Complexity of the cop & robber guarding game

R. Šámal, R. Stolař, T. Valla

Charles University, Faculty of Mathematics and Physics,
Institute for Theoretical Computer Science (ITI)
Malostranské nám. 2/25, 118 00, Prague, Czech Republic
{samal, ruda, valla}@kam.mff.cuni.cz

Abstract

The guarding game is a game in which a set of cops has to guard a region in a digraph against a robber. The robber and the cops are placed on vertices of the graph; they take turns in moving to adjacent vertices (or staying). The goal of the robber is to enter the guarded region at a vertex with no cop on it. The problem is to find the minimum number of cops needed to prevent the robber from entering the guarded region. The problem is highly nontrivial even for very simple graphs – when the robber moves in a tree, then the decision version of the problem is NP-complete. Furthermore, if the robber is moving in a DAG, the problem becomes PSPACE-complete. This was the work of Fomin, Golovach, Hall, Mihalák, Vicari and Widmayer [1]. We solve the question asked by Golovach and we show that if the graph is an arbitrary directed or undirected graph, the problem becomes ETIME-complete.

1 Introduction and motivation

The *guarding game* $(\vec{G}, V_c, c)$ is played on directed graph $\vec{G} = (V, E)$ by two players, the *cop-player* and the *robber-player*, each having his pawns (c cops and one robber, respectively) on V. There is a protected region (called also cop-region) $V_C \subset V$. The remaining region $V \setminus V_C$ is called robber-region and denoted V_R. The cops guard V_C by preventing the robber to enter the protected region without being caught, which happens when the robber is

[0]Supported by the GAUK Project 66010 of Charles University Prague.

on a vertex of V_C with a cop on it. The game is played in alternating turns. In the first turn the robber-player places the robber on some vertex of V_R. In the second turn the cop-player places his c cops on vertices of V_C (more cops can share one vertex). In each subsequent turn the respective player can move each of his pawns to a neighbouring vertex of the pawn's position (or leave it where it is). However, the cops can move only inside V_C and the robber can move only on vertices with no cops. At any time of the game both players know the positions of all pawns. The robber-player wins if he moves the robber to some vertex of V_C with no cop on it. The cop-player wins if in finite number of turns the robber-player did not win. Note that after exponentially many turns the positions has to repeat and obviously if the robber can win, he can win in less than $|V|^{(c+1)}$ turns.

For a given graph $\vec{G}$ and guarded region V_C, the task is to find the minimum number c such that cop-player wins.

Here we focus on the complexity issues of the decision problem. Given the guarding game $\mathcal{G} = (\vec{G}, V_C, c)$, how difficult is to find out the winner of the game? Observe that the task of finding the minimum c such that $\mathcal{G}$ is cop-win is at least as hard as the decision version of the problem. The decision problem was studied by Fomin et al [1]. The computational complexity of the problem depends heavily on the chosen restrictions. In particular, in [1] the authors show that if the robber's region is only a path, then the problem can be solved in polynomial time. When the robber moves in a tree (or even in a star), then the problem is NP-complete. Furthermore, if the robber is moving in a directed acyclic graph, the problem becomes PSPACE-complete.

We investigate the general case for directed graphs. Let us consider the class $ETIME = DTIME(2^{O(n)})$ of languages recognisable by a deterministic Turing machine in time $2^{O(n)}$. We consider log-space reductions, this means that the reducing Turing machine is log-space bounded. We prove the following theorem.

Theorem 1 *The game $\mathcal{G} = (\vec{G}, V_C, c)$, where $\vec{G}$ is a directed graph, is $ETIME$-complete under log-space reductions.*

After proving Theorem 1, we modify the proof and all constructions such that it works even for undirected graph, therefore yielding the following theorem.

Theorem 2 *The game $\mathcal{G} = (G, V_C, c)$, where G is an undirected graph, is $ETIME$-complete under log-space reductions.*

We would like to point out the fact that we can prove the previous theorems without both with and withou prescribing the starting positions of players.

2 The directed case

In order to prove $ETIME$-completeness of the guarding game $\mathcal{G} = (\vec{G}, V_C, c)$, we first show that $\mathcal{G} \in ETIME$. Then we take an instance of the game $\mathcal{F}$ and construct an equivalent guarding game $\mathcal{G} = (\vec{G}, V_C, c)$, therefore proving its $ETIME$-completeness.

Lemma 3 *Let $\mathcal{G} = (\vec{G}, V_C, c)$ be a cop guarding game. Then $\mathcal{G} \in ETIME$.*

Proof. We need to show that there is an algorithm deciding the outcome of $\mathcal{G}$ in $2^{O(n)}$ time, where n is the size of the instance of $\mathcal{G}$. Consider the graph H of all configurations of the game $\mathcal{G}$ – these are all possible positions of all cops and the robber, together with the information whose turn it is. The edge (c_1, c_2) belongs to $E(H)$ if and only if c_1 is cop turn and c_2 is robber turn (or vice versa) and the pawns of c_1 can be legally moved into pawn positions of c_2. There is also a starting vertex s representing empty board with edges going to every possible initial placement p_i of cop pawns. For every p_i there are edge to every possible initial subsequent placement of robber pawn r_i.

Using the following backwards-labelling algorithm we can decide the outcome of every position in polynomial time in the size of the graph H. Let us denote the robber-winning configurations by W_R.

1. Construct the graph H.
2. Put all vertices that are a win for the robber-player to W_R.
3. Add to W_R all vertices v where robber is on turn and there is an edge $(v, w) \in E(H)$ and $w \in W_R$.
4. Add to W_R all vertices v where cop is on turn and for every edge $(v, w) \in E(H)$ the vertex $w \in W_R$.
5. Repeat $|V(H)|$ times steps 3 and 4.
6. If $s \in W_R$ the game $\mathcal{G}$ is robber-win, otherwise the game $\mathcal{G}$ is cop-win.

It remains to show that the size of H is $2^{O(n)}$. The simplest upper bound on $|V(H)|$ is $(c+1)^n$, which is unfortunately superexponential if c is close to n. To find better upper bound, we use the fact that the cops are mutually indistinguishable. Then, using elementary combinatorial counting, $|V(H)|$ is bounded by

$$|V(H)| \leq 2|V_R|\binom{|V_C|+c-1}{c} \leq 2n\binom{n+c-1}{c} \leq n2^{n+c} = 2^{O(n)},$$

thus the total size of H is $2^{O(n)}$ as well. □

Let us first study the problem after the second move, where both players have already placed their pawns. We reduce the problem from the following formula-satisfying game $\mathcal{F}$.

A position in $\mathcal{F}$ is a 4-tuple $(\tau, F_R(C,R), F_C(C,R), \alpha)$ where $\tau \in \{1,2\}$, F_R and F_C are formulas in 12-DNF whose variables have been partitioned into disjoint sets R and C, and α is an $(C \cup R)$-assignment. The symbol τ serves only to differentiate the positions where the first or the second player is on move. Player I (II) moves by changing the values assigned to at most one variable in R (C); either player may pass since changing no variable amounts to a "pass". Player I (II) wins if the formula F_R (F_C) is true after some move of player I (II). More precisely, player I can move from $(1, F_R, F_C, \alpha)$ to $(2, F_R, F_C, \alpha')$ in one move if and only if α' differs from α in the assignment given to at most one variable in R and F_C is false under the assignment α; the moves of player II are defined symmetrically.

According to Stockmeyer and Chandra [2], the set of winning positions of player I in the game $\mathcal{F}$ is an $ETIME$-complete language under log-space reduction.

Let us first informally sketch the reduction from $\mathcal{F}$ to $\mathcal{G}$. We would like to construct a guarding game, which simulates the alternating setting of variables of R and C by players I and II such that whenever the formula F_R is satisfied, the robber is allowed to access the protected region V_C, and whenever the formula F_C is satisfied, the cops are able to block the entrance to V_C forever. The setting of variables is represented by positions of certain cops so that only one of these cops may move at a time. The variables (or more precisely the corresponding cops) of C are under control of cop-player. However, in spite of being represented by cops, the variables of R are under control of the robber-player by a gadget in the graph $\vec{G}$, which allows him to force any setting of cops representing R.

When describing the features of various gadgets, we will often use the term *normal scenario*. By normal scenario S of certain gadget (or even the whole game) we mean a flow of the game such that whenever the robber-player or cop-player do not exactly follow S, they will surely lose.

There are four cyclically repeating phases of the game, determined by the current position of the robber. The normal scenario is that robber cyclically goes through the following phases marked by four special vertices and in different phases he can enter certain gadgets.

1. "Robber Move" (RM): Here the robber can enter the Manipulator gadget, allowing him to force setting of variable in R.

2. "Robber Test" (RT): Here robber has a possibility to pass through the *Robber Gate* into the protected region V_C. The gate is only passable if the formula F_R is satisfied under the current setting of variables.

3. "Cop Move" (CM): If robber is here, this is the place (and the only place in the whole game) where one (and at most one) variable cell V_x for $x \in C$ is allowed to change its value. This is realized by a gadget called *Commander*.

4. "Cop Test" (CT): Here, if the formula F_C is satisfied under the current setting of variables, the cops are able to block the entrance to the protected region forever (by temporarily leaving the *Cop Gate* gadget unguarded and sending a cop to block the entrance to V_C provided by the Robber Gate gadgets).

See Figure 1 for the overview of the construction.

2.1 The variable cells

For every variable $x \in C \cup R$ we introduce a *variable cell* V_x, see Figure 2. There is one cop (*variable cop*) located in every V_x and the position of the cop on vertices T_x, F_x represents the boolean values true and false. All the vertices of V_x belongs to V_C.

The cells are organised into blocks C and R. The block C is under control of cop-player, the block R is represented by cops as well, however, there are the *Manipulator* gadgets allowing the robber-player to force any setting of variables in R, by changing at most one variable in his turn.

Every variable cell V_y, $y \in R$ has assigned the Manipulator gadget M_y (see Figure 3).

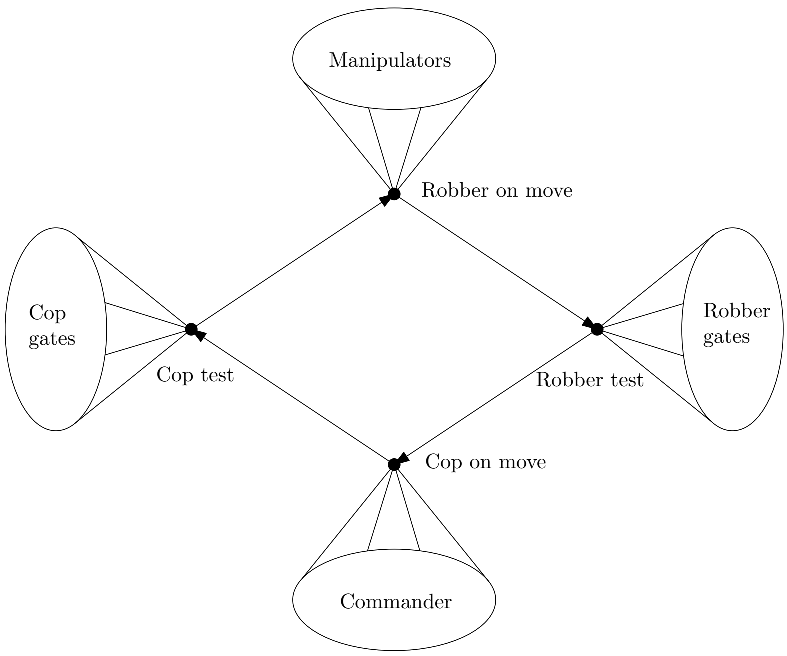

Figure 1: The sketch of the construction

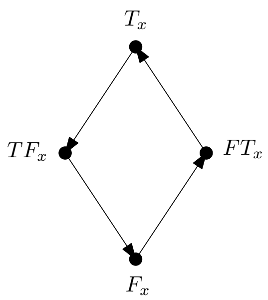

Figure 2: Variable cell V_x

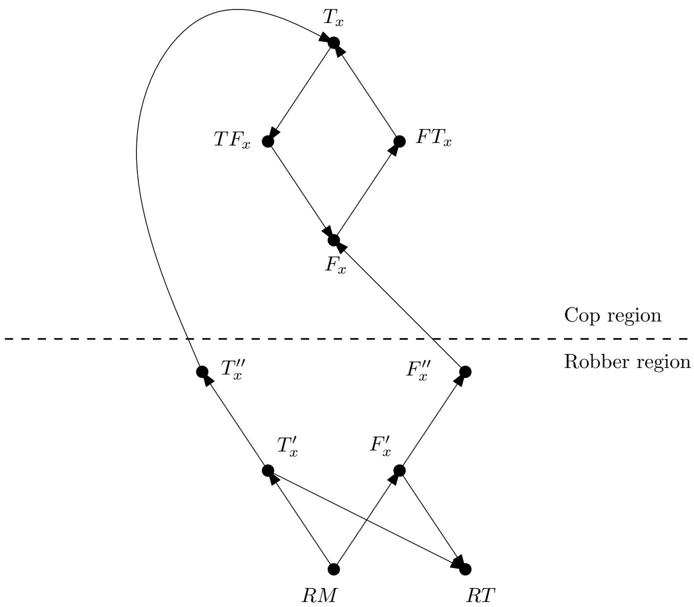

Figure 3: The Manipulator gadget M_y

The vertices $\{T'_y, F'_y, T''_y, F''_y, CM, CT\} \subset V_R$, the rest belongs to V_C.

Lemma 4 *Let us consider variable cell V_y, $y \in R$, and the corresponding Manipulator M_y. Let the robber be at the vertex CM, let the cop be either on T_y or F_y and let no other cop can access any vertex of M_y. Then the normal scenario is following: By entering the vertex T'_y (F'_y), robber forces the cop to move towards the vertex T_y (F_y). Robber then has to enter the vertex CT.*

Proof. If the cop refuses to move, the robber advances to T''_y or F''_y and easily reaches V_C before the cop can block him. $\square$

Note that this is not enough to ensure that the variable cop really reaches the opposite vertex and that only one variable cop from variable cells can move. We deal with this issue later.

When changing variables of C, we have to make sure that at most one variable is changed at a time. We ensure that by the gadget Commander (see Figure 4), built over every V_x, $x \in C$.

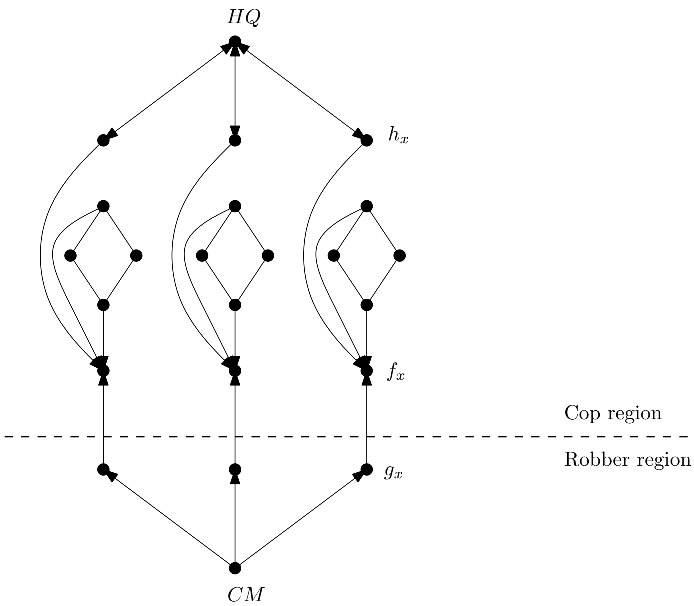

Figure 4: The Commander gadget

The vertices $\{g_x;\ x \in C\}$ and CM belong to V_R, the rest belongs to V_C. There is one cop, the "commander", sitting at the vertex HQ. From every vertex $w \in V \setminus (V_C \cup \{CM\} \cup \{g_x;\ x \in C\})$ we add the edge (w, HQ) to $\vec{G}$, thus the only time the commander can leave HQ is when the robber stands at CM. The normal scenario is following: The commander decides one variable x to be changed and moves to h_x, simultaneously the cop in the variable cell V_x starts its movement towards the opposite vertex. The commander temporarily guards the vertex f_x, which is otherwise guarded by the cop in the cell V_x. Then the robber moves (away from CM) and the commander has to return to HQ in the next move.

Lemma 5 *Let us consider the Commander gadget and the variable cells V_x for $x \in C$ with exactly one cop each, standing either on T_x or F_x. Let the robber be at the vertex CM and the cop at HQ, with the cop-player on* *move. Let no other cop can access the vertices in the Commander gadget. Then the normal scenario is that in at most one variable cell V_x, $x \in X$ the cop can change the position from T_x to F_x or vice versa.*

Proof. Only the vertex f_x is temporarily (for one move) guarded by the commander. If two variable cops starts moving, some f_y is unsecured and robber exploits it. $\square$

2.2 The gates to V_C

For every clause ϕ of F_R, there is one Robber gate gadget R_ϕ. If ϕ is satisfied by the current setting of variables, R_ϕ allows the robber to enter V_C.

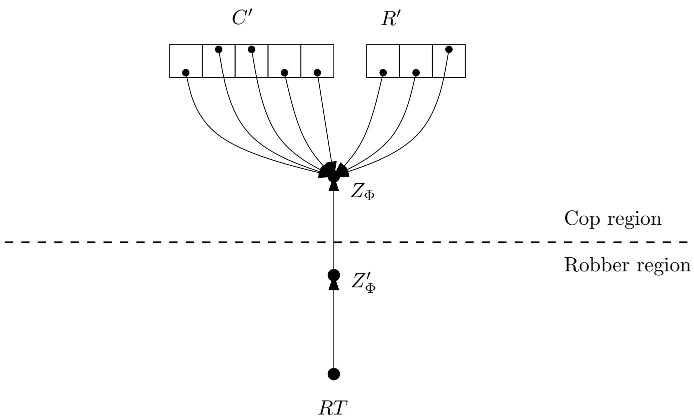

Figure 5: The Robber Gate R_ϕ

Let $\phi = (\ell_1 \& \ldots \& \ell_{12})$ where each ℓ_i is a literal. If $\ell_i = x$ then we put the edge (F_x, z_ϕ) to $\vec{G}$. If $\ell_i = \neg x$ then we put the edge (T_x, z_ϕ) to $\vec{G}$. See Figure 5 for illustration. The vertices $\{z'_\phi;\ \phi \in F_R\}$ and RT belong to V_R, the rest belongs to V_C.

Lemma 6 *Let us consider a Robber Gate R_ϕ. Let the robber stand at the vertex RT and let there be exactly one cop in each V_x, $x \in \phi$, standing either on T_x or F_x. Let no other cop can access R_ϕ. Then in the normal scenario* *robber can reach z_ϕ if and only if ϕ is satisfied under the current setting of variables (given by the positions of cops on variable cells).*

Proof. If ϕ is satisfied, no cop at the variable cells can reach z_ϕ in one step. Therefore, the robber may enter z_ϕ. On the other hand, if ϕ is not satisfied, at least one cop is one step from z_ϕ and the robber would be blocked forever if he dares to approach z_ϕ. $\square$

For every clause ψ of F_C, there is one *Cop Gate* gadget C_ψ (see Figure 6). If ψ is satisfied, C_ψ allows cops to forever block the entrance to V_C, the vertices z_ϕ from each Robber Gate R_ϕ.

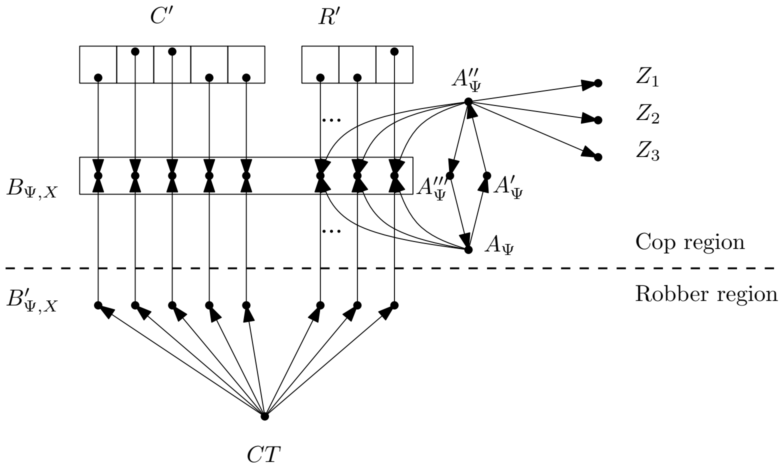

Figure 6: The Cop Gate C_ψ

Let $\psi = (\ell_1 \& \dots \& \ell_{12})$ where each ℓ_i is a literal. If $\ell_i = x$ then we put the edge $(T_x, b_{\psi,x})$ to $\vec{G}$. If $\ell_i = \neg x$ then we put the edge $(F_x, b_{\psi,x})$ to $\vec{G}$. From the vertices a_ψ and a''_ψ there is an edge to every $b_{\psi,x}$ and from a''_ψ there is an edge to every z_ϕ (from each Robber Gate R_ϕ). There is a cop at the vertex a_ψ, we call him Arnold. The vertices $\{b'_{\psi,x};\ \psi \in F_C,\ x \in \psi\}$ and CT belong to V_R, the rest belongs to V_C.

Lemma 7 *Let us consider a Cop Gate C_ψ. Let there be one cop at the vertex a_ψ (we call him Arnold) and let there be exactly one cop in each V_x,*

$x \in \psi$, standing on either T_x or F_x. Let the robber be at the vertex CT and let no other cop can access C_ψ. Then in normal scenario, Arnold is able to move to a''_ψ (and therefore block all the entrances z_ϕ forever) without permitting robber to enter C_C if and only if ψ is satisfied under the current setting of variables (given by the position of cops in the variable cells).

Proof. If ψ is satisfied, the vertices $b_{\psi,x}$, $x \in \psi$ are all guarded by the variable cops, therefore Arnold can start moving from a_ψ towards a''_ψ. If the robber meanwhile moves to some $b'_{\psi,x}$, the variable cop from V_x will intercept him by moving to $b_{\psi,x}$. On the other hand, if ψ is not satisfied, there is some $b_{\psi,x}$ unguarded by the cop from V_x. Therefore, Arnold cannot leave a_ψ, because otherwise robber would reach $b_{\psi,x}$ before Arnold or the cop from V_x could block him. □

2.3 The big picture

We further need to assure that the cops cannot move arbitrarily. This means, that the following must be the normal scenario:

1. During the "Robber Move" phase, the only cop who can move is the cop in variable cell V_x chosen by the robber when he enters Manipulator M_x. All other variable cops must stand on either T_x of F_x vertices for some variable x. The cop in V_x must reach the vertex T_x from F_x (or vice versa) in two consecutive moves.

2. During the "Robber Test" phase, no cop can move.

3. During the "Cop Move" phase, only the commander and the cop in exactly one variable cell V_x can move. The cop in V_x must reach the vertex T_x from F_x (or vice versa) in two consecutive moves.

4. During the "Cop Test" phase, no other cop than Arnold may move. Arnold may move from vertex a_ψ to a''_ψ and he must do that in two consecutive steps (and of course Arnold may do that only if the clause ψ is satisfied).

We say that we force the vertex w by the vertex set S, when for every $v \in S$ we add the oriented path $P_{v,w} = (v, p_{vw}, p'_{vw}, w)$ of length 3 to the graph $\vec{G}$. We say that we block the vertex w by the vertex set S, when for every $v \in S$ we add the edge (v, w) to $\vec{G}$.

Case 1: For every variable $x \in C \cup R$ do the following construction. Let $S_x = \{RM, RT\} \cup \{V(V_y);\ y \in C \cup R, x \neq y\}$. We force the vertices T_x and F_x by the set S_x. Let $S = \{RM\} \cup \{V(V_x);\ x \in C \cup R\}$. For each Cop Gate C_ψ, we force the vertex a_ψ by the set S. Finally, we block the vertex HQ by the set S. Observe that whenever a cop from any other V_y than given by the Manipulator M_x is not on T_y or F_y, the robber can reach V_C faster than the variable cop can block him. On the other hand, if all variable cops are in right places, the robber may never reach V_C unless being forever blocked. The same holds for Arnold on vertices a_ψ and a''_ψ. The commander cannot move trivially. If the variable cop does not use his second turn to finish his movement, the robber will exploit this by reaching V_C by a path from the vertex RT.

Case 2: Let $S = \{RT\} \cup \{z'_\phi;\ \phi \in F_R\}$ and let $F = \{T_x, F_x;\ x \in C \cup R\} \cup \{a_\psi;\ \psi \in F_C\}$. We force every $v \in F$ by the set S and we block the vertex HQ by S. Observe that in the normal scenario no cop may move.

Case 3: Let $S = \{CM\}$ and let $F = \{T_x, F_x;\ x \in R\} \cup \{a_\psi;\ \psi \in F_C\}$. We force every $v \in F$ by S. Now, in normal scenario, no variable cop from V_x, $x \in R$ may move a by Lemma 5, only commander and exactly one variable cop from V_y, $y \in C$ may move.

Case 4: Let $S = \{CT\}$ and let $F = \{T_x, F_x;\ x \in C \cup R\}$. We force every $v \in F$ by S and we block the vertex HQ by S. Observe that in normal scenario no variable cop and the commander may move. The rest follows from Lemma 7 and the fact, that a''_ψ is forced by the vertex RM.

All the construction elements so far presented prove the following corollary.

Corollary 8 *For every game $\mathcal{F} = (\tau, F_C(C,R), F_R(C,R), \alpha)$ there exist a guarding game $\mathcal{G} = (\vec{G}, V_C, c)$ with a prescribed starting positions such that player I wins $\mathcal{F}$ if and only if the cop-player wins the game $\mathcal{G}$.*

2.4 Forcing the starting position of pawns

Here we show, that we can modify our current construction so that it fully conforms to the definition of the guarding game.

Lemma 9 *Let $\mathcal{G} = (\vec{G}, V_C, c)$ be a guarding game with a prescribed starting positions $P \subseteq V_C$, $|P| = c$ of cops and a prescribed starting position $r \in V_R$ of robber. The position r must have no in-going edge. Then there exists a guarding game $\mathcal{G}' = (\vec{G}', V'_C, c')$, $\vec{G} \subseteq \vec{G}'$, $V_C \subseteq V'_C$ such that*

- *the result and complexity of both games are equal*
- *if the cops do not place their pawns exactly on P in their first move, they will lose*
- *if the robber does not place his pawn on r in his first move, the cops win.*

Proof. Consider an edge $(u, v) \in E(\vec{G})$ such that $u \in V_R$ and $v \in V_C$ (a border edge). Observe, that the outdegree of each such vertex u in our construction is exactly 1. Let $t = |\{v \in V_C;\ (u, v) \in E(\vec{G}),\ u \in V_R\}|$ be the number of vertices from V_C directly threatened by the robber region.

Let us define the graph $\vec{G}' = (V', E')$ such that $V' = V(\vec{G}) \cup \{r\} \cup T$ where $T = \{t_1, \dots, t_t\}$ and $E' = E(\vec{G}) \cup \{(r, v);\ v \in T \cup P\}$. Consider the game $\mathcal{G}' = (\vec{G}', V'_C, c')$ where $V'_C = V_C \cup T$ and $c' = c + t$. See Figure 7 for illustration.

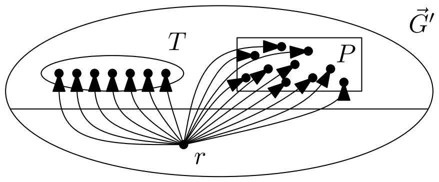

Figure 7: Forcing starting positions

Suppose that robber-player places the robber in the first move to some vertex $v \in V_R \setminus \{r\}$. Then there are t vertices in V_C directly threatened by edges going from V_R and because we have at least t cops available, the cops in the second move can occupy all these vertices and prevent the robber from entering V_C forever. So the robber must start at the vertex r. Then observe, that c cops must occupy the positions P and t cops must occupy the vertices T. If any cop does not start either on T or P, the robber wins in the next move. The cops on T remain there harmless to the end of the game. The game is frozen until robber decides to leave the vertex r. $\square$

Let us have a guarding game $\mathcal{G} = (\vec{G}, V_C, c)$ with prescribed starting positions. We add a vertex r and the edge (r, CM) to $\vec{G}$ and by the previous lemma there is an equivalent guarding game $\mathcal{G}'$, $\mathcal{F}' \subseteq \mathcal{G}$ without prescribed starting positions.

Theorem 1 is now proved.

3 The undirected case

In this section we prove Theorem 2. The idea is following: We take the same construction of directed graph $\vec{G}$ we used to prove Corollary 8. Then we forget the orientation of all edges. However, for each edge we build a gadget such that whenever the resulting undirected edge is used by cop (robber) in bad direction, the cop-player (robber-player) will lose the game.

To obtain the undirected graph G, let us take the graph $\vec{G}$, forget the orientation of the edges and subdivide each edge $e \in E(\vec{G})$, $e = (u, v)$ by three vertices (see Figure 8). We number all vertices by $0, 1, 2, 3$, where 0 belongs to former starting point of each edge $e \in E(\vec{G})$ and the newly added vertices e_1, e_2, e_3 are numbered by $1, 2, 3$ according to the orientation of e. If $u \in V_R(\vec{G})$ then $u = e_0, e_1, e_2, e_3 \in V_R(G)$, otherwise $e_0, \ldots, e_3 \in V_C(G)$.

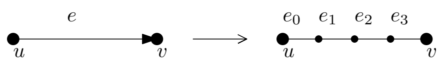

Figure 8: Subdividing directed edges

We introduce the gadget *Clock* (see Figure 9). From the vertex Ω there are edges to every vertex v such that $\{u, v\} \in E(G)$, $u \in V_R$, $v \in V_C$. Every subdivided edge is connected to Clock as in Figure 9. There is one cop (we call him Chuck) in the Clock. His purpose is following: If the robber is on vertex with number i and moves to vertex with number $j = (i + 1) \bmod 4$, the only thing he can do is to go to vertex c_j. However, whenever robber does an illegal move (to vertex with number $j = (i - 1) \bmod 4$), Chuck may enter Ω, thus winning the game. Therefore, this gadget forces the robber to pass through undirected edges only in the direction from the old graph $\vec{G}$.

Lemma 10 *Let there be exactly one cop in the Clock gadget (we call him Chuck). Let the robber be at the vertex e_i and let Chuck be at the vertex c_i. Then the normal scenario is that robber must move to a vertex e_{i+1} or f_0 for some f (ie. with number $j = (i + 1) \bmod 4$) and Chuck must move to vertex c_j.*

Proof. Suppose first that the robber moved to a vertex e_j and Chuck did not move to c_j (he may be at c'_i, $c'_{i-1 \bmod 4}$, $c_{i-1 \bmod 4}$ or stay at c_i). Then

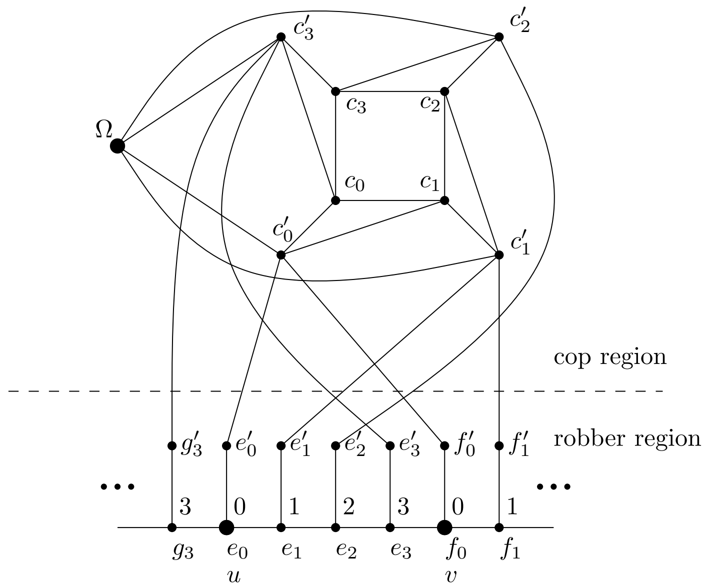

Figure 9: The Clock gadget

robber may move to e'_j and Chuck cannot prevent him from entering cop-region in the next move.

Suppose now that the robber moved to vertex e_k where $k = (i-1) \bmod 4$. The Chuck goes to c'_k, preventing robber from immediately enter cop-region. In the next move, robber may or may not enter e'_k. Is he does so, Chuck moves c'_k and guards it until the robber leaves. In both cases, Chuck moves afterwards to Ω, thus being able to block all entrances to cop-region in one move. This is because for every border edge $\{u, v\} \in E(G)$ where $u \in V_R$ and $v \in V_C$, the degree of u is exactly 1, so whenever the robber moves to u, Chuck may block the vertex v.

If the robber does not move in his turn, Chuck approaches c'_i. If robber move to e'_i, Chuck keeps guarding c'_i until robber leaves e'_i and then Chuck moves to Ω. Using the same argument as above, the game is a cop-win now.

It remains to show that robber may not enter the vertex e'_i. If he does so, Chuck will move to c'_i, preventing the robber from entering V_C, and guarding there until robber leaves e'_i. Then Chuck moves to Ω and again wins the game using the same argument as above. □

We have ensured the correct movement of robber, imitating the functionality of the graph $\vec{G}$ on an undirected graph. We still have to do similar thing for cops, as they have to respect the edge orientation as well. In our argument, every pawn has to move in his move. Because staying at one vertex may be a desired part of cop-player strategy, for every $v \in V_C(G)$ we glue a subdivided loop l^v of length 4 to v, such that the pawn will move, but in fact stay at one vertex of the original graph $\vec{G}$. The vertices of l_v are again numbered as above.

For every subdivided edge $e = \{u, v\} \in E(G)$, $u, v \in V_C(G)$, and loop l^v, we add to G the *CopDir* gadget D_e (see Figure 10). For every edge $f \in E(\vec{G})$, we add edges $\{f_i, d_{e,i}\}$, $i = 0, 1, 2, 3$, to G. We connect all vertices $d'_{e,i}$ to the vertex Ω in Clock. The vertices $d_{e,0}, \ldots, d_{e,3}$ belong to $V_R(G)$, the vertices $d'_{e,0}, \ldots, d_{e,3}$ belong to $V_C(G)$.

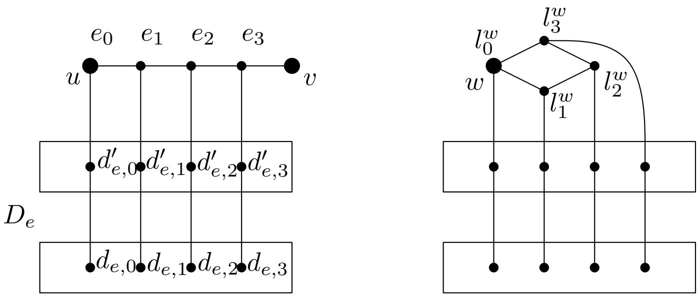

Figure 10: The CopDir gadget D_e

Lemma 11 *Let us have the CopDir gadget D_e for the subdivided edge (or loop) e with a cop on it, and let all pawns be on a vertex with number i, robber being on move. Suppose the robber moves to a vertex with number $j = (i + 1) \bmod 4$. Then the normal scenario is that the cop on the vertex e_i moves to a vertex with number j.*

Proof. Suppose the cop does not move and stays on e_i. Then the robber moves to $d_{e,j}$ and the cop is unable to block him in time. If the cop move to a vertex with number $k = (i - 1) \bmod 4$, again, the robber wins by moving to $d_{e,j}$. Thus the only thing the cop can do is to go to a vertex with number j. $\square$

Note that we have now proved that in the normal scenario, at the beginning of the robber move all pawns stand on vertices with the same number. Observe that using Lemma 10 and Lemma 11, the following corollary holds.

Corollary 12 *For every game $\mathcal{F} = (\tau, F_C(C,R), F_R(C,R), \alpha)$ there exist a guarding game $\mathcal{G} = (G, V_C, c)$, G undirected, with a prescribed starting positions such that player I wins $\mathcal{F}$ if and only if the cop-player wins the game $\mathcal{G}$.*

It remains to show, that we can force the starting position of all pawns, starting with an empty graph. Observe that the same method as used in Lemma 9 works. We have proved Theorem 2.

4 Further questions

For a guarding game $\mathcal{G} = (G, V_C, c)$, what happens if we restrict the size of strongly connected components of G? If the sizes are restricted by 1, we get DAG, for which the decision problem is $PSPACE$-complete. For unrestricted sizes we have shown that $\mathcal{G}$ is $ETIME$-complete. What is the threshold for $\mathcal{G}$ to become $ETIME$-complete from being $PSPACE$-complete?

What happens if we restrict the degrees of vertices of G? And what if we restrict the treewidth, treedepth, pathwidth, ...

5 Acknowledgements

The authors would like to thank Peter Golovach for giving a nice talk about the problem, which inspired them to work on it. We would also like to thank Jarik Nešetřil for suggesting the previous open questions.

References

[1] F. Fomin, P. Golovach, A. Hall, M. Mihalák, E. Vicari, P. Widmayer, *How to Guard a Graph?*, Algorithmica 2008.

[2] L. Stockmeyer, A. Chandra, *Provably Difficult Combinatorial Games* SIAM J. Comput. Volume 8, Issue 2, pp. 151–174 (1979).