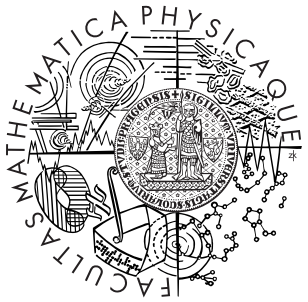
Univerzita Karlova v Praze
Matematicko-fyzikální fakulta

BAKALÁŘSKÁ PRÁCE



Tomáš Gavenčiak

Hry na grafech

Katedra aplikované matematiky

Vedoucí bakalářské práce: Prof. RNDr. Jan Kratochvíl, CSc.
Studijní program: Informatika

2007

Děkuji svému vedoucímu za pomoc při řešení problému i za čas věnovaný úpravě textu. Dále děkuji Petru Škovroňovi a dalším kamarádům za pomoc při korekturování práce a dotváření její konečné podoby.

Prohlašuji, že jsem svou bakalářskou práci napsal samostatně a výhradně s použitím citovaných pramenů. Souhlasím se zapůjčováním práce a jejím zveřejňováním.

V Praze dne 30.5.2007 Tomáš Gavenčiak

Název práce:	Hry na grafech
Autor:	Tomáš Gavenčiak
e-mail autora:	gavento@atrey.karlin.mff.cuni.cz
Katedra:	Katedra aplikované matematiky
Vedoucí bakalářské práce:	Prof. RNDr. Jan Kratochvíl, CSc.
e-mail vedoucího práce:	honza@kam.mff.cuni.cz

Abstrakt: V této práci studujeme některé vlastnosti jedné hry typu cop&robber, při které se dva hráči – Policista (Cop) a Lupič (Robber) – střídají v tazích na konečném neorientovaném grafu. Oba hráči se pohybují rychlostí nanejvýš jedna hrana za tah a znají v každém okamžiku celý stav hry. Pokud se Policista kdykoli ocitne na stejném poli jako Lupič, vyhraje Policista. Pokud k tomuto nikdy nedojde, vyhrává Lupič. Hry tohoto typu jsou důležité jako modely prohledávání grafu i pro svou souvislost s invarianty zdvihu grafu. Zabýváme se blíže vlastnostmi grafů, na kterých existuje výherní strategie pro Policistu (tzv. *cop-win* grafy), a hledáním nejlepších strategií pro oba hráče. Již dříve bylo známo, že počet tahů, za který je Policista schopen vyhrát na jakémkoli cop-win grafu na n vrcholech je zhora omezen $n - 3$, a existují grafy vyžadující $n - 4$. V této práci ukazujeme, že tento počet je právě $n - 4$.

Klíčová slova: hry typu cop&robber, teorie her, teorie grafů, prohledávání grafu

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

Abstract: In this thesis we study properties of one cop&robber game. In this game two players (Cop and Robber) take turns in moving on a finite undirected graph. Both players move with the speed at most one edge per turn. They both know the complete game status. If at any time Cop shares a vertex with Robber, Cop wins. If that never happens, Robber wins. Games of this type are important as models of searching in graphs and networks and for the connection to the width parameters of graphs. We closely examine the class of graphs with a winning strategy for Cop (the so called *cop-win* graphs) and construct best strategies for both Cop and Robber. The previously known results include the fact that the number of moves in which Cop can catch Robber on every cop-win graph on n vertices is bounded by $n-3$ and there are graphs which require $n - 4$. We show that this number is exactly $n - 4$.

Keywords: cop&robber games, game theory, graph theory, graph searching

Contents

1 Introduction **5**

2 Graph theory used **5**

3 Game description **7**
3.1 Game rules 7
3.2 An Example Game 8
3.3 Game origin and previous work 8
3.4 Variants of the game 9

4 Strategies and time **9**
4.1 Winning strategies 10

5 Cop-win graphs **13**
5.1 Examples of cop-win graphs 13
5.2 The Triangle lemma 14
5.3 The Folding Theorem 15

6 Cop-time **17**
6.1 Upper bounds 18
6.2 Lower bound 20
6.3 The Function $\mathcal{MCT}$ 22

7 Algorithms and software **22**
7.1 Cop-time algorithm 23

8 Conclusion **24**

Appendix A **25**

References **26**

1 Introduction

In Section 2 we introduce the definitions of all the graph-theoretic expressions used in this thesis. Section 3 describes the rules of the examined game and gives game examples, references to the articles about the related topics, and summarizes the prevoius work on the studied game. The definitions and basic properties of strategies are in Section 4. A close examination of the class of cop-win graphs can be found in Section 5. Section 6 contains the main theorem of this thesis limiting the maximum cop-time. The Section 7 briefly describes the algorithms and programs used in early hypotesis testing.

2 Graph theory used

Here we present the definitions of all the graph-theoretic terms used in this thesis. We mostly use the terminology and notation presented in Chapter 1 of Diestel's book [Die06]. A reader familiar with general graph theory and the common notation can skip this section.

Def. An *undirected graph* 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 *digraph* is a pair (V, E), where V is a set of vertices and $E \subseteq V^2$ are ordered tuples also called the *directed edges*. The directed edge (u, v) is said to lead from the vertex u to the vertex v. Digraphs are used only as auxiliary structures in this thesis.

Def. A *graph* means an undirected graph unless stated otherwise. If $G = (V, E)$ is a graph, $V(G) = V$ denotes its vertex set and $E(G) = E$ denotes its edge set. Note that some properties are common for both graph types and then the distinction is not necessary.

Def. The *order* of a graph G, denoted by $|G|$, is the number of its vertices. In this thesis we deal only with finite graphs and therefore the graph order is always finite. The number of edges of G is denoted by $||G||$.

Def. The *endpoints* or the *ends* of an edge $e = \{u, v\}$ are the two vertices u and v. Vertices $u, v \in V$ are *adjacent* in G if $\{u, v\}$ is an edge of G. Instead of writing the edges as sets, we usually write them briefly just as $e = uv$ where this is unambiguous. Note that no vertex is adjacent to itself.

Def. The *neighbours* $N_G(v)$, or briefly $N(v)$, of a vertex v in a graph G are all the vertices adjacent to v. $N_G[v]$ or briefly $N[v]$ denotes the set of neighbours of v including the vertex v.

Def. *Deleting edges and vertices:* In case v is a vertex of G, the graph $G \setminus v$ denotes the graph G without v and all the edges ending in v. In case e is an edge of G, the graph $G \setminus e$ denotes the graph G without that edge, but still containing both of its endpoints.

Def. *Graph with an additional edge:* In case $e = uv$ is not an edge of G and both ends of e are vertices of G, $G + e$ denotes graph $(V, E \cup \{e\})$.

Def. The *contraction* of an edge $e = uv$ is a graph where the vertices u and v are unified into the vertex u. All the edges ending in v are replaced by edges from the same vertices but ending in u.

Def. The *degree* $\deg_G(v)$ of a vertex v is the number of edges of G ending in v. Usually written shortly as $\deg(v)$ where unambiguous. The *maximum degree* of a graph, denoted by $\Delta(G)$, is the maximum degree of its vertices. Note that for undirected graphs $\Delta(G) \leq |G| - 1$.

Def. $G' = (V', E')$ is a *subgraph* of $G = (V, E)$ if $V' \subseteq V$ and $E' \subseteq E$. Sometimes we say that G' *is present in* G, or shortly *is in* G.

There are several common graph types used in this thesis. They either have interesting properties related to the cop&robber game or appear in proofs and constructions.

Def. The *complete graph* on n vertices is a graph $K^n = (V, \binom{V}{2})$ on an arbitrary set $|V|$=n. In this graph all the vertices are adjacent.

Def. The *path* of length $n \geq 1$ is a graph $P^n = (V, E)$ where $V = \{x_0, x_1, x_2, \ldots, x_n\}$, all the x_i are distinct, and $E = \{x_0x_1, x_1x_2, x_2x_3, \ldots, x_{n-1}x_n\}$. The vertices x_0 and x_n are the *endpoints* or *ends* of the path. The path is said to *connect* its ends.

Def. A graph G is *connected* if for every distinct $u, v \in V(G)$ there is a path in G connecting u and v. Otherwise the graph is *disconnected* and its maximal connected subgraphs are called its *components*.

Def. The *distance* of vertices u and v in G, denoted by $\text{dist}_G(u, v)$ or shortly $\text{dist}(u, v)$, is length of a shortest path connecting u and v in G. If no such path exists, $\text{dist}(u, v) = \infty$.

Def. The *circle* or *cycle* of length $n \geq 3$ is a graph $C^n = (V, E)$ where $V = \{x_1, x_2, \ldots, x_n\}$, all the x_i are distinct, and $E = \{x_1x_2, x_2x_3, \ldots, x_{n-1}x_n, x_nx_1\}$. This graph has n edges forming a single cycle. Any subgraph of G which is a circle is usually called a *cycle in* G. Note that $C^n = P^{n-1} + x_{n-1}x_0$.

Def. Graph G is a *tree* when it satisfies the following equivalent conditions:
G is a connected graph with no cycles.
G has an unique path connecting every two $u, v \in V(G)$.
G is connected and $||G|| = |G| - 1$.
Proving the equivalence is an easy exercise. See Diestel's book [Die06] for details.

Def. A *chord* of a cycle C of graph G is an edge $e \in E(G)$ with both ends in $V(C)$ but itself not an edge of C. A cycle of G is said to be *chordless* if it has no chord in G.

Def. Graph G is said to be *chordal* if every cycle on more than 3 vertices present in G has a chord. Equivalently, G must have no chordless cycles of length more than 3. Note that every tree is chordal.

Def. An edge $e = uv$ in G is a *bridge* if there is only a single path from u to v. If G is connected, $G \setminus e$ is disconnected iff e is a bridge. The *ratio* of a bridge is the ratio of the sizes of the two arising components.

Def. The *wheel* of order $n \geq 3$, denoted by W^n, is a graph consisting of C^n (the *circumference*), one additional vertex v (the *axis*), and edges from v to all vertices of C^n (the *spokes*).

Vertices of a graph are usually displayed as dots or small circles, sometimes labelled with the vertex names. The edges are displayed as lines or arcs connecting their endpoints. In the drawing the edges may cross outside vertices; for some graphs this is unavoidable. Details of the drawing, such as the concrete positions of the vertices or the lengths of edge arcs or lines are unimportant. In the case analysis used in Lemma 5 we draw potential edges with a dashed line. Note that usually not every subset of potential edges can be present at once.

3 Game description

This section describes the considered cop&robber game in detail, provides an example game, and contains references to similar games and previous work on this game.

3.1 Game rules

The game is played on a finite connected undirected graph G. One player is called *Cop* and the second player is called *Robber*. They both can only stand at a vertex of G and use an edge of G to move to an adjacent vertex. They both know where the other player is and they both have a full knowledge of the structure of G and of the rules.

First, Cop chooses her starting vertex, then Robber chooses his with respect to Cop's position. Then the game is played in turns, Cop and Robber alternate in moves starting with Cop. One move consists of changing one's position to an adjacent vertex or deciding to stay at the current one. If at any moment Cop and Robber stand at the same vertex, Cop catches Robber and wins. If Robber can escape for arbitrarily long time, he wins.

If the game is finite, its *length* is the number of Cop's moves including the turns when Cop decides to stay at the same vertex but excluding the initial choosing of the starting vertices.

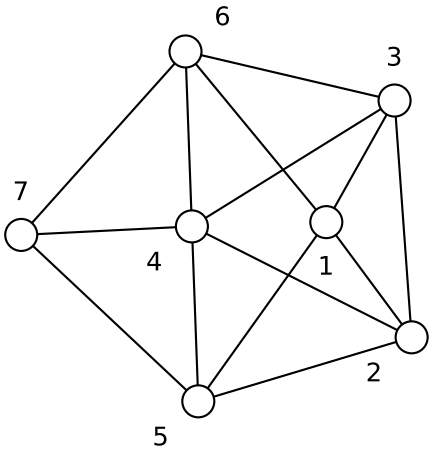


Figure 1: Example graph

3.2 An Example Game

Consider the graph on Figure 1. If, for example, Cop chooses to start at the vertex 7, Robber must start on the vertex 1, 2, or 3 to avoid an immediate defeat. It could seem a wise move for Cop to move to 4 and thus force Robber to move to 1, but then whichever vertex 2, 3, 5, or 6 Cop chooses, Robber can escape to one of 2, 3, 5, or 6. Suppose Cop moved to 6. Robber then must escape to 2 or 5. Cop playing in this way would probably sooner or later catch Robber, but her strategy is poor and slow against such a clever Robber.

Now we show Cop's optimal strategy for this graph. Cop will start at 3 (she could start at 2 due to symmetry), Clever Robber then must start at 5 or 7. If Robber started at 7, Cop moves to 4 and whatever Robber does, Cop catches him in her next move. If Robber started at 5, Cop should move to 2, forcing Robber to flee to 7. But then again Cop moves to 4, Robber moves arbitrarily and Cop catches him with her third move.

3.3 Game origin and previous work

The game was first described independently by Nowakowski and Winkler [NW83] and Quilliot [Qui78][Qui83]. Their results characterise the cop-win graphs. Nowakowski and Winkler propose a polynomial algorithm to determine the cop-time of a given finite graph.

A simple variant of the studied game can be obtained by giving the first player more Cops to command. For every graph G, $|G|$ Cops clearly suffice to catch Robber. The least number of Cops able to catch any Robber is called the *cop-number* or the *search-number* of G. All the variants with discrete alternating vertex-to-vertex moves are called *searching* of graphs.

Several authors studied bounds on search number (even on infinite graphs), many results and references may be found in Geňa [Hahn06] and Bonato et al. [BGHK06]. The previously known results include a description of the structure of cop-win graphs

and a sequence of graphs with $\mathcal{CT}(G) = |G| - 4$ different from the one presented in Section 6.2. The previously known bounds on the maximal cop-time were $|G| - 4 \leq \mathcal{MCT}(G) \leq |G| - 3$.

3.4 Variants of the game

A rich family of similar games with selected results and a brief history can be found in Alspach [Als04]. Here we shortly present a few examples.

Historically first was a problem of locating a person lost in a cave system. The cave system is simplified as a graph. A limited number of rescuers can continuously move on graph edges and vertices with a limited speed. They can meet, turn or stop at any point in an edge or at a vertex. The goal is to devise their movement such that they will catch the invisible victim regardless of victim's trajectory or speed. Inspired by his spelunker friend Richard Breisch, T. D. Parsons [Par76, Par78] first considered this as a graph theoretic *sweeping* problem.

The rules of the game considered in this thesis allow both players to pass a move, this can be represented by disabling waiting and playing on a reflexive graph (a graph with a loop edge at every vertex). By placing the loops only at some vertices we can naturally disable waiting in some or all the vertices.

Another variant with more Cops may allow Robber to move with higher speed than the Cops, but disallow Robber to walk over a standing Cop. The problem becomes closer to the sweeping problem when Robber can move with an infinite speed (but still has to end his move on a vertex). Playing with an invisible Robber is even closer to sweeping.

Directing the edges of G yields a digraph; in this variant moving is allowed only in prescribed direction.

There are countless variants of the problem, some of them with an useful relationship to other areas of mathematics (i.e. computational complexity, tree width). Surprisingly, many similar variants have often a very different solution.

Concepts of the considered game are somehow similar to the board game "Scotland Yard" (Ravensburger, 1983) or its Czech modification "Fantom Staré Prahy" (JAVOZ Jablonec nad Nisou, 1987), where several cops pursue one phantom across a simplified graph of a city public transport system. The parallel is only distant, because in these games more independent players play the cops and the phantom's position is partially secret.

4 Strategies and time

In this section we formalise the game rules and the concept of strategies. We prove that either Cop or Robber has a winning strategy and we sketch efficient method for construction of a winning strategy.

The *game of cop&robber* is a sequence of Cop and Robber positions $c_0, r_0, c_1, r_1, c_2, r_2, c_3, \ldots$. First, Cop selects a starting vertex c_0, Robber then selects r_0 with respect to c_0. In i-th round (starting with $i = 1$) Cop chooses her next position c_i from

set $N_G[c_{i-1}]$ with respect to Robber's position r_{i-1}, then Robber chooses his next position r_i from set $N_G[r_{i-1}]$ with respect to Cop's position c_i.

A *strategy* is, generally, a sort of a plan that advices player his/hers next move in every game situation. The strategy advice can depend on the actual game state or the previous course of play, it can even use randomness and answer differently in exactly the same situations. In this thesis we consider only *pure strategies* which in identical situations advice the same move, but they still may consider the previous course of play.

However, as we show below, there is always a pure winning strategy considering only the current game state, We work only with these strategies and call them *forgetful strategies*. Theorem 2 shows that one of the players has a pure forgetful *winning strategy*; the player using this strategy wins the game regardless how the other player plays. The theorem also shows that if Cop has a winning strategy, no Robber's play can delay her win more than the play of Robber using some pure forgetful strategy.

4.1 Winning strategies

Here we formally define Cop's and Robber's forgetful strategies, game states, the state space and the graph of possible state changes. For a given graph G we also construct the *time function* t giving the number of Cop's moves sufficing to catch any Robber from a given state. Then we state and prove the theorem justifying the restriction to forgetful strategies.

Def. A *Game state* of a game on G is an ordered triple of the form

$$(p, c, r) \in \{Cop, Robber\} \times (V(G) \cup \{\emptyset\})^2,$$

where p is the player on turn, c is the Cop's position, and r is the Robber's position.

Def. A *cop-state* is a state with Cop on turn and a *robber-state* is a state with Robber on turn.

Def. A *Cop's strategy* for game on a graph $G = (V, E)$ is a function

$$S_C : (V \cup \{\emptyset\})^2 \to V,$$

such that $S_C(c, r)$ gives next Cop's move from a state (Cop, c, r). $S_C(\emptyset, \emptyset)$ gives the preferred Cop's starting position. The strategy must advise only legal moves in any state.

Def. A *Robber's strategy* for game on a graph $G = (V, E)$ is a function

$$S_R : V \times (V \cup \{\emptyset\}) \to V,$$

such that $S_R(c, r)$ gives next Robber's move from a state $(Robber, c, r)$. $r_0 = S_C(c, \emptyset)$ gives Robber's starting position for Cop's starting position c. The strategy must advise only legal moves in any state.

Def. P_G denotes the *game state space* of a game on a graph G, which is a set of all the possible legal game states.

Def. $\mathcal{G}_G = (P_G, M_G)$ is the digraph of the possible moves (Cop's move from cop-state to robber-state and vice versa), M_G denotes the set of all ordered pairs of possible successive states. Note that $\mathcal{G}_G$ is finite.

For the rest of this section we define one *action* to be either one move as described in the rules, or the initial placing of Cop or Robber. This term is used only in this section and makes counting of the moves a bit easier. Note that every game after $n \geq 1$ Cop's moves contains $n+1$ Cop's actions. The same holds for Robber's moves.

In order to prove the main theorem of this section we construct a sequence of sets T_i for a given fixed graph G. For the rest of this section we fix the graph G, $V = V(G)$, $P = P_G$, and $(P, M) = \mathcal{G} = \mathcal{G}_G$.

Def. T_0 is the set of all the states where Cop just won (either because she caught Robber or Robber ran in her arms):

$$T_0 = \{\, (p, v, v) \mid p \in \{Cop, Robber\}, v \in V(G) \,\}.$$

Only in these states Cop needs no more actions to capture Robber.

Def. T'_i is the set T_{i-1} plus all the cop-states p from $P \setminus T_{i-1}$ such that there is an edge in $\mathcal{G}$ leading from p to some state in T_{i-1}. Then T_i is the set T'_i plus all the robber-states q from $P \setminus T'_{i-1}$ such that all the edges in $\mathcal{G}$ leading from q end in some state in T'_i.

The described construction yields an infinite sequence of the sets T_i. If $T_{i+1} = T_i$ for some i then $T_j = T_i$ for all $j > i$. This follows from the fact, that the set T_{i+1} depends only on T_i.

The equality $T_i = T_{i+1}$ must be true for some i because $T_0 \subseteq T_1 \subseteq T_2 \subseteq T_3 \subseteq \cdots \subseteq P$ and the set P is finite. T denotes the union $\bigcup_{i \to \infty} T_i$.

Lemma 1. *For T_i construed for G as above and every $i \geq 0$ the following is satisfied:*

T_i is the set of all the states p such that in every game on G starting in the state p Cop can catch Robber in at most i actions regardless how Robber plays. (1)

Proof. We inductively prove that (1) holds for every $i \geq 0$. For $i = 0$ this is trivially true as noted in the definition of T_0. For $i \geq 1$, suppose that (1) holds for every smaller i.

First we prove that in every game starting in every state of T_i Cop can catch Robber in at most i actions regardless how Robber plays. In $\mathcal{G}$ all the edges from every robber-state $p \in T_i$ end in states from T'_i, so whichever action Robber chooses, next game state p' will be in T'_i. In $\mathcal{G}$ from every cop-state $p' \in T'_i$ there is an edge to some robber-state $p'' \in T_{i-1}$, so with one action Cop can change game state to p'' and catch Robber in at most $i - 1$ actions according to (1). Note that every Cop-state from T_i is also in T'_i.

It remains to prove that from every state in $P \setminus T_i$ Robber can avoid Cop for at least i Cop's actions. By definition of T_i, from every $p \in P \setminus T_i$ there is an edge ending in state p' outside T'_i. Robber can use this edge and next game cop-state will be in $P \setminus T'_i$. From every cop-state $p' \in P \setminus T_i$ it is impossible for Cop to change game state to some state in T_{i-1}, so by (1) she can't catch Robber in $(i-1)+1 = i$ actions. Note that every cop-state from $P \setminus T_i$ is also in $P \setminus T'_i$.

This induction completes the proof of (1) for every i. □

Def. The *time function* $t(p)$ is the smallest i such that $p \in T_i$. If there is no such i, we set $t(p) = \infty$. The time function for a graph G is denoted by t_G.

Theorem 2. *In every simple finite connected graph G either Cop or Robber has a forgetful winning strategy.*

Proof. Here we prove this theorem from scratch, although it could be derived from existing general game-theoretic theorems.

Let T_i, T, and t be constructed for G as defined above. If $(Cop, \emptyset, \emptyset) \in T$, then Cop can catch every Robber in $\mathcal{CT} = t((Cop, \emptyset, \emptyset)) - 1$ moves and Robber can avoid Cop for $\mathcal{CT} - 1$ Cop's moves regardless how Cop plays. Remember, that initial placing of Cop and Robber doesn't count as move but does count as one action.

If $\mathcal{CT} < \infty$ and therefore Cop can always win in $\mathcal{CT}$ moves, she can do so with the following forgetful strategy:

In the cop-state p choose (from all states reachable by one action) a state q with the lowest $t(q)$. According to Lemma 1 there is such a state with $t(q) = t(p) - 1$. This strategy assures Cop's win in $\mathcal{CT}$ moves for any Robber's behaviour.

Robber can use the following simple strategy to escape for at least $\mathcal{CT} - 1$ Cop's moves (or indefinitely in the case $\mathcal{CT} = \infty$):

In state p select a possible next state q with $t(q) \geq t(p)$. There must be at least one such q, otherwise Cop would be able to catch Robber in fewer than $t(p)$ actions, which is a contradiction with Lemma 1. This strategy assures Robber's survival for at least $\mathcal{CT} - 1$ Cop's moves (or indefinitely in case $\mathcal{CT} = \infty$) for any Cop's behaviour. □

Def. If there is a winning strategy for Cop on G, we call G a *cop-win* graph; otherwise there is a winning strategy for Robber and we call G a *robber-win* graph.

$\mathcal{CT}$ and constructed strategies are useful in the following sections. Below we define them as graph properties.

Def. The *cop-time* of graph G, denoted by $\mathcal{CT}(G)$, is $\mathcal{CT}(G) = t_G((Cop, \emptyset, \emptyset)) - 1$.

Def. A *best Cop's strategy* is one of the Cop's fastest forgetful winning strategies constructed for G as in the proof above in case of a cop-win graph G. If G is a robber-win graph, $S_C(G)$ is an arbitrary strategy.

Def. A *best Robber's strategy* is one of the forgetful strategies assuring Robber the longest possible survival in every game on G as constructed in the proof above.

Def. The *time of cop's strategy* S_C on a graph G denoted by $\mathcal{CT}^*(S_C, G)$ is the minimum number of moves necessary for Cop using strategy S_C to catch Robber playing in every way. If Robber can avoid Cop using S_C for arbitrarily long time, $\mathcal{CT}^*(S_C, G) = \infty$. Note that a best Cop's strategy S'_C has $\mathcal{CT}^*(S'_C, G) = \mathcal{CT}(G)$ and no Cop's strategy S_C can have $\mathcal{CT}^*(S_C, G) < \mathcal{CT}(G)$.

Def. The *time of Robber's strategy* S_R on a graph G denoted by $\mathcal{RT}^*(S_R, G)$ is the maximum number such that Robber using S_R will survive at least $\mathcal{RT}^*$ Cop's moves against every Cop. If Robber using S_R can escape every Cop, $\mathcal{RT}^*(S_R, G) = \infty$. If G is a cop-win graph, then $\mathcal{RT}^*(S_R, G) < \mathcal{CT}(G)$ and for a best Robber's strategy S'_R $\mathcal{RT}^*(S'_R, G) = \mathcal{CT}(G) - 1$.

The proof of Theorem 2 does not only show the existence of a winning strategy for one player, but also shows how this strategy can be constructed from the time function t and the T_i's. All these can be easily constructed algorithmically. The algorithm determining t_G, $S_C(G)$, $S_R(G)$ and $\mathcal{CT}(G)$ is shown in Section 7.

5 Cop-win graphs

Before we state the main property of the cop-win graphs, we provide a few examples of cop-win graph classes. Then we prove a simple lemma about a necessary condition on local graph structure, which is useful in Section 6 below.

5.1 Examples of cop-win graphs

Two very simple graph classes containing only cop-win graphs are the complete graphs, where for $|G| > 1$ always $\mathcal{CT}(G) = 1$, and trees, where $\mathcal{CT}(G) \leq \lfloor |G|/2 \rfloor$. In a tree, Cop selects a vertex with the lowest distance to the farthest leaf. The upper bound for trees is reached by paths, where Cop must walk from the path centre to one of its ends.

Both these simple classes are subclasses of the class of chordal graphs. In a chordal graph Cop can catch Robber by simply moving towards him. We briefly sketch a proof of this strategy.

When Cop stands on some v in a chordal graph G, Robber can move safely only to the vertices in the components of the graph $G \setminus N_G[v]$; moving to a vertex in $N_G[v]$ would allow Cop to directly catch him. If Cop moves closer to Robber to v', Robber can not change the component, because every path between his old and another component of $G \setminus N_G[v']$ goes through $I = N_G[v] \cap N_G[v']$. If there would be such a path P connecting the Robber's component with another component and avoiding I, it would give rise to a chordless cycle gained from the cyclic sequence of adjacent vertices of the form $vv' \ldots P \ldots v$ (where dots represent some path to and

from the ends of P) by taking the smallest cycle in G on vertices of P containing vv'.

Robber gets eventually caught, because the size of his "safe" component decreases to zero. All the chordal graphs have $\mathcal{CT}(G) \leq \lfloor |G|/2 \rfloor$, because in every G there is always a vertex with distance at most $\lfloor |G|/2 \rfloor$ from every other vertex. For proof, details and other interesting properties of chordal graphs see Diestel [Die06]. This proof is very informal and incomplete, but we present stronger results below.

There are also non-chordal cop-win graphs, for example wheels. Generally, every graph changes to cop-win graph by adding edges from one selected vertex to all others. For all such graphs $\mathcal{CT} = 1$. Examples of a chordal graph and a wheel are on Figure 2.

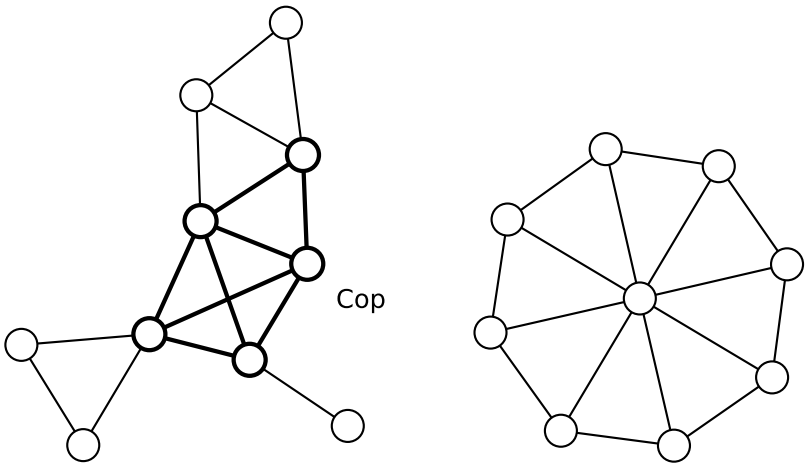


Figure 2: Some cop-win graphs – a chordal graph with the area threatened by Cop in bold and a wheel

Yet another example not falling into any category above is the graph on Figure 1. This graph is used as example in Section 3 and is again encountered in Section 7.

5.2 The Triangle lemma

Intuition suggests that every cop-win graph containing a big chordless cycle must contain additional vertices somehow specially connected to the cycle, otherwise Robber could keep his distance from Cop just by moving around the cycle. The following lemma describes one local property of all cop-win graphs. It is used in Section 6.

Triangle Lemma. *If $G = (V, E)$ is a cop-win graph, then each edge $e = uv$ of G is either a bridge or there is a vertex $w \in V$ directly connected with both u and v. Such w is said to form a C^3* over *e.*

Proof. For the sake of contradiction suppose that there is a cop-win G with a non-bridge edge $e = vv'$ such that no C_3 contains e. Then there is a shortest path

$P{=}v_0v_1\dots v_{k-1}$ with $v = v_0$ and $v' = v_{k-1}$ in $G \setminus e$. $P \cup e$ forms a circle C in G of a length $k \geq 4$.

We construct a strategy for Robber. First, define $R(v) = (\mathrm{dist}_{G\setminus e}(v, v_0) + 2) \bmod k$. Note that the distance is measured in the graph without e, so $R(v_j) = (j+2) \bmod k$. Whenever Cop is on a vertex c, Robber should move to the position $r = v_{R(c)}$, where he cannot be attacked, because r is in a distance at least 2 from c. Whenever Cop changes position to c', her distance from v_0 changes at most by 1, so Robber can always move to an appropriate vertex on C. We must only check that Robber's move to $v_{R(c')}$ is always possible, but that simply follows from the fact, that the change of Cop's distance from v_0 is at most 1. When Cop moves from the vertex c with $R(c) = k - 2$ to c' with $R(c') = k - 1$, Robber should move from v_{k-1} to v_0. For first Cop's move c_0, Robber should play to $v_{R(c_0)}$.

This strategy could be made more intuitive for a human player by loosening the strictness and letting Robber's stand still or move almost arbitrarily around C when Cop is very far from C. □

5.3 The Folding Theorem

How must the game state look like just before Cop catches Robber? Suppose it's Robber's move, Robber is in r_i and he wants to survive as long as possible. Cop just moved to c_{i+1} and should catch Robber with her next move. The vertex r_i must be in $N(c_{i+1})$, otherwise Robber could just wait. Similarly, every neighbour of r_i must be a neighbour of c_{i+1}, otherwise Robber could move there and avoid getting caught for this time. Such state is illustrated on Figure 3. Cop just moved to Robber's vicinity and it's Robber's turn, but he has no neighbour safe from Cop. Note that some of Robber's neighbour vertices can be adjacent to each other.

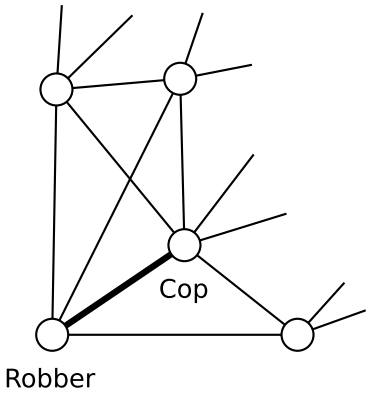


Figure 3: Situation just before catch — Robber's turn

To reflex that fact we define a new relation on vertices and state a simple lemma.

Def. Vertex v *dominates* vertex $u \neq v$ iff $N[u] \subseteq N[v]$. In other words, u and v are adjacent and every another neighbour of v is also a neighbour of u. Vertex v is *dominated* iff there is a vertex $u \neq v$ dominating v. If a vertex is not dominated, it is *non-dominated*. The set of vertices dominating a vertex v is denoted by $\mathrm{dom}(v)$ or $\mathrm{dom}_G(v)$ in a given graph G.

Lemma 3. *In every cop-win graph on at least two vertices there is at least one dominated vertex.*

Proof. For contradiction suppose that every vertex of G is non-dominated. We construct a simple strategy for Robber showing that G is robber-win.

If every vertex of G is non-dominated, then for every vertex there is some (different) non-adjacent vertex. This implies that wherever Cop starts, Robber has a safe starting vertex. Whenever Cop moves to Robber's neighbourhood, Robber can move to a vertex non-adjacent to Cop (such non-adjacent vertex exists, because Robber's current vertex is non-dominated). In other cases Robber can wait or move to unthreatened vertices. With this strategy Robber always avoids getting caught. □

Def. In situation where u dominates v, the contraction of the edge uv is called the *folding of* v. Note that the resulting graph is the same as $G \setminus v$. The reverse process is adding a new isolated vertex v and attaching the new vertex v to u and to arbitrary number of neighbours of the vertex u. This is called an *unfolding of vertex* u.

Folding of a vertex keeps the graph either cop-win or robber-win and its cop-time does not change too much, as stated by the following theorem.

Folding Theorem. *Let G be a simple connected graph with a vertex v dominated by a vertex u. Let $G' = G \setminus v$.*

a) If G' is robber-win, G is also robber-win.

b) If G' is cop-win, G is also cop-win and $\mathcal{CT}(G') \leq \mathcal{CT}(G) \leq \mathcal{CT}(G') + 1$.

Proof. a) If Robber has a winning strategy S'_R for the game on G', he has almost the same winning strategy S_R for the game on G, except that in a state with Cop at v the strategy S_R plays as if Cop was at u. This strategy never moves to v. Cop on v has fewer options than on u, so if Robber could avoid her using S'_R, he can avoid her with S_R.

b) Let S' be a best Cop's strategy on G' with $\mathcal{CT}^*(S', G') = \mathcal{CT}(G')$. We extend the strategy S' to a winning strategy S on G. These rules are added or modified in S:

• Cop can move to v only to directly catch Robber and if she can, she must do so.

• In a state when Robber is at v and Cop is at a vertex non-adjacent to v, Cop moves as if Robber was on u.

• In a state when Robber is at v and Cop is at u, Cop catches Robber by moving to u.

The new vertex v does not help Robber much — whenever he moves to v he could also move to u and with his next move he can move to $N[v] \subseteq N[u]$. The strategy S assures that at latest in $\mathcal{CT}(G)$ moves Cop either catches Robber or Robber is in v and Cop in u with Robber on the move. But in the later situation Cop catches Robber with her next move. So G is cop-win and $\mathcal{CT}(G) \leq \mathcal{CT}^*(S, G) \leq \mathcal{CT}^*(S', G') + 1 = \mathcal{CT}(G') + 1$.

For contradiction now suppose that $\mathcal{CT}(G') > \mathcal{CT}(G)$, in that case Robber has a best strategy S'_R on G' with $\mathcal{RT}^*(S'_R, G') = \mathcal{CT}(G') - 1$ according to Theorem 2 and the definition of $\mathcal{RT}^*$. We extend S'_R to a strategy S_R on G: Robber should neither move to v nor start there; if Cop moves to v or starts at v, Robber plays as if Cop was at u. Moving to v doesn't help Cop in any way, she won't catch Robber there and she limits her possibilities. Thus every Cop's best strategy S_C catching Robber using S_R can be adjusted to S'_C avoiding v without slowing Cop down. But from this follows

$$\mathcal{CT}(G') = \mathcal{RT}^*(S'_R, G')+1 \leq \mathcal{RT}^*(S_R, G)+1 \leq \mathcal{CT}^*(S'_C, G) \leq \mathcal{CT}^*(S_C, G) = \mathcal{CT}(G),$$

which is a contradiction and therefore $\mathcal{CT}(G') \leq \mathcal{CT}(G)$. □

Corollary. *Every cop-win graph can be folded down to one vertex. Every cop-win graph can be constructed from the graph on one vertex by consecutive unfolds. Every unfold increases cop-time of a graph by zero or one.*

Proof. This immediately follows from the Folding Theorem and its proof. □

Note that the Triangle Lemma also simply follows from the Folding Theorem, but we consider it's proof presented in Section 5.2 to be more synoptic.

6 Cop-time

The main focus of this thesis is on graphs with the maximum cop-time. This section defines the function $\mathcal{MCT}$ as the maximum finite cop-time of graphs with a given number of vertices and shows its exact value.

Def. We define the *maximum cop-time on n vertices* as a function

$$\mathcal{MCT}(n) = \max_{\substack{\text{cop-win G,} \\ |G|=n}} \mathcal{CT}(G).$$

Lemma 4. *For every $n \geq 1$:*

$$\mathcal{MCT}(n) \leq \mathcal{MCT}(n+1) \leq \mathcal{MCT}(n) + 1.$$

Proof. This easily follows from Conclusion 5.3. Every graph G with $n+1$ vertices and $\mathcal{CT}(G) = \mathcal{MCT}(n+1)$, G can be folded to G' satisfying $\mathcal{MCT}(n)+1 \geq \mathcal{CT}(G')+1 \geq \mathcal{CT}(G) = \mathcal{MCT}(n+1)$. $\mathcal{MCT}$ is non-decreasing, because every G unfolded in every way to G^* satisfies $\mathcal{CT}(G) \leq \mathcal{CT}(G^*)$. □

From this lemma and the fact that $\mathcal{MCT}(2) = \mathcal{CT}(P^1) = 1$ immediately follows the upper bound $\mathcal{MCT}(n) \leq n-1$. On paths we also have $\mathcal{MCT}(n) \geq \mathcal{CT}(P^{n-1}) = \lfloor n/2 \rfloor$, so we get $\mathcal{MCT} = \Theta(n)$.

6.1 Upper bounds

A simple case analysis of all cop-win graphs on 3 vertices gives $\mathcal{MCT}(3) = 1$ and thus $\mathcal{MCT}(n) \leq n-2$. A similar analysis of all cop-win graphs on 5 vertices yields the upper bound $\mathcal{MCT}(n) \leq n-3$, but there are too many cop-win graphs on 7 vertices for a hand-analysis. These and larger graphs can be analysed on a computer, as shown below in Section 7. Here we find all the graphs on 6 vertices with the maximum cop-time and show that no unfold can increase their cop-time.

Lemma 5. *The only graph on* 6 *vertices with* $\mathcal{CT}(G) \geq 3$ *is* P^5.

Proof. We analyse all connected cop-win graphs on 6 vertices separately according to their maximum degree. We estimate $\mathcal{CT}$ of every case and also write some good starting vertices for Cop. Dashed edges in the illustration figures represent the possible edges (sometimes not all the edges can be present at once).

- G with $\Delta(G) \in \{0, 1\}$ is clearly not connected.
- A connected graph G with $\Delta(G) = 2$ is either a path P^5 or a circle C^6. C^6 is not a cop-win graph and $\mathcal{CT}(P^5) = 3$.
- If G with $\Delta(G) = 3$ has no bridge, then every edge has to be in some C^3 according to the Triangle Lemma. Select an arbitrary v with $\deg(v) = 3$. We denote the neighbours of v v_1, v_2 and v_3. v_1v must be in C^3, but v can have no more neighbours. Therefore v_1 is adjacent to, say, v_2. A similar argument shows that v_3 must be adjacent to v_1 or v_2, without loss of generality suppose it is adjacent to v_2. The remaining vertices p, q must be connected to v; suppose that v_1p is an edge (v_2 can have no more neighbours). There is no way to form a C^3 over v_1p. The situation is drawn on Figure 4. No cop-win graph with $\Delta = 3$ and without a bridge exists.

  If G with $\Delta(G) = 3$ has a bridge which divides the vertices in the ratio $3:3$, G clearly has $\mathcal{CT} = 2$, Cop can start in any endpoint of the bridge. If G has no bridge with the ratio $3:3$, but has a bridge uv with the ratio $2:4$ with 2 vertices on the side of u, then v must have $\deg(v) = 3$, otherwise G would have a bridge with the ratio $3:3$. The situation is drawn on Figure 4. Whichever of the remaining possible edges exist, $\mathcal{CT}(G) = 2$ with Cop starting at v.

  If G with $\Delta(G) = 3$ has a bridge uv with the ratio $1:5$ with a single vertex on the side of u, but has no bridges with the ratio $2:4$ or $3:3$, then again must $\deg(v) = 3$. Denote neighbours of v u, v_1 and v_2. The edges v_1v and v_2v can't be both bridges (they can't both have the ratio $1:5$ and the other possible ratios are forbidden), so one of them has to be in a C^3, but this can happen only if v_1 and v_2 are adjacent. The remaining vertex p is connected to v_1 or v_2, without loss of generality suppose that it is connected to v_1. The situation is drawn on Figure 4. If pv_1 is a bridge, remaining q cannot be connected to p (pv_1 would be bridge with ratio $2:4$), so q is adjacent to v_2. If pv_1 is not a bridge, it must be in C^3, but that

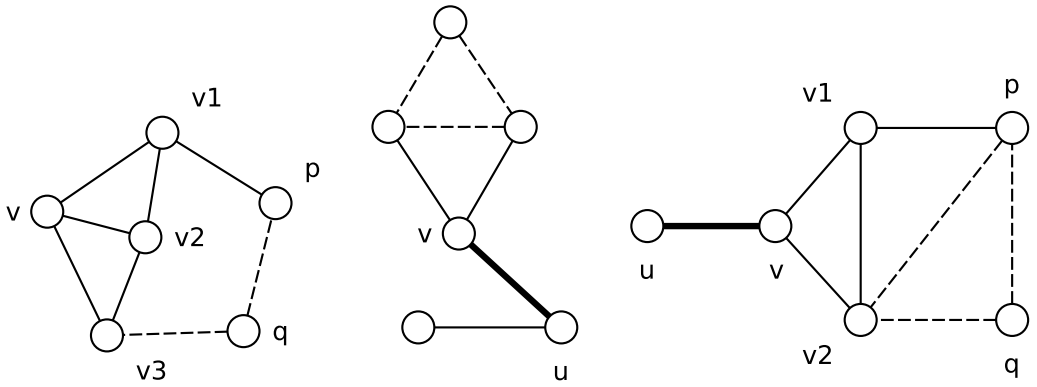


Figure 4: The cases of $\Delta(G) = 3$ and no bridge, a bridge with the ratio $2 : 4$, and a bridge with the ratio $1 : 5$

is possible only if pv_2 is an edge. In that case q is adjacent only to p. In both cases $\mathcal{CT}(G) = 2$ with Cop starting at v_1 or v_2.

• In G with $\Delta(G) = 4$ denote by v one vertex with $\deg(v) = 4$ and u the single vertex non-adjacent to v. If u has a single neighbour v_1, clearly $\mathcal{CT}(G) = 2$ with Cop starting at v. If u has neighbours v_1 and v_2, these have to be adjacent (to form C^3 over uv_1) and again $\mathcal{CT}(G) = 2$ with Cop starting at v. If u has three neighbours v_1, v_2 and v_3, there have to be at least two edges between v_1, v_2 and v_3 to form C^3s over uv_1, uv_2 and uv_3. Without loss of generality suppose that v_1v_2 and v_2v_3 are edges. Then Cop starting at v can catch Robber either immediately or after moving to v_2. This case is illustrated on Figure 5.

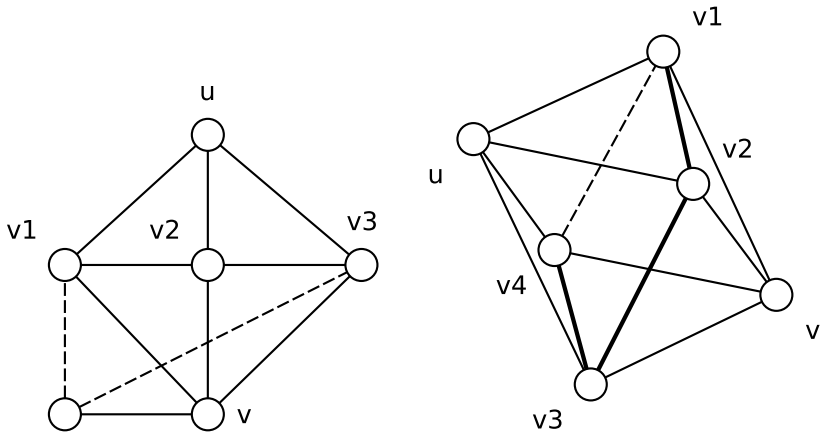


Figure 5: Cases of $\Delta(G) = 4$ with $\deg(u) = 3$ and $\deg(u) = 4$

In the last case when u has four neighbours v_1, v_2, v_3 and v_4, there have to be at least two edges e_1 and e_2 between v_{1-4} to form C^3s over uv_1, uv_2, uv_3 and uv_4. No v_i can be connected to all the other v_i (it would have $\deg(v_i) = 5$). If e_1 and e_2 share an endpoint, there has to be an additional e_3 forming P^3 (together with e_1 and e_2) on the set $\{v_1, v_2, v_3, v_4\}$. If e_1 and e_2 do not share an endpoint, there must be another edge connecting the endpoints of e_1 and e_2, otherwise G would be robber-win. Both these situations necessarily lead to a graph isomorphic to that on Figure 5, where a concrete situation with the path $v_1v_2v_3v_4$ is drawn. If v_1v_4 is an edge, the graph is the net of a regular octahedron which is robber-win (Robber can always move to the vertex opposite of Cop). If v_1v_4 is not an edge, Cop wins in 2 moves by starting at v_2 or v_3 and therefore $\mathcal{CT}(G) = 2$.

- G with $\Delta(G) = 5$ has $\mathcal{CT}(G) = 1$ because Cop can start in a vertex with the maximum degree and catch Robber with her first move.

This shows that every cop-win graph G on 6 vertices except P^5 has $\mathcal{CT}(G) \leq 2$. $\mathcal{CT}(P^5) = 3$ and that finishes the proof. □

Lemma 6. *$\mathcal{MCT}(n) \leq n - 4$ for all $n \geq 7$.*

Proof. Every cop-win graph G on 7 vertices with $\mathcal{CT}(G) \geq 4$ can be constructed by an unfold from a cop-win graph G' on 6 vertices with $\mathcal{CT}(G') \geq 3$ according to the Folding Theorem. The only cop-win graph with cop-time at least 3 is P^5. An easy case analysis shows that no unfold of P^5 increases the cop-time and therefore there is no graph on 7 vertices with $\mathcal{CT} \geq 4$. In the analysis it suffices to check all the unfoldings of 3 vertices of P^5 due to symmetry. This reduces the computation of $\mathcal{CT}$ to 8 graphs, all chordal. Cop can start in one of the central vertices of P^5 closer to the unfolded vertex and then simply walk towards Robber. The detailed analysis is left to the reader as a simple exercise. □

6.2 Lower bound

The lower bound for the function $\mathcal{MCT}$ is proved constructively. Paths on at most 7 vertices give $\mathcal{MCT}(7) \geq \mathcal{MCT}(6) \geq 3$, $\mathcal{MCT}(5) \geq \mathcal{MCT}(4) \geq 2$, $\mathcal{MCT}(3) \geq \mathcal{MCT}(2) = 1$, and $\mathcal{MCT}(1) = 0$. For every $n \geq 8$ we construct a graph G^n on n vertices with $\mathcal{CT}(G^n) = n - 4$.

Def. G^n is the graph on Figure 1 used in the example game in Section 3.2 with a path P^{n-8} attached by an edge to its only dominated vertex as shown on Figure 6. G^7 is the graph without the path. The vertices of G^n are numbered an will be referred to by their numbers.

Lemma 7. *$\mathcal{CT}(G^n) = n - 4$ for every $n \geq 7$.*

Proof. The only dominated vertex and thus the only place where Cop can catch Robber is the vertex n. The optimal strategy for Cop is to start at 2 (or symmetrically at 3). Then Robber can start at $6, 7, \ldots n$. Whenever Robber moves to Cop's vicinity,

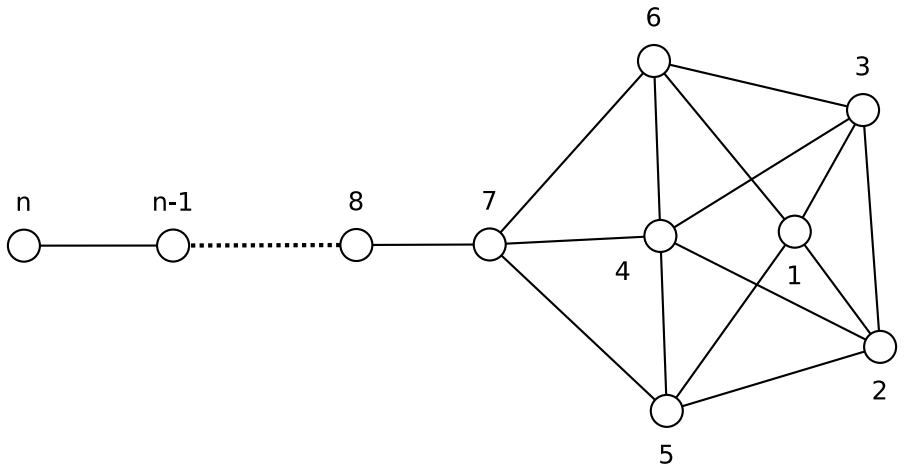


Figure 6: Graph G^n

she should catch him immediately. If Robber starts at 6, Cop moves to 3 forcing Robber to 7, then she moves to 4 forcing Robber to 8 (or catching him in the case $n = 7$). Then Cop moves towards vertex n and catches Robber in at most $n-4$ moves (of course Robber can get caught sooner if he does not flee from Cop's vicinity). If Robber stats at $7, 8, \ldots n$, Cop moves to 4, then to 7 and continues to n. This way she catches Robber in at most $n-3$ moves. This shows, that $\mathcal{CT}(G^n) \leq n-4$. This also follows from Lemma 6, but we find this constructive proof more illustrating.

To prove the lower bound, we construct Robber's best strategy. Robber avoiding the threatened vertices can't get caught at vertices $1 \ldots 6$ — none of these vertices is dominated and therefore Robber always has an unthreatened neighbour. If Robber moves to unthreatened 7 with Cop at 1, 2, or 3 after at least one Cop's move, he can simply flee to n and Cop must make at least $(n-7)+3 = n-4$ moves to catch him there (at least one move before Robber's move to 7, 2 more to reach vertex 7 and $n-7$ to reach vertex n). It suffices to show that for every Cop's starting position Robber can start at an unthreatened vertex $1 \ldots 6$. We analyse several cases depending on Cop's starting position.

In the case when Cop starts at 1, Robber should start at 4. If Cop's first move is to 2 or 3, Robber moves to 7 and flees to n. If Cop's first move is to 5 (or 6), Robber moves to 3 (or 2). Then he plays arbitrarily just avoiding moves to the threatened vertices, eventually moves to 7 and then flees to n.

If Cop starts at 2 (or 3), Robber should start at 6 or (5). If Cop moves to 1, 2 or 3, Robber moves to 7 and flees. If Cop's first move is to 4, 5 or 6, Robber moves to the unthreatened vertex 1, 3 or 2 (respectively) and then he plays arbitrarily avoiding the threatened vertices.

If Cop starts at $4, 5, 6, \ldots n$, Robber should start at an unthreatened vertex 1, 2, or 3 and wait for Cop to come into his vicinity. Then Robber plays arbitrarily avoiding the threatened vertices. This way Robber doesn't move to 7 sooner than after two Cop's moves.

Wherever Cop starts, it takes her at least $n-4$ moves to catch Robber and therefore $\mathcal{CT}(G^n) \geq n-4$. If Robber would be never forced to move to 7, he would win the game.

This finishes the proof. □

6.3 The Function $\mathcal{MCT}$

Facts gathered in the previous sections suffice to formulate and prove the main theorem of this thesis:

Maximum cop-time Theorem. *The function $\mathcal{MCT}$ has the values $\mathcal{MCT}(1) = 0$, $\mathcal{MCT}(2) = \mathcal{MCT}(3) = 1$, $\mathcal{MCT}(4) = \mathcal{MCT}(5) = 2$, $\mathcal{MCT}(6) = 3$, and $\mathcal{MCT}(n) = n-4$ for all $n \geq 7$.*

Proof. The values of $\mathcal{MCT}(1)$, $\mathcal{MCT}(2)$, and $\mathcal{MCT}(3)$ can be simply verified. The cop-time of paths gives $2 \leq \mathcal{MCT}(4) \leq \mathcal{MCT}(5)$, but if a graph G on 5 vertices would have $\mathcal{CT}(G) = 3$, then it could be unfolded to a graph G' on 6 vertices with $\mathcal{CT}(G') \geq 3$. Such G' can be only P^5 according to Lemma 5. But every fold of P^5 generates P^4 with $\mathcal{CT}(P^4) = 2$ and thus $\mathcal{MCT}(5) = \mathcal{MCT}(4) = 2$. For $n \geq 7$ the theorem follows from Lemma 6 and Lemma 7. □

7 Algorithms and software

For an early verification of parts of the hypothesis we used a computer program analysing all the non-isomorphic connected graphs on at most 10 vertices. To generate the list of all such graphs we used the software package *Nauty* [McK81]. Nauty is a program for computing automorphism groups of graphs and digraphs and can be used to generate all the non-isomorphic graphs satisfying some simple properties (such as connectivity).

The program computing the cop-time of a graph is written in the programming language Python[1]. It loads the graphs from a list generated by Nauty, and writes the $\mathcal{CT}$ and the set of Cop's best starting positions for all the graphs with the maximum $\mathcal{CT}$. The program itself is quite simple and not of high importance — it is a straight implementation of the described algorithm without any optimisations. Neither the program nor its results figure in any of the proofs presented in this thesis.

The computation of the cop-time for all the connected non-isomorphic graphs on 9 vertices takes approximately 2 minutes on a 2GHz PC computer. The same for all the connected non-isomorphic graphs on 10 vertices takes over 2 hours on the same computer.

[1] `http://www.python.org`

7.1 Cop-time algorithm

The described algorithm is listed in Appendix A.

For a fixed graph G the algorithm gradually constructs the time function t as a map from P to $\mathbb{N} \cup \{\infty\}$. Initially, $t[p] = \infty$ for all the states $p \in P$.

First the algorithm sets $t[(Cop, v, v)] := 0$ and $t[(Robber, v, v)] := 0$ for every $v \in V$ (catch states). The algorithm then proceeds in several iterations. In every iteration it first sets

$$t[(Cop, c, r)] := \min_{c' \in N[c]} \{t[(Robber, c', r)] + 1\}, \tag{2}$$

and then similarly for Robber

$$t[(Robber, c, r)] := \max_{r' \in N[r]} \{t[(Cop, c, r')]\}, \tag{3}$$

both for every $c, r \in V$ in any order. It also sets t for the initial placement states

$$t[(Robber, c, \emptyset)] := \max_{r \in V} \{t[(Cop, c, r)]\} \tag{4}$$

for every $c \in V$, and

$$t[(Cop, \emptyset, \emptyset)] := \min_{c \in V} \{t[(Robber, c, \emptyset)] + 1\}. \tag{5}$$

The algorithm stops when neither of these calculations change value of any $t[p]$.

Note that the steps never change a finite value of t to another value. The values being assigned only increase with the time so the calculated minimums never changes. When a recalculated maximum becomes finite, all its arguments are already finite and therefore they will not change.

This algorithm is very similar to the construction of the sets T_i defined in Section 4.1, therefore we omit the proof of this algorithm. The algorithm has a polynomial time complexity in $|G|$, as the outer cycle will finish after at most $\mathrm{O}(|G|^2)$ iterations, as $|P| = \mathrm{O}(|G|^2)$ and at least one value of t is changed from infinity to a finite number each iteration.

This algorithm could be improved in several ways, but that is not the focus of this thesis. Also, it is easy to extend this algorithm to a game with k Cops. The state space would be the set $\{Cop, Robber\} \times V^{k+1}$ and the algorithm would set

$$t[(Cop, c_1, c_2, \ldots, c_k, r)] := \min_{c_1' \in N[c_1], c_2' \in N[c_2], \ldots} \{t[(Robber, c_1', \ldots, c_k', r)] + 1\}, \tag{6}$$

for all $c_1, c_2, c_3, \cdots \in V$, and other three equations similar to (3), (4), and (5).

This algorithm has time complexity polynomial in $|G|$ when k is fixed, but exponential in k. This also yields an algorithm determining the cop-number (search number) of G with the complexity exponential in $|G|$.

8 Conclusion

All results presented in this thesis were proved independently by the author of this thesis only with consultation with the supervisor. All the results were first verified by a computer exhaustive search. Some of the notation and names were later adapted from the referenced publications.

This thesis answers the question of maximum cop-time, but the shape of all the graphs with the maximum cop-time is still unknown. We conjecture that all the graphs on $n \geq 9$ vertices with the maximum cop-time contain G^n from Section 6.2 as a subgraph. It still remains to verify this conjecture and determine how many cop-win graphs on n vertices have the maximum cop-time.

Appendix A

This appendix contains listing of the algorithm described in Section 7.1.

Algorithm 1: Computing the time function

Input: connected graph $G = (V, E)$
Output: graph time function t
t:=new map with default value ∞
for $v \in V(G)$ **do**
  $t[(Cop, v, v)] := t[(Robber, v, v)] = 0$; `#Set` t `of catch states`
repeat
  changed:=**false**
  for $c \in V$ **do**
    for $r \in V$ **do**
      if $t[(Cop, c, r)] = \infty$ **then**
        m:=**min** $\{t[(Robber, c', r)] \mid c' \in N[c]\}$
        if $m < \infty$ **then**
          $t[(Cop, c, r)] := m + 1$; `#Set` $t[(Cop, c, r)]$ `as in (2)`
          changed:=**true**
      if $t[(Robber, c, r)] = \infty$ **then**
        m:=**max** $\{t[(Cop, c, r')] \mid r' \in N[r]\}$
        if $m < \infty$ **then**
          $t[(Robber, c, r)] := m$; `#Set` $t[(Robber, c, r)]$ `as in (3)`
          changed:=**true**
    if $t[(Robber, c, \emptyset)] = \infty$ **then**
      m:=**max** $\{t[(Cop, c, r)] \mid r \in V\}$
      if $m < \infty$ **then**
        $t[(Robber, c, \emptyset)] := m$; `#Set` $t[(Robber, c, \emptyset)]$ `as in (4)`
        changed:=**true**
  if $t[(Cop, \emptyset, \emptyset)] = \infty$ **then**
    m:=**min** $\{t[(Robber, c, \emptyset)] \mid c \in V\}$
    if $m < \infty$ **then**
      $t[(Cop, \emptyset, \emptyset)] := m + 1$; `#Set` $t[(Cop, \emptyset, \emptyset)]$ `as in (5)`
      changed:=**true**
until not *changed;* `#Stop when no updates happen`
return t

References

[Als04] Brian Alspach: *Searching and Sweeping Graphs: A Brief Survey*, Le Matematische, Vol. LIX, 2004

[BGHK06] A. Bonato, P. Golovach, G. Hahn, AND J. Kratochvíl, *The search-time of a graph*, submitted (2006)

[Die06] Reinhard Diestel: *Graph Theory (Third Edition)*, Springer-Verlag, Heidelberg, 2006

[Hahn06] Geňa Hahn, *Cops, robbers and graphs* Université de Montréal, submitted (2006)

[McK81] Brendan D. McKay: *Practical Graph Isomorphism*, Congressus Numerantium, Vol 30, pp 45-87, 1981.
Nauty web page: `http://cs.anu.edu.au/~bdm/nauty/`

[NW83] R. Nowakowski and P. Winkler, *Vertex-to-vertex pursuit in a graph*, Discrete Math. 43 (1983), pp 235 - 239.

[Par76] T. D. Parsons: *Pursuit-evasion in a graph*, Theory and Applications of Graphs, pp 426–441, Springer-Verlag, 1976

[Par78] T. D. Parsons: *The search number of a connected graph*, Proc. 10th Southeastern Conf. Combinatorics, Graph Theory, and Computing, pp. 549–554, 1978

[Qui78] A. Quilliot, *Jeux et Points Fixes sur les graphes*, Ph.D Dissertation, Université de Paris VI, 1978

[Qui83] A. Quilliot, *Problèmes de jeux, de point fixe, de connectivité et de représentation sur des graphes, des ensembles ordonnés et des hypergraphes*, Université de Paris VI, 1983