops holding L and R stay on their places. The resulting state corresponds to (L, R', L', R) as the robber could not escape from the playground.

This case continues with a robber's move. Let $r \in A(L, R)$ be the vertex robber has moved to.

- If $r \in A(L, R')$, let the cops take the cut R' while holding L. The new corresponding cop-state is (L, R').
- If $r \in A(L', R)$, let the cops take the cut L' while holding R. The new corresponding cop-state is (L', R).
- If $r \in A(R', L')$ and $R' < L'$, the robber is immediately captured by the cops, as in that case $A(R', L')$ is dominated by $C_{(L,R',L',R)}$.

All these moves are possible since the cops occupy $C_{(L,R',L',R)}$ and the robber cannot escape from (L, R). All the possible resulting robber-states correspond to cop-states directly reachable from (L, R) in $\mathcal{S}$, or robber's capture.

The only way $\mathcal{S}$ can move to a playground with no internal vertices (for example when $L = R$) is by a type 2 move from (L, R) with the witness dominating the entire playground. In this case, the strategy $\mathcal{S}'$ immediately captures the robber.

That the number of cop-moves in the Fast robber game between consecutive hits of states in $\mathcal{S}$ is bounded by the diameter of the graph, and that is bounded by $n - 1$. □

7.2 From Fast robber game to Barrier game

Here we show the converse, that we can create a winning strategy for the Barrier game from a winning strategy for the Fast robber game on the same graph.

Lemma 20. *If there is a winning strategy $\mathcal{S}$ for k cops in the* Fast robber game *on G, then there is a winning strategy $\mathcal{S}'$ for k cops in the* Barrier game *on G.*
Also, $\text{len}(\mathcal{S}') = O(\text{len}(\mathcal{S})|V_G|)$.

Proof. Let $\mathcal{S}$ be the winning strategy for k cops in the Fast robber game. We construct a strategy $\mathcal{S}'$ for the Barrier game.

We make a small changes to the Fast robber game to saves us some technicalities. We consider all the robber-states (C, M) with M dominated by C to be winning for the cops. From such state, the cops can win the game with one move.

We assume that $\mathcal{S}$ starts with all the cops on the vertex with an interval with the rightmost left endpoint. This is easy to achieve, as $\mathcal{S}$ can then move the cops to their originally intended starting positions ignoring the robber. Let this initial position be (C_0, M_0) and note that $M_0 = G \setminus C_0$. This saves us trouble discussing the initial state of the Barrier game, as (C_0, M_0) corresponds to a cop-state directly reachable from the initial state $(-\infty, +\infty)$.

In the initial state, strategy $\mathcal{S}'$ moves from $(-\infty, +\infty)$ to $(-\infty, -\infty, -\infty, R_0)$ where $(-\infty, R_0)$ is the state corresponding to (C_0, M_0). From this state the robber either loses or moves to $(-\infty, R_0)$ — the state corresponding to (C_0, M_0).

If we have a cop-state (L, R) corresponding to a winning state (C, M), $\mathcal{S}'$ should move to (L, L, R, R) with C as a witness and win in one turn.

Now we consider a general cop-state (L, R) corresponding to a robber-state (C, M) reachable in $\mathcal{S}$. Let $P = (L_i^*, R_i^*)$ be the cop-states corresponding to the robber-states $F = (C_i^*, M_i^*)$ reachable in $\mathcal{S}$ from (C, M) within one turn in the Fast robber game.

Let $P' \subseteq P$ be inclusion-minimal subset of playgrounds covering $M \setminus N[C]$, that is, every vertex of M not adjacent to C (and therefore a reasonable candidate for robber's next position) is in some interval in P'. Let $F' \subseteq F$ be a inclusion-minimal set of robber-states corresponding to P'. We assume that the indices of corresponding states in F' and P' match.

Notice that every playground (L_i^*, R_i^*) of P' contains at least one vertex not threatened by C and not contained in any other playground in P' (otherwise we could remove (L_i^*, R_i^*) from P'). This also ensures that the groups of cops threatening L_i^* and R_i^* are disjoint and not adjacent and that no vertex is contained in more than two playgrounds in P'.

We ensure that strategy $\mathcal{S}'$ reaches one of the playgrounds in P' in at most n moves.

See Figure 8 for a sketch of the situation.

If $|P'| = 0$ then C dominates M and the state is winning. Let $\mathcal{S}'$ move to (L, L, R, R) with C as a witness. From (L, L, R, R), the robber loses in one turn.

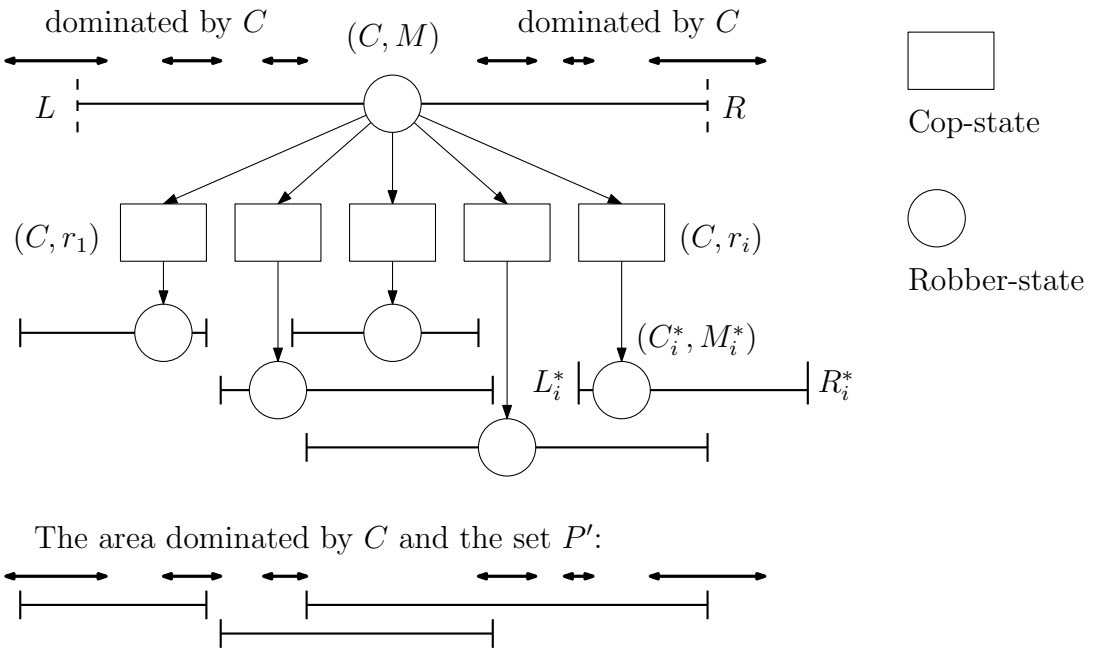


Figure 8: One move of $\mathcal{S}$ in the FAST ROBBER GAME and the playgrounds corresponding to the robber-states. There is one cop-state (C, r) for every $r \in M$. Note that in this case P' contains only three resulting robber-states and the rest is redundant.

For $|P'| = 1$ and $(L_1^*, R_1^*) \in P'$ consider several cases:

- If $A(L, R) \subseteq A(L_1^*, R_1^*)$, then let $\mathcal{S}'$move to (L_1^*, R_1^*) by expanding the playground, using C as the witness.
- If $A(L_1^*, R_1^*) \subseteq A(L, R)$, let $\mathcal{S}'$move to (L, L, L_1^*, R_1^*) with C as the witness and let the robber choose one of the playgrounds. The cops either win or end up in (L_1^*, R_1^*).
- If $A(L_1^*, R_1^*)$ and $A(L, R)$ overlap, combine the two moves: First expand the playground to (L_1^*, R) or (L, R_1^*) and then move to $(L_1^*, L_1^*, L_1^*, R_1^*)$ or $(L_1^*, R_1^*, R_1^*, R_1^*)$, using C modified by the expansion as the witness.

Situation with $|P'| \geq 2$ is solved by induction on $|P'|$. We shall reduce the size of P' by repeatedly giving the robber a choice between two subsets of P'. It would seem fastest to always divide P' into two halves, but that would not give us any advantage and we prefer to take the playgrounds one by one from the left side.

See Figure 9 for an illustration of the original and resulting game tree.

Initially, let $P_1 = P'$, $F_1 = F'$, $L_1 = L$, $R_1 = R$, $C_1 = C$ and $M_1 = M$.

We shall keep the invariant that (L_i, R_i) corresponds to (C_i, M_i), $M_i \setminus N[C_i]$ is covered by P_i, $|P_i| \leq |P'| - i$, the playgrounds in P_i correspond to the states in F_i and it is possible to take the barriers of any $(C_j^*, M_j^*) \in F_i$ from C_i with one move. We also require that P_i is minimal in the same sense that P' is. These are satisfied for $i = 1$.

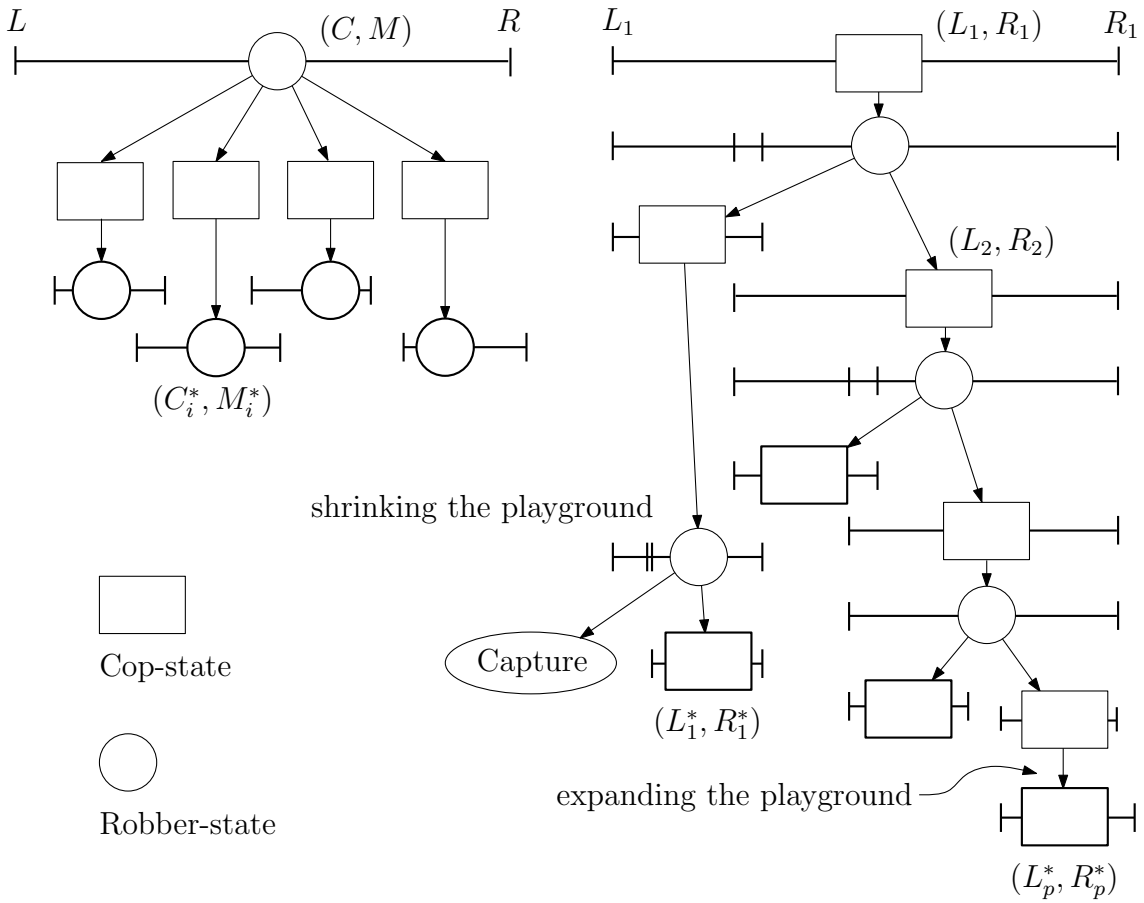


Figure 9: The transition from one step of a strategy in the FAST ROBBER GAME *(left)* to a series of steps in a strategy for the BARRIER GAME *(right)*. The set P_i of target playgrounds shrinks with each choice of the robber. Note that for $j = 1, p$, it might be necessary to expand or shrink the resulting playground to get (L_j^*, R_j^*).

From (L_i, R_i), strategy $\mathcal{S}'$ plays a type 2 move with C_i as the witness. The goal is to let the robber choose between the leftmost playground of P_i and the remaining $|P_i| - 1$ playgrounds.

Let L_1^*, R_1^* be the first and L_2^*, R_2^* be the second leftmost playground of P_i. Strategy $\mathcal{S}'$ moves to the state $(L_i, R_1^*, L*_2, R_i)$, using C_i as the witness. The witness satisfies all the conditions since we have the condition that from C it is possible to take either R_1^* or $L*_2$ and that these cops do not incide with these threatening the other barriers in P_i nor with these holding L_i or R_i. This follows from the fact that every playground contains a vertex of $A(R_i, L_i)$ not contained in any other playground and not threatened by C_i.

Left. In case the robber chooses (L_{i+1}, R_{i+1}) as the new playground, we let $P_{i+1} = \{(L_{i+1}, R_{i+1})\}$ and the new C_{i+1} to be C_i with the cops threatening R_1^* moved to R_1^*. M_{i+1} is now the interior of $A(L_{i+1}, R_{i+1})$. F_{i+1} is set to contain only (C_{i+1}, M_{i+1}). Checking the conditions for $i+1$ is straightforward, $|P_{i+1}| = 1$ and the induction stops.

Right. In case the robber chooses (L_2^*, R_{i+1}) as the new playground, we set P_{i+1} to be P_i without the first playground and the new C_{i+1} to be C_i with the cops threatening L_2^* moved to L_2^*. M_{i+1} is now the interior of $A(L_{i+1}, R_{i+1})$. F_{i+1} is F_i without the state corresponding to the leftmost playground. Again, checking the conditions for $i+1$ is straightforward. If $|P_{i+1}| = 1$, the induction stops, otherwise continue with $i+1$.

In both cases we need to ensure that P_i is a minimal cover of $M_i \setminus N[C_i]$. This can mean that we get $|P_i| = 0$, but that leaves us in the same situation as with $|P'| = 0$ above and the cops capture the robber.

In case we get to $|P_i| = \{(L_j^*)\}$, it remains to ensure that $L_{i+1} = L_j^*$ and $R_{i+1} = R_j^*$. This is a situation very similar to the case $|P'| = 1$ and we only have to check the possibility of the moves. Observe, that if $1 < j < p$ (the final playground is neither the leftmost nor the rightmost in P'), then $A(L_j^*, R_j^*)$ is not incident nor adjacent to L_1 or R_1 (the original playgrounds barriers) and therefore $L_i = L_j^*$ and $R_i = R_j^*$ and we are done. This follows from the observation that if L_i (or R_i) changes, it is set to some $L_{j'}^*$ (or $R_{j'}^*$).

If $j = 1$ (or $j = p$), we have that $R_i = R_1^*$ (or $L_i = L_p^*$) by the same reasoning. The left (or right) barrier is shifted exactly as in the case of $|P'| = 1$. Note that the cops left of $N[R_1]$ (or right of $N[L_p]$) are in the same positions as in C_1.

We have shown that in at most $p \leq n$ steps in the Barrier game, $\mathcal{S}'$ moves to one of the playgrounds in P'. That playground corresponds to one of the Fast robber game robber-states in F'. This way, $\mathcal{S}'$ moves the game to a state corresponding to a state in $\mathcal{S}$ at least one Fast robber game move closer to capturing the robber.

To get the bound on the length of the strategy, note that $|P'| = O(|V_G|)$ and that each pair of cop and robber move decreases P_i by at least one.

This finishes the proof. □

8 Conclusion

In this section, we briefly sum up author's results and propose some future directions.

We define the Fast robber game to be the Cop and robber game with an infinitely fast robber and compare it to the Helicopter game and the Cop and robber game. That game is a specific version of the Cop and robber game with s-fast robber introduced by Fomin et al. [12, 13].

We answer an open question from the same paper by showing that the Fast robber game is polynomially decidable on interval graphs. In order to prove this, we introduce a new artificial game, called the Barrier game, and prove its decidability in polynomial time and the equivalence with the Fast robber game.

Theorem 7 characterizes all the graphs where one cop has a winning strategy in the Fast robber game. We also describe a linear time algorithm recognizing such graphs.

Section 5 includes author's proofs of Lemma 11 and Theorems 10 and 12. These results are older and are included for completeness.

We conclude the results with a sketch of a possibility to extend the Fast robber game, the Barrier game and the reductions to a super-class of interval graphs and a subclass of chordal graphs.

Let the *leafage* of a chordal graph G be the minimum number of leaves of a tree T such that G can be represented as an intersection graph of subgraphs of T. The definition is due to Lin et al. [21].

For a fixed l, it might be possible to decide the Fast robber game on graphs of leafage at most l in polynomial time using a reduction to a Barrier game on the underlying tree with at most l leaves. Unfortunately, the Barrier game would have order of $|V_G|^l$ cop-states and slightly more robber-states. Also, any generalized reduction using the same ideas as we do in Section 7 would be technically very difficult. Therefore, we leave the generalization unsolved.

References

[1] Albert, M. H., Nowakowski, R. J., and Wolfe, D. *Lessons in Play: An Introduction to Combinatorial Game Theory.* AK Peters, USA, 2007.

[2] Alspach, B. Searching and sweeping graphs: A brief survey. *Le Matematiche 59* (2004), 5–37.

[3] Banaschewski, B., and Pultr, A. Tarski's Fixpoint Lemma and combinatorial games. *Order 7*, 4 (1990), 375–386.

[4] Berlekamp, E. R., Conway, J. H., and Guy, R. K. *Winning ways for your mathematical plays.* Academic Press, New York, 1982.

[5] Bertossi, A. A. Dominating sets for split and bipartite graphs. *Inf. Process. Lett. 19* (1984), 37–40.

[6] Bollobás, B. *Modern graph theory.* Graduate Texts in Mathematics 184. Springer New York., 1998.

[7] Bonato, A., Golovach, P., Hahn, G., and Kratochvíl, J. The search-time of a graph. *Discrete Mathematics (to appear).*

[8] Booth, K. S., and Lueker, G. S. Testing for the consecutive ones property, interval graphs, and graph planarity using PQ-tree algorithms. *J. Comput. Syst. Sci. 13* (1976), 335–379.

[9] Courcelle, B., and Mosbah, M. Monadic second-order evaluations on tree-decomposable graphs. *Theor. Comput. Sci. 109*, 1&2 (1993), 49–82.

[10] Diestel, R. *Graph theory. 3rd ed.* Graduate Texts in Mathematics 173. Springer Berlin., 2006.

[11] Downey, R. G., and Fellows, M. R. *Parameterized Complexity.* Springer-Verlag, Heidelberg, 1998.

[12] Fomin, F. V., Golovach, P., Kratochvil, J., Nisse, N., and Suchan, K. Pursuing a fast robber on a graph. *Theoretical Computer Science (submitted)* (2009).

[13] Fomin, F. V., Golovach, P. A., and Kratochvíl, J. On tractability of cops and robbers game. In *IFIP TCS* (2008), pp. 171–185.

[14] Fomin, F. V., and Thilikos, D. M. An annotated bibliography on guaranteed graph searching. *Theor. Comput. Sci. 399*, 3 (2008), 236–245.

[15] Garey, M. R., and Johnson, D. S. *Computers and Intractability; A Guide to the Theory of NP-Completeness.* W. H. Freeman & Co., New York, NY, USA, 1990.

[16] Gavenčiak, T. Cop-win graphs with maximum capture-time. *Discrete Mathematics (submitted)*.

[17] Gavenčiak, T. Cop-win graphs with maximum capture-time. Master's thesis, Charles University, Prague, Czech Republic, 2007.

[18] Goldstein, A., and Reingold, E. The complexity of pursuit on a graph. *Theoretical computer science 143*, 1 (1995), 93–112.

[19] Golombic, M. C. *Algorithmic Graph Theory and Perfect Graphs*. Academic Press, New York, 1980.

[20] Kloks, T. *Treewidth, Computations and Approximations*, vol. 842 of *Lecture Notes in Computer Science*. Springer, 1994.

[21] Lin, I.-J., McKee, T. A., and West, D. B. The leafage of a chordal graph. *Discussiones Mathematicae – Graph Theory 18* (1998), 23.

[22] McKee, T. A., and McMorris, F. R. *Topics in Intersection Graph Theory*. Society for Industrial Mathematics, 1999.

[23] Nowakowski, R., and Winkler, P. Vertex to vertex pursuit in a graph. *Discrete Math. 43*, 2 (1983), 235–239.

[24] Papadimitriou, C. H. *Computational complexity*. Addison-Wesley Longman, 1995.

[25] Quilliot, A. A short note about pursuit games played on a graph with a given genus. *Journal of combinatorial theory. Series B 38*, 1 (1985), 89–92.

[26] Raman, V., and Saurabh, S. Short cycles make w-hard problems hard: Fpt algorithms for w-hard problems in graphs with no short cycles. *Algorithmica 52*, 2 (2008), 203–225.

[27] Robertson, N., and Seymour, P. D. Graph minors II: algorithmic aspects of tree-width. *Journal Algorithms 7* (1986), 309–322.