Fast winning strategies for the attacker in eternal domination

Guillaume Bagan, Nicolas Bousquet, Nacim Oijid, Théo Pierron

Fast winning strategies for the attacker in eternal domination[1]

Guillaume Bagan[a], Nicolas Bousquet[a], Nacim Oijid[a], Théo Pierron[a]

[a]*Lab. LIRIS, UMR CNRS 5205, University of Lyon, F-69003*
University of Claude Bernard Lyon 1
43 Bd du 11 Novembre 1918, F-69622, Villeurbanne, France.

Abstract

Dominating sets in graphs are often used to model some monitoring of the graph: guards are posted on the vertices of the dominating set, and they can thus react to attacks occurring on the unguarded vertices by moving there (yielding a new set of guards, which may not be dominating anymore). A dominating set is *eternal* if it can endlessly resist to attacks.

From the attacker's perspective, if we are given a non-eternal dominating set, the question is to determine how fast can we provoke an attack that cannot be handled by a neighboring guard. We investigate this question from a computational complexity point of view, by showing that this question is PSPACE-hard, even for graph classes where finding a minimum eternal dominating set is in P.

We then complement this result by giving polynomial time algorithms for cographs and trees, and showing a connection with tree-depth for the latter. We also investigate the problem from a parameterized complexity perspective, mainly considering two parameters: the number of guards and the number of steps.

Keywords: eternal dominating set, tree-depth, PSPACE-completeness, parameterized complexity

1. Introduction

Let $G = (V, E)$ be a graph and $D \subseteq V$ be a set of vertices called *guards*. The *mobile domination game* is played by two players: Attacker and Defender. At each turn, Attacker chooses a non-guarded vertex v (we say that Attacker *attacks* v). Then, Defender has to move a guard along an edge uv incident to v (we say that Defender defends against the attack); see Figure 1 for an illustration. If she cannot move such a guard (that is v had no neighbor u in D), Attacker wins. Otherwise, the game continues with the set $D \cup \{v\} \setminus \{u\}$ of guards. If Defender can defend against any infinite sequence of attacks, the initial set of guards D is called an *eternal dominating set*. The *eternal domination number* $\gamma^\infty(G)$ is the size of the smallest eternal dominating set of G. This notion has been widely studied, see e.g. [27] for a recent survey. Many variants have been introduced in the literature, for example when all the guards can move at each attack [19]. This leads to similar notions of m-eternal dominating set and m-eternal domination number $\gamma_m^\infty(G)$.

For every graph G, we have $\alpha(G) \leq \gamma^\infty(G) \leq \theta(G)$ [11] and $\gamma^\infty(G) \leq \binom{\alpha(G)+1}{2}$ [25]. The latter inequality has been proved to be tight for an infinite class of graphs [20]. Several existing works also focus on classes of graphs for which $\gamma^\infty(G) = \theta(G)$, see e.g. [11, 1, 26].

[1]This research was supported by the ANR project P-GASE (ANR-21-CE48-0001-01).

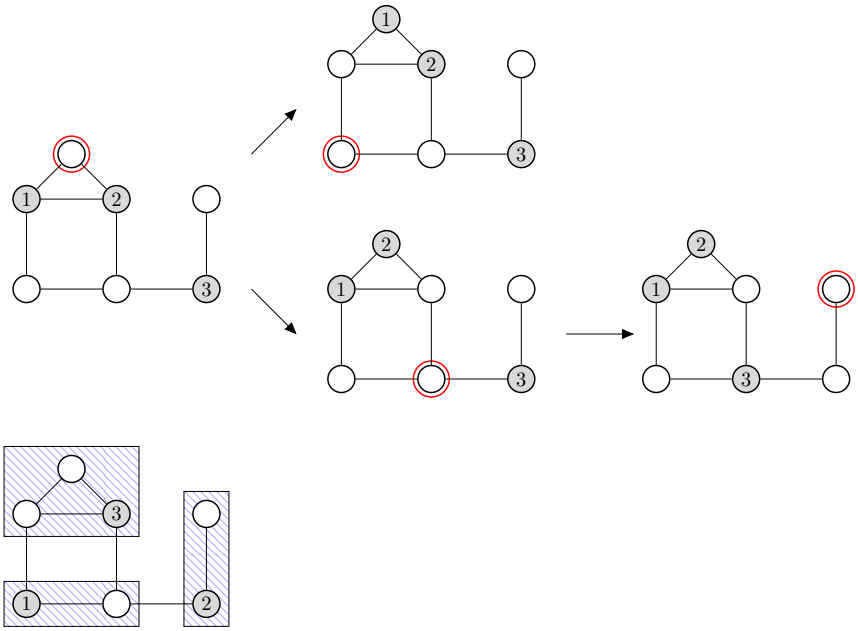

Figure 1: Above, an example of a winning strategy for Attacker in 3 turns, depending on the answer of Defender. Each number represents a guard, and the attack is circled. Below, an eternal dominating set with 3 guards. Each guard protects its own clique.

One can naturally wonder if these parameters can be easily computed. This motivates the introduction of the following problem.

ETERNAL-DOMINATING-NUMBER
Input: A graph G, an integer $k \geq 1$ **Question:** $\gamma^\infty(G) \leq k$?

This problem lies in EXP and is coNP-hard [3] but the exact complexity class this problem belongs to remains open. However, on restricted graph classes, there exist some polynomial time algorithms such as on perfect graphs. Indeed, for perfect graphs, we have $\alpha(G) = \gamma^\infty(G) = \theta(G)$, which yields a polynomial algorithm for ETERNAL-DOMINATING-NUMBER [11].

Deciding M-ETERNAL-DOMINATING-NUMBER is harder since deciding if $\gamma_m^\infty(G) \leq k$ is already NP-hard even on Hamiltonian split graphs [4]. However, polynomial time algorithms have been proposed on trees [26], unit interval graphs [8], interval graphs [30] and cactus graphs [6]. Moreover, great attention has been paid to the study of lower and upper bounds for the m-eternal dominating number in grids. See for example [28, 23]. The spy game, a generalization of m-eternal domination has been proved PSPACE-hard on directed graphs and NP-hard on (non-directed) graphs [12]. Another generalization of m-eternal domination, the guarding game, has been proved E-complete [31].

Surprisingly, the situation is quite different for the following seemingly close problem.

ETERNAL-DOMINATING-SET
Input: A graph G, a set of guards $D \subseteq V$ **Question:** Is D an eternal dominating set of G?

It is not clear that this problem is easier than the first. Indeed, there is nothing preventing

graphs to have an easy-to-find eternal dominating set of size k but such that some other such sets are much harder to determine. In particular, on perfect graphs, the complexity of Eternal-Dominating-Set is still open contrary to the complexity of Eternal-Dominating-Number.

In this paper, we are not only interested to determine whether a configuration of guards is an eternal dominating set but also to find, in the negative case, a winning strategy for Attacker that minimizes the number of steps. This question naturally arises for board games, like Chess or Go, but has also attracted a lot of attention in the combinatorial game theory literature: for the domination game [9], maker-breaker games [18, 22], the non-planarity game [2] and the spy game [12]. Recently, this problem has been studied for m-eternal domination on trees [5]. The authors prove if Defender has less than the necessary number of guards to m-eternally dominate a graph G, then Attacker can win in at most d turns where d is the diameter of G.

Informally, we say that Attacker wins in t turns on (G, D) if he has a strategy that makes Defender lose before the t-th turn, regardless of her strategy. We denote by $t_G(D)$ the minimum t such that Attacker wins in t turns on (G, D) (We postpone the formal definitions to the next section.) In particular, $t_G(D)$ is $+\infty$ if and only if D is an eternal dominating set of G.

The goal of this paper is to study the complexity of computing $t_G(D)$, that is of the following problem.

Fast-Strategy
Input: A graph $G = (V, E)$, a set of guards $D \subseteq V$, an integer $t \geq 1$
Question: is $t_G(D) \leq t$?

Our results and organization of the paper.. In Section 3, we show that Fast-Strategy is coNP-hard on bipartite and split graphs. On the positive side, we show that Eternal-Dominating-Set can be decided in polynomial time on bipartite graphs. In Section 4, we show that Fast-Strategy is PSPACE-hard even restricted to perfect graphs (recall that Eternal-Dominating-Number can be decided in polynomial time on perfect graphs). More specifically, we prove that the problem is PSPACE-complete on 2-unimodal graphs with a reduction from Unordered-CNF.

Then, we give two positive results: we prove that Fast-Strategy can be decided in polynomial time on trees (Section 5) and on cographs (Section 6). For trees, the main step of the proof consists in introducing arenas, which are special subtrees and proving that Attacker can win in at most k steps if and only if there exists an arena for which some treedepth-related parameter is at most k. We finally prove that the polynomial time algorithm for computing the treedepth on trees can be adapted for this new parameter. For cographs, we use their decomposition as disjoint unions or joins of smaller cographs and analyze the strategies of Attacker and Defender in each case to provide a recursive algorithm for Fast-Strategy.

Finally, in Section 7, we study the parameterized complexity of Fast-Strategy for several classes of graphs.

Related work. The notion of k-secure dominating set has been introduced Burger et al. [10]. In their setting, a dominating set is *k-secure* if it remains dominating after a sequence of k attacks, which looks like the problem we are considering in this paper. However, in their setting, that the sequence of attacks is known in advance (oblivious adversary) whereas, in our case, Attacker can adapt his moves according to how Defender defends the attacks (adaptative adversary).

2. Preliminaries

For an integer $n > 0$, $[n]$ represents the set $\{1, 2, \ldots, n\}$. All graphs considered in this paper are finite, loopless, and simple. Let $G = (V, E)$ be a graph. Given a vertex $v \in V$, $N(v)$ denotes the *neighborhood* of v, i.e., the set $\{y \in V : vy \in E\}$. The *degree* $d(v)$ of v is $|N(v)|$. A set $S \subseteq V$ is an *independent set* of G if there is no edge uv for every u, v in S. The *independence number* $\alpha(G)$ is the size of a largest independent set of G. A *clique* is a subset of pairwise adjacent vertices. The *clique covering number* $\theta(G)$ is the minimum number of cliques in which V can be partitioned. A *vertex cover* of G is a set $S \subseteq V$ such that every edge e in E has an endpoint in S. A *matching* of G is a set of edges of G that pairwise do not share an endpoint. A graph G is *perfect* if $\alpha(H) = \theta(H)$ for every induced subgraph H of G.

A *rooted tree* T is a tree with a distinguished vertex r called the root. Consider the orientation of T such that every vertex is reachable from r. u is a *child* of v if there is an arc vu in this orientation and u is a *descendant* of v if there is an oriented path from v to u along this orientation. A rooted subtree of T is a subtree of T induced by a node and all its descendants. The *height* of T is the number of vertices in a longest directed path starting at r.

A *td-decomposition* of a connected graph $G = (V, E)$ is a rooted tree T with set of vertices V such that u is a descendant of v or v is a descendant of u for every $uv \in E$. The *tree-depth* of G is the minimum height of a td-decomposition of G (see Figure 2).

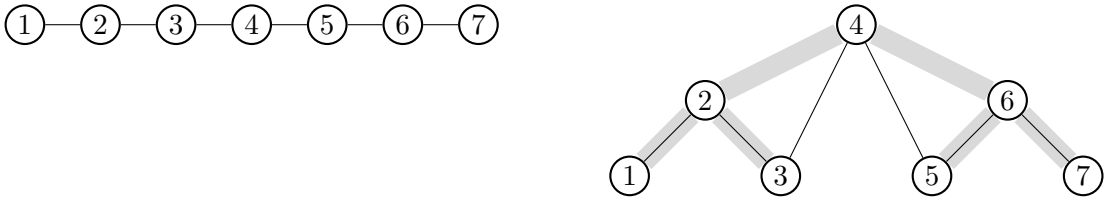

Figure 2: On the left, the graph P_7 of tree-depth 3. On the right, an optimal td-decomposition of P_7

Let $G = (V, E)$ be a graph and $D \subseteq V$ be a set of g vertices called *guards*. In the eternal domination game, a *strategy for Attacker* is a function mapping each set S of g vertices to a vertex of $V \setminus S$. A *strategy for Defender* is a partial function mapping pairs (S, v) where S is a set of g vertices and $v \in V \setminus S$ to some vertex of $N(v) \cap S$ if it exists.

Given two strategies f, g for Attacker and Defender, we say that Defender can *resist t attacks* on (G, D) using (f, g) if either $t = 0$, or $g(D, f(D))$ is defined, and she can resist $t - 1$ turns on $(G, D \cup \{g(D, f(D))\} \setminus \{f(D)\})$ using (f, g). We extend this notion: Defender can resist t attacks on (G, D) when Attacker plays according to f if she has a strategy g so that the previous condition is satisfied.

We are interested in the *fastest way* for Attacker to win (if he can), that is the minimum t such that Attacker has a strategy f such that Defender cannot resist t attacks on (G, D) when Attacker plays according to f. We denote this number t by $t_G(D)$. Note that $t_G(D)$ might be $+\infty$ if Defender can eternally answer to Attacker's move, *i.e.* if Attacker has no winning strategy.

3. Bipartite and split graphs

The goal of this section is to study the complexity of the problems Eternal-Dominating-Set and Fast-Strategy on bipartite graphs. Recall that bipartite graphs are perfect, hence

determining the smallest eternal dominating set can be done in polynomial time [19]. We extend this result to Eternal-Dominating-Set.

Theorem 1. Eternal-Dominating-Set *is in* PTime *on bipartite graphs.*

On the opposite, we show that determining the smallest number of moves in a winning strategy is significantly harder, as summarized by the following statement.

Theorem 2. Fast-Strategy *is in* PSPACE*and* coNP*-hard on bipartite graphs.*

Both these statements are based on the following lemma, which gives some insights about the shape of winning strategies for the eternal domination game on bipartite graphs.

Lemma 3. *Let G be a bipartite graph and D be a set of guards. If Attacker wins in t turns on (G, D), then he can win in at most t turns by always attacking the same side of the bipartition.*

Proof. We prove this result by induction on t. The result holds when $t = 1$. Hence, we can assume that $t > 1$.

Denote by (A, B) a bipartition of G. Let v be the first move of Attacker in a winning strategy. By symmetry, we can assume that $v \in A$. Let $u \in N(v) \cap D$ be the answer of Defender. Let us denote by $D' = (D \setminus \{u\}) \cup \{v\}$ the new set of guards.

By assumption on v, Attacker has a winning strategy in $t - 1$ steps on (G, D'). By induction, there exists such a strategy consisting in playing only on A or only on B. In the first case, the conclusion follows. So we may assume that Attacker has a winning strategy in $t-1$ turns on (G, D') by playing only in B.

Assume by contradiction that, in (G, D), Defender can resist to $t - 1$ attacks on B. Then she can also resist to $t-1$ attacks on B in (G, D') by answering v if Attacker plays on u, and answering with a guard from $D \cap A$ otherwise, following her strategy on (G, D).

This is a contradiction, hence Attacker can win in $t - 1$ turns on (G, D) by playing only on B, which concludes. □

An important consequence of this result is the following kernelization statement.

Lemma 4. *Let G be a bipartite graph and D be a set of guards. Then, one can compute in polynomial time a bipartite graph G' on the same set of vertices such that $t_G(D) = t_{G'}(D)$ where D and $V(G') \setminus D$ form a bipartition of $V(G')$.*

Proof. Let G' be the graph obtained from G by removing the edges between two guarded vertices, and between two unguarded vertices. G' is a bipartite subgraph of G with bipartition $(D, V(G)\setminus D)$.

It remains to show that $t_G(D) = t_{G'}(D)$. Since G' is a subgraph of G, any winning strategy of Attacker on (G, D) is also winning in (G', D), hence $t_G(D) \geqslant t_{G'}(D)$. For the converse direction, let $A \cup B$ be a bipartition of G. In particular, it is also a bipartition of G', and by Lemma 3 there exists a winning strategy for Attacker in (G', D) in $t_{G'}(D)$ turns which consists in only playing in the same side of the bipartition, say B. We claim that following this strategy, Attacker wins in $t_{G'}(D)$ turns on (G, D). More precisely, we claim that whenever Attacker attacks a vertex, Defender has the same possible answers in G and G'.

Since Attacker never plays in A, guards are only moved from A to B. In particular, guards on B cannot move and unguarded vertices in A stay unguarded during the whole game. Therefore, if Attacker attacks $b \in B$, then $b \notin D$. Moreover, if $a \in A$ is a guarded neighbor of b in G then $a \in D$, hence $ab \in E(G')$.

Therefore, Attacker wins in $t_{G'}(D)$ turns on (G, D), hence $t_G(D) = t_{G'}(D)$. □

We are now ready to prove the main results of this section.

Proof of Theorem 1. Let $G = ((A, B), E)$ be a bipartite graph and D be a set of guards. By Lemma 4, we can modify the instance in order to assume that $B = D$. We claim that D is an eternal dominating set of G if and only if G admits a matching saturating A (i.e. such that all the vertices of A are matched). Proving this claim is enough to conclude since this criterion can be tested in polynomial time.

Let us now prove the claim. If G has a matching M saturating A then the edges of M and the unmatched vertices in B form a clique cover of G where each clique contains a guarded vertex. Hence, G can be eternally dominated (Defender always defend an attack with a guard on the same clique of the clique cover).

For the converse direction, assume there is no matching saturating A. By Hall's marriage theorem (see e.g.[15]), there is a set of vertices $S \subseteq A$ such that $|N(S)| < |S|$. In particular, Attacker wins in at most $|N(S)| + 1$ turns by successively attacking vertices in S which can only defended by vertices of $N(S)$. □

The rest of this section is devoted to the proof of Theorem 2. The fact that is belongs to PSPACE is actually an easy consequence of Lemma 4, since it gives a polynomial bound (namely the number of guards plus one) on the number of minimum number of turns of a winning strategy for Attacker.

It remains to show that FAST-STRATEGY is coNP-hard on bipartite graphs. To this end, we reduce the problem INDEPENDENT-SET to co-FAST-STRATEGY. Let G be a graph and k be an integer such that (G, k) is an instance of INDEPENDENT-SET. We assume that vertices of G are labeled by integers from $[n]$. We build an instance (G', D, t) of co-FAST-STRATEGY as follows (see Figure 3):

1. We create in G' a graph $K_{k,n}$, and denote by $U = \{u_1, \ldots, u_k\}$ and $V = \{v_1, \ldots, v_n\}$ the vertices of each part.
2. For each edge $e \in E(G)$, we create a new complete bipartite graph $K_{k+1,k}$, with bipartition S_e, T_e.
3. For each edge $e = ij \in E(G)$, we connect each vertex of S_e to v_i and v_j.
4. The set D of guards contains V and all the T_e's.

Let $t = 2k + 1$. The graph G' is bipartite and can be constructed in polynomial time. Now, Theorem 2 boils down to prove the following.

Lemma 5. *The graph G has no independent set of size k if and only if Attacker wins in at most t turns on (G', D).*

Proof. If G has no independent set of size k, then, Attacker can win as follows. Attacker first attack all the vertices of U one by one. Defender must choose k vertices $v_{j_1}, \ldots, v_{j_k}$ in V to defend these attacks. Since G has no independent set of size k, Defender must have chosen two vertices v_i, v_j of V that are connected by an edge $e = v_iv_j$ in G. Attacker then attacks the $k + 1$ vertices of S_e one after another. Since both v_i and v_j have been slid on U, Defender can only defend these attacks by sliding guards from T_e to S_e. Since S_e has size $k + 1$ and T_e only has size k, Attacker wins in at most $t = 2k + 1$ turns.

For the converse direction, assume that G has an independent set $S = \{j_1, \ldots, j_k\}$ of size k. By Lemma 3, we can assume that Attacker only plays on U or vertices in S_e (since the other side of the

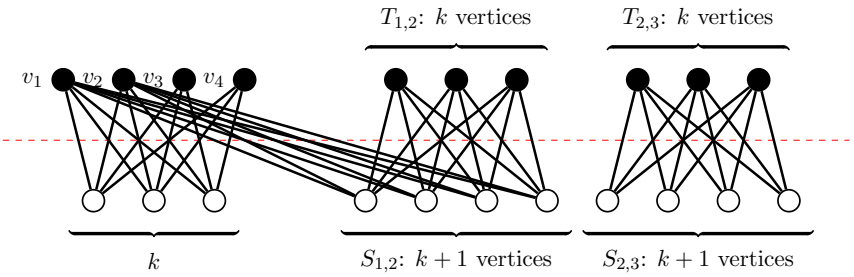

Figure 3: A partial representation of the reduction from (G, k) where G is a graph with 4 vertices $\{v_1, v_2, v_3, v_4\}$ and two edges v_1v_2 and v_2v_3 and $k = 3$. Every vertex in $S_{2,3}$ is connected to v_2 and v_3. The vertices above the dashed line are guarded. The length of the game is $t = 2k + 1 = 7$.

bipartition only contains guarded vertices). We claim that Defender can defend against Attacker during t turns with the following strategy:

1. If Attacker plays on u_i, Defender moves the guard v_{j_i}.
2. If Attacked attacks a vertex in S_e and that at least one other vertex of S_e has not been attacked, then Defender moves a guard from a vertex of T_e.
3. If Attacker and attacks a vertex in S_e with $e = v_iv_j$ and all the other vertices of S_e have been attacked then Defender moves a guard from $\{v_i, v_j\} \setminus S$ to defend the attack if such a vertex exists.

This strategy indeed permits to defend against all the attacks on U. So if Defender cannot defend, it is at Step 3. Since S is an independent set, $\{v_i, v_j\} \setminus S$ is not empty for every edge and then Defender can defend against the attack of the last vertex of S_e at least once. So Attacker cannot win in t turns on (G', D). □

This completes the proof of Theorem 2. We can actually easily adapt this reduction to obtain the same result on split graphs.

Theorem 6. Fast-Strategy *is* coNP*-hard on split graphs.*

To do so, we define the graph G'' as the graph obtained from G' by adding a new vertex s and all edges between s, V and the vertices of each T_e. Fix also $D' = D \cup \{s\}$. Now we get the following analogue of Lemma 5.

Lemma 7. *G has no independent set of size k if and only if Attacker wins in at most t turns on (G'', D').*

Proof. If G has no independent set of size k, then Attacker wins in t turns on (G'', D') using the same strategy as in Lemma 5.

Conversely, if G has an independent set of size k, then Defender can also resist t turns with a very similar strategy to Lemma 5, except that if Attacker plays on $V \cup \{s\}$ or some T_e, she then answers by moving the guard initially posted on s. □

4. Perfect graphs

Let us now prove that Fast-Strategy is PSpace-hard on 2-*unipolar graphs*. A graph is *unipolar* if its vertices can be bipartitioned into a clique V_1 and a disjoint union of cliques V_2. It is moreover *k-unipolar* if the cliques in V_2 have size at most k. Since unipolar graphs are perfect [16], Fast-Strategy also is PSPACE-hard on perfect graphs. Moreover, 2-unipolar are weakly chordal, hence the problem also is actually PSPACE-complete on weakly chordal graphs.

Theorem 8. Fast-Strategy *is* PSPACE*-hard on* 2*-unipolar graphs.*

The rest of this section is devoted to prove Theorem 8. We make a reduction from Unordered-CNF. An instance of Unordered-CNF consists of two sets of variables X, Y with $|X| = |Y|$ and a CNF formula φ with variables $X \cup Y$. The output is true if and only if Satisfier has a winning strategy in the following game: Two players, called Falsifier and Satisfier, successively choose a variable (Falsifier in X and Satisfier in Y) and assigns it a truth value, until no variable remains. Satisfier wins if the assignment satisfies the formula φ. Schaefer proves that Unordered-CNF is PSPACE-complete [32].

Our reduction. Let (X, Y, φ) be an instance of Unordered-CNF and let $k = |X| = |Y|$ and $M = 8k^2$. Without loss of generality, we can assume that $k \geqslant 2$ and every clause of φ contains at least one variable from Y. We create an instance (G_φ, D_φ) of Fast-Strategy as follows (see Figure 4 for an illustration). Let us denote by $\overline{X}$ (resp. $\overline{Y}$) the set $\{\overline{x}, x \in X\}$ (resp. $\{\overline{y}, y \in Y\}$).

- For each $x \in X$, we create two adjacent vertices u_x and $u_{\bar{x}}$ in G_φ.

  Let us denote by U the union of all these vertices.

- For each $x \in X$ and $y \in Y$, we create four vertices $v_{x,y}, v_{x,\bar{y}}, v_{\bar{x},y}$ and $v_{\bar{x},\bar{y}}$, that are all in D_φ. For every $y \in Y$, we denote by $V_y^* = \{v_{a,b} \text{ with } b \in \{y, \overline{y}\}\}$. Note that V_y^* has size $4k$.

- For every $a \in X \cup \overline{X}$ and $b \in Y \cup \overline{Y}$, we create an edge between u_a and $v_{a,b}$.

- For each variable $y \in Y$, we create two new sets of vertices V_y, V_y' of size respectively M and $M - 4k + 1$. We add a complete bipartite graph between V_y and V_y' called the *variable checker of* y.

  We add all the possible edges between V_y and V_y^* and we put guards on all the vertices of V_y'.

- For each clause $C_i = \bigvee_j \ell_{i,j}$, let L_i be the set of vertices $v_{\ell,\ell'}$ where neither ℓ nor ℓ' appears in C_i.

  We create two new sets W_i, W_i' of size respectively M and $M - |L_i| + k - 1$ and all the edges between W_i and W_i'. This bipartite graph is called the *clause checker* of C_i.

  We finally connect all the vertices of W_i to all vertices of L_i, and put a guard on each vertex of W_i'.

- We create a last guarded vertex s.

- We add an edge between every pair of guarded vertices.

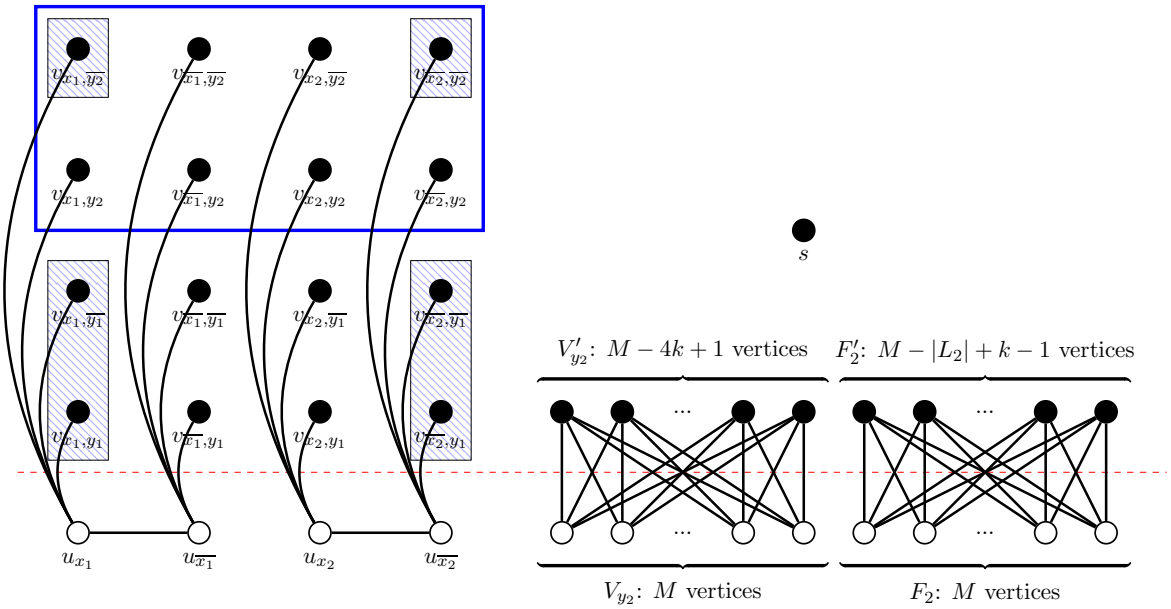

Figure 4: A partial representation of the reduction from the formula $(x_1 \vee \overline{y_1}) \wedge (\overline{x_1} \vee x_2 \vee y_2)$. Only two checkers are represented, one for the variable y_2 and the other for the clause $C_2 = \overline{x_1} \vee x_2 \vee y_2$. The vertices above the dashed line are guarded and forms a clique. All vertices of V_{y_2} are connected to the 8 vertices in the thick rectangle. All vertices of F_2 are connected to the vertices in the hatched zones. The length of the game is $t = k + M = 34$.

The graph G_φ is 2-unipolar since removing the clique induced by the guarded vertices leaves isolated vertices or isolated edges. Moreover, the construction can be carried out in polynomial time. It remains to show that the reduction is correct. We split this result in two lemmas.

Lemma 9. *If Falsifier wins* UNORDERED-CNF *on φ, then Attacker wins the mobile domination game on (G_φ, D_φ) in at most $k + M$ turns.*

Proof. Assume that Falsifier has a winning strategy on (X, Y, φ). We construct a winning strategy for Attacker in $k + M$ turns on (G_φ, D_φ).

The strategy of Attacker will consist in only attacking vertices of U until he has a simple winning strategy. One of them is the following:

- If at most k attacks were already performed and if there exists y such that V_y^* contains at most $4k - 2$ guards then Attacker successively attacks the vertices of V_y.

Since V'_y has size $M-4k+1$ and V_y has size M, V_y is defended by at most $M-4k+1+(4k-2) < M$ guards. And then Attacker has a winning strategy in at most M rounds. So we can assume that, during the k first steps, if Attacker chooses a vertex of U, Defender should move at most one vertex in each set V_y^*.

During the first k turns, Attacker will either play the strategy described above or attack a vertex u in $\{u_x, u_{\overline{x}}\}$ for some $x \in X$. By construction of the instance, Defender can only defend such an attack by moving a vertex from some vertex $v_{\ell,\ell'}$ to u with $\ell \in \{x, \bar{x}\}$.

Let us denote respectively by u_{a_i} and $v_{a'_i,b_i}$ the attacked vertex at step i and the vertex where the guard defending the attack was posted. Note that because of the strategy explained above, we

can assume that, for every $i, j \leq k$, $b_i \neq b_j$ and $b_i \neq \overline{b_j}$ (otherwise, two vertices of some V_y^* were slid to U and then V_y^* contains at most $4k-2$ guards). So the sequence of the j first rounds provides a partial truth assignment where j variables of X are given a truth value as well as j variables of Y.

So the strategy of Attacker during the first k rounds consists either in playing the simple strategy if he can or following the strategy of Falsifier in (X, Y, ϕ) by playing on u_x whenever Falsifier sets the variable x to true in the partial truth assignment described above.

Since Falsifier wins on (X, Y, φ), the resulting truth assignment does not satisfy some clause C_i of φ. In particular, during each of the k first turns, Defender moved a guard from L_i to some u_ℓ, so only $|L_i| - k$ guards remain on L_i. Now the M vertices of W_i are defended by at most $|L_i| - k + |W_i'| = M - 1$ guards, hence Attacker can win in M turns by successively attacking each vertex of W_i.

□

Lemma 10. *If Satisfier wins* Unordered-CNF *on φ, then Defender can resist $k + M$ attacks in the mobile domination game on (G_φ, D_φ).*

Proof. Assume that Satisfier has a winning strategy on φ. We call the guard posted initially on s the *special guard*. The strategy of Defender on (G_φ, D_φ) consists in applying the following rules:

1. If Attacker attacks a vertex v that was guarded, then Defender moves the special guard to v. Defender never moves the special guard outside $N[s]$ and then the move is possible since the set of guarded vertices form a clique[2].
2. If Attacker attacks an (initially unguarded) vertex of a (variable or clause) checker, Defender defends, if she can, by moving a vertex initially guarded in the same checker (hence not the special guard). If she cannot, we say that Attacker *engaged* this checker, and Defender defends by moving a guard from V_y^* (for the variable checker (V_y, V_y')) or L_i (for the clause checker (W_i, W_i')).
   Note that, by construction and since guards in checkers will only be used to defend vertices in the checker, Attacker has to attack at least $M - 4k + 1$ times a variable checker and at least $M - |L_i| + k - 1 \geqslant M - 4k^2$ times a clause checker to engage the checker. Therefore, Attacker can engage at most one checker in $M + k$ turns, by our choice of M. Note that a checker can be engaged several times.
3. If Attacker attacks the vertex u_ℓ for some literal ℓ and a token is already on $u_{\bar{\ell}}$, then Defender moves the guard from $u_{\bar{\ell}}$ to u_ℓ.
   In the rest of the strategy, if we move a guard on a vertex of the pair $\{u_\ell, u_{\bar{\ell}}\}$, the guard will stay on that pair of vertices (note that by 1., it cannot be the special guard).
4. Assume now Attacker attacks the vertex u_ℓ for some literal ℓ and no token is already on $u_{\bar{\ell}}$ and no variable checker has been engaged. In this case, let ℓ' be the literal set to true by Satisfier when Falsifier sets ℓ to true (this is a valid move in the Unordered-CNF game by the previous rule). Observe that the guard initially on $v_{\ell,\ell'}$ did not move before, hence Defender defends by moving it.
5. Finally, assume that Attacker attacks the vertex u_ℓ for some literal ℓ and no token is already on $u_{\bar{\ell}}$ and a checker has been engaged. Then Defender answers by moving a non-special guard from some vertex $v_{\ell,\ell'}$ not adjacent to the engaged checker. In particular, this is always

[2]One can easily remark that this strategy is compatible with the remaining rules.

possible if the engaged checker is a variable checker since $k \geqslant 2$, or if it is a clause checker since each clause contains a literal from Y.

Observe that the moves played by the second-to-last rule yield a partial truth assignment for the variables of $X \cup Y$. Moreover, since Satisfier wins, one can extend this assignment so that φ becomes true.

Since, by construction, the vertices initially guarded and the vertices u_ℓ for every ℓ can always be defended during the $M + k$ steps, it simply remains to show that Defender can always answer attacks in an engaged checker.

Assume that Attacker engaged the variable checker (V_y, V'_y) for the $(r+1)$-th time. Note that after engaging the checker for the first time, no guard from V_y^* will leave $V_y^* \cup V_y \cup V'_y$ by Rule 5. Moreover, before the checker is engaged, Rule 4 ensures that at most one non-special guard from V_y^* has been moved (to some u_ℓ). Each time the checker is engaged, one non-special guard from V_y^* is moved to V_y. Therefore, in the current situation, there are at least $4k - r - 1$ non-special guards in V_y^*. Note that the number of unguarded vertices in V_y is $|V_y| - |V'_y| - r = 4k - 1 - r$. Since there is an unguarded vertex in V_y (because Attacker played on it), we have $4k - r - 1 \geqslant 1$, hence there is a non-special guard on V_y^* that Defender can move.

A similar argument holds when the engaged checker is a clause checker (W_i, W'_i). Indeed, assume that at some point k non-special guards of L_i have been moved to some u_ℓ. This means that Rule 5 never applied and Rule 4 was applied k times. Moreover, all the literals set to true by Rule 4 do not appear in C_i, hence C_i (and furthermore φ) is not satisfied, a contradiction since Defender played following Satisfier's winning strategy on (X, Y, φ). Therefore, at most $k - 1$ non-special guards of L_i have been moved to some u_ℓ, so at least $|L_i| - k + 1$ remain at any time. Therefore, Defender can answer attacks on W_i by playing a vertex in L_i every time there is no guard in W'_i.

□

5. Trees

The goal of this section consists in proving the following theorem:

Theorem 11. Fast-Strategy *can be solved in polynomial time on trees.*

In a first part, given G and D, we will introduce the concept of arenas, and we will show that, when G is a tree, $t_G(D)$ is related to the maximum possible treedepth of an arena of G. We will then prove that this parameter is exactly the parameter of interest to decide Fast-Strategy on trees. We then recall an algorithm from Iyer et al. [24] to compute the treedepth of a tree, and adapt it in a last part to compute $t_G(D)$ in polynomial time.

5.1. The right parameter

The main concepts introduced in this section are arenas.

Definition 12. *Let T be a tree and $D \subseteq V(T)$ be a set of guards. An arena of (T, D) is a subtree X of T such that:*

1. *$V(X) \cap D$ and $V(X) \setminus D$ both are independent;*
2. *the guarded vertices in X have degree exactly 2 in X (in particular no leaf is guarded);*
3. *there is no vertex in $V(X) \setminus D$ with a neighbor in $D \setminus V(X)$.*

For the ease of notation, we identify a subtree X with its set of vertices, so that $X \cap D$ denotes the set of guarded vertices in X.

Equivalently (even if we will not prove it since we will not use it), an arena is a set satisfying the following properties:

- $|D \cap X| < \alpha(T[X])$;
- there is no unguarded vertex in X with a guarded neighbor outside X (that is, there is no edge between $X \setminus D$ and $D \setminus X$);
- X is minimal for these properties.

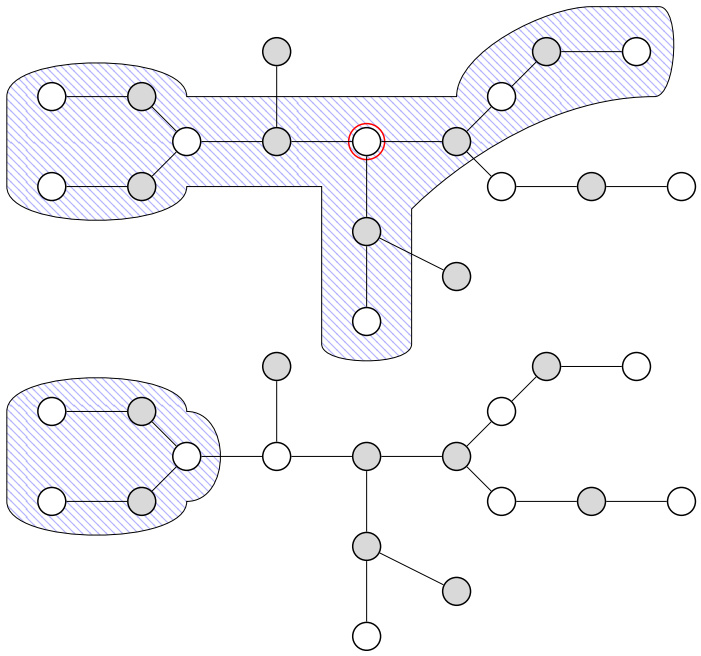

Figure 5: Above, a tree with an arena of contracted tree-depth 3. Each grayed vertex is guarded, and the attack is circled. Below, the same tree after the Defender has moved the guard from left to the attack. A smaller arena of contracted tree-depth 2 is created (its contracted tree is a path on three vertices).

Definition 13. *Let T be a tree, $D \subset V(T)$ and X an arena of (T, D). The* contracted tree *of X is obtained from X by replacing each vertex from $V(X) \cap D$ by an edge between its neighbors. The* contracted treedepth *of X, denoted by* $\mathrm{ctd}(X)$*, is the treedepth of this contracted tree.*

In the rest of this part, we will prove the following:

Theorem 14. *Given a tree T and a set of guards D, $t_T(D)$ is the minimum value of* $\mathrm{ctd}(X)$ *over all arenas X of (T, D).*

In particular, Theorem 14 ensures that $t_T(D) = +\infty$ if and only if (T, D) has no arena.

Let T be a tree and D be a set of guards. We split the proof of Theorem 14 into two independent lemmas.

Lemma 15. *If (T, D) has an arena X of contracted tree-depth t, then $t_T(D) \leq t$.*

Proof. We proceed by induction on t. If $t = 1$, then, since in the construction of the contracted tree, we only remove vertices of degree 2, the contracted tree should initially contain only one vertex v. The vertex v should be unguarded by (2) and has no guarded neighbor in T by (3). Hence, Attacker wins in one turn by attacking v.

Assume that $t > 1$, and take a td-decomposition T^* of $T[X]$ of depth t. The strategy of the Attacker is as follows: he attacks the root v of T^*. Denote by u the vertex of D moved by Defender (if Defender cannot defend the conclusion follows). Note that $u \in D$ and then u has degree 2 in $T[X]$. Let w be the other neighbor of u in X.

Consider X' the component of $T[X] - uw$ containing w. Observe that X' is an arena of $(T, D \cup \{v\} \setminus \{u\})$. Indeed, X' is a subset of X which contains neither u nor v so (1) holds. Since w is at even distance from the root, (2) holds. Finally, by (3) and the fact that u has been moved on v ensures that (3) holds. Moreover, the contracted tree of X' is a subtree of $T^* \setminus \{v\}$, therefore it has depth at most $t - 1$ and $\mathrm{ctd}(X') \leqslant t - 1$. By induction, Attacker wins on $(T, D \cup \{v\} \setminus \{u\})$ in at most $t - 1$ turns and thus, wins on (T, D) in at most t turns. □

In particular, Attacker wins the eternal domination game if and only if there is an arena.

Lemma 16. *If $t_T(D) = t < +\infty$, then there exists an arena of contracted tree-depth at most t.*

Proof. We proceed again by induction. If $t = 1$, then there exists an unguarded vertex v without neighbor in D. In particular, it is an arena of contracted tree-depth 1.

Assume that $t > 1$, and let v be the first attack of Attacker in a shortest winning strategy. Let u be a guarded neighbor of v. Assume that Defender moves the guard of u to answer the attack on v. By induction, after the move, Attacker wins in $t - 1$ turns at most, hence there exists an arena X_u of contracted tree-depth at most $t - 1$ in (T, D') (where $D' = D \cup \{v\} \setminus \{u\}$).

Assume that $v \in X_u$. By definition v has degree 2 in X_u, hence it has a neighbor $w \neq u$. The subtree containing w in $X_u \setminus \{vw\}$ is an arena in (T, D) of contracted tree-depth at most $t - 1$, contradicting $t_T(D) = t$ by Lemma 15. Therefore, $v \notin X_u$ and by definition of arenas, $u \notin X_u$ either.

Note that X_u must contain an unguarded neighbor of u, otherwise it is an arena in (T, D), again a contradiction with $t_T(D) = t$.

Now consider the set $X = \{v\} \cup \bigcup_{u \in N(v) \cap D} (X_u \cup \{u\})$. Observe that it is an arena of contracted tree-depth at most t (since removing v and its guarded neighbors from X yields components of contracted tree-depth at most $t - 1$). □

This concludes the proof of Theorem 14. Note in particular that, since the treedepth of a tree is logarithmic in its number of vertices, if $t_T(D)$ is finite, then it is at most logarithmic in $|V(T)|$.

It remains to show that we can compute the minimum contracted treedepth of an arena in polynomial time. This is the goal of the rest of this section.

5.2. Ranking lists

Our algorithm is built upon the algorithm for computing the treedepth of a tree by Iyer et al. [24]. We start by presenting (a rephrased version of) their algorithm, and stating some of their results about its correctness.

Definition 17. *Given two non-increasing lists of integers L_1 and L_2, we define the merged list $L_1 \oplus L_2$ as the concatenation of L_1 and L_2 sorted by decreasing order.*

Definition 18. *[24] Let T be a rooted tree. The* ranking list *of T is*

$$\mathrm{rl}(T) = \begin{cases} \varnothing & \text{if } T \text{ is empty} \\ (\mathrm{td}(T)) \oplus \mathrm{rl}(T - T') & \text{otherwise} \end{cases}$$

where T' is the unique inclusion-wise minimal rooted subtree of T with $\mathrm{td}(T') = \mathrm{td}(T)$.

The uniqueness of T' is guaranteed by the following lemma, since two rooted subtrees are either disjoint or contained one in the other.

Lemma 19. *[24] Let T be a tree and V_1, V_2 be two disjoint subsets of $V(T)$. If $\mathrm{td}(T[V_1]) = \mathrm{td}(T[V_2]) = d$, then $\mathrm{td}(T) > d$.*

The algorithm from Iyer et al. actually computes the ranking list of a given rooted tree T in polynomial time, and then outputs its first element (namely, $\mathrm{td}(T)$). More precisely, they give a procedure RankRoot that computes the ranking list of a tree T from the ranking lists of the subtrees rooted at children of its root. We rephrase their algorithm in a way more suited to our purposes, using two operations: the merge $\oplus$ of two sorted lists, and the closure of a list.

Definition 20. *Given a non-increasing list L, we define its closure $\uparrow L$ as follows. Let L_s the longest suffix of L which is either empty or satisfies $L_s \geq_{lex} (k, k-1, \ldots, 1)$ where k is the first element of L_s. Then, $\uparrow L$ is build from L by replacing L_s by the 1-element list $(k+1)$ (or by (1) if L_s is empty).*

Example 21. $\uparrow(7, 6, 4, 3, 2, 2) = (7, 6, 5)$ *since the largest suffix we can find is* $(4, 3, 2, 2)$. *Moreover,* $\uparrow(4, 3, 2) = (4, 3, 2, 1)$.

The following lemma states the recurrence relation used in [24] to compute the ranking list of a given tree.

Lemma 22. *[24] Let T be a tree rooted in r and $T_1, \ldots, T_k$ the subtrees of T rooted at the children of r. Then* $\mathrm{rl}(T) = \uparrow(\mathrm{rl}(T_1) \oplus \cdots \oplus \mathrm{rl}(T_m))$.

5.3. Computing $t_T(D)$

The main part of our algorithm is a recursive procedure that takes as input a tree T rooted at an unguarded vertex and determines the smallest contracted treedepth of an arena containing its root (if such an arena exists). Following the steps of Iyer et al., computing contracted tree-depth is not sufficient for the recursion to carry out. We actually compute the *contracted ranking list* of an arena, that is the ranking list of its contracted tree.

To compute $t_T(D)$, we just have to apply this procedure to all possible ways of rooting T at an unguarded vertex, and output the minimum contracted treedepth we encountered which can indeed be done in polynomial time.

First observe that we can simplify the tree T by trimming some parts that will never be in any arena. Namely, we can remove all the edges between two guarded (resp. unguarded) vertices. Moreover we can also remove every guarded leaf of T together with its neighbor. Note that this may disconnect T, but since arenas are connected, we can just handle each connected component

separately. Without loss of generality, we may thus assume that no such operation can be applied, that is T is rooted at an unguarded vertex r, the sets D and $V(T) \setminus D$ are both independent, and all leaves of T are unguarded. In that case, we say that T is *nice*.

Denote by $u_1, \dots, u_k$ the children of r, which must thus be guarded. Moreover, for $1 \leqslant i \leqslant k$, let $v_{i,1}, \dots, v_{i,\ell_i}$ be the children of u_i (which are unguarded). Note that any arena of T containing r must contain $u_1, \dots, u_k$, and for every $1 \leqslant i \leqslant k$, exactly one $v_{i,j}$. Moreover, the restriction of X to the subtree $T_{i,j}$ of T rooted at $v_{i,j}$ is an arena of this subtree containing its root.

Finding an arena of smallest contracted tree-depth thus amounts to choosing the right children of each u_i. We proceed in a greedy way, each time choosing $v_{i,j}$ with the (lexicographically) smallest contracted ranking list. Any of these move will force Defender to move a guard from the arena to one of its two neighbors. The previously guarded vertex will then be unguarded and adjacent to another unguarded vertex, therefore, this will create an arena of smaller contracted tree-depth in the first one. With these notations, we obtain the following algorithm.

Data: A nice tree T rooted at r, a set of vertices D with $r \notin D$.
Result: An arena of minimal contracted ranking list (w.r.t. $\leq_{lex}$) among arenas of (T, D) containing r, together with its contracted ranking list.

```
if |V(T)| = 1 then
    return {r}, (r) ;
else
    for i = 1 to k do
        for j = 1 to ℓ_i do
            (A_{i,j}, L_{i,j}) ← Comp-Arena(T_{i,j}, D ∩ V(T_{i,j})) ;
        end
        (A_i, L_i) ← the pair (A_{i,j}, L_{i,j}) with minimum L_{i,j} (w.r.t. ≤_lex) ;
    end
    return {r} ∪ ⋃_i (A_i + u_i), ↑(L_1 ⊕ ··· ⊕ L_k) ;
end
```

Algorithm 1: Comp-Arena

This algorithm clearly runs in polynomial time. It remains to show that it is correct. Note that the list we output is indeed the contracted ranking list of the arena we output, by Lemma 22. The hard part is to show that the result has actually minimal contracted treedepth. This is a consequence of the following lemma.

Lemma 23. *Let T be a rooted tree with a rooted subtree T_s, T'_s be a rooted tree and T' obtained from T by replacing T_s with T'_s. If $\mathrm{rl}(T_s) \leq_{lex} \mathrm{rl}(T'_s)$, then $\mathrm{rl}(T) \leq_{lex} \mathrm{rl}(T')$.*

Note that by transitivity, it is enough to show the Lemma when T_s is rooted at a children of the root of T. Observe then that $\mathrm{rl}(T) = \uparrow(\mathrm{rl}(T_s) \oplus L)$ and $\mathrm{rl}(T') = \uparrow(\mathrm{rl}(T'_s) \oplus L)$ for some list L. We can easily see that $\oplus$ is compatible with $\leq_{lex}$ as summarized in the following fact.

Fact 24. *Let L_1, L_2 and L be three non-increasing lists such that $L_1 \leq_{lex} L_2$. Then $L_1 \oplus L \leq_{lex} L_2 \oplus L$.*

Therefore, we only have to show that $\uparrow$is also increasing w.r.t. $\leq_{lex}$.

Lemma 25. *Let L_1, L_2 be two non-increasing lists such that $L_1 \leq_{lex} L_2$. Then $\uparrow L_1 \leq_{lex} \uparrow L_2$.*

Proof. Let P be the largest common prefix of L_1 and L_2, so that $L_1 = P, x_1, \dots$ and $L_2 = P, x_2, \dots$ with $x_1 < x_2$. For $i = 1, 2$, let S_i be the suffix of L_i used when computing $\uparrow L_i$, that is the longest suffix of L_i such that $S_i \geq_{lex} (k_i, k_i - 1, \dots, 1)$ where k_i is the first element of L_i.

If S_1 does not intersect P, then $\uparrow L_1 \leq_{lex} P, x_1 + 1 \leq_{lex} L_2$ since $x_2 \geqslant x_1 + 1$. This concludes since $L \leq_{lex} \uparrow L$ for every non-increasing list L.

Otherwise, we have $k_1 \in P$. In particular, k_1 appears in L_2 and the suffix of L_2 starting at the position k_1 is larger than S_1, hence than $(k_1, \dots, 1)$. Therefore $\uparrow L_2 = P', k_2 + 1$ where P' is a prefix of P, and $\uparrow L_1$ starts with P', k_2, hence $\uparrow L_1 \leq_{lex} \uparrow L_2$ again. □

6. Cographs

In this section, we will prove the following.

Theorem 26. FAST-STRATEGY *is computable in polynomial time on cographs.*

A cograph is either an isolated vertex, or is obtained by taking the disjoint union (denoted by $\cup$) or the complete join (denoted by $\bowtie$) of two smaller cographs. This inductive definition naturally provides a representation of each cograph as a decomposition tree called *cotree*, whose leaves are the vertices of the cograph, and the internal nodes are either *join nodes* or *union nodes*[3]. We can recover the adjacency between the vertices of a cograph G from its cotree as follows: two vertices uv are adjacent in G if and only if their closest ancestor $\text{lca}(u, v)$ in the cotree is a join node.

Our polynomial time algorithm for FAST-STRATEGY on cographs will compute $t_G(D)$ using a recursive procedure applied to the cotree of G. Note that this is not restrictive since one can compute the cotree of a given cograph in linear time [13]. We actually consider a more general game, the *eternal domination game with reservists*, defined as follows.

This game is played on a board (G, D, r) where G is a graph, D is a set of guards and $r \geq 1$ is a number of reservists. Attacker plays as usual by attacking a non-guarded vertex v. Now Defender defends by either moving a guard from a neighbor of v to v, or by adding a new guard on v and decrease r by one. Attacker wins when r reaches 0, that is when Defender moves her last reservist on G. We denote by $t_G(D, r)$ the smallest number of turns required for Attacker to win. In particular, observe that $t_G(D, 1) = t_G(D)$. By convention, we assume that $t_G(D, 0) = 0$.

Theorem 26 relies on the three following results, that handle respectively the leaves, the join nodes and the union nodes of a cotree. For the base case, observe that Attacker can only win on an isolated vertex if it is not guarded and there is only one reservist, and in that case he wins in one turn. Therefore we get the following.

Fact 27. *Let G be a graph containing only an isolated vertex. Then,*

$$t_G(D, r) = \begin{cases} 1, & \text{if } D = \emptyset \text{ and } r = 1 \\ +\infty, & \text{otherwise} \end{cases}$$

We now state two recursion formulas satisfied by $t_G(D, r)$ that allow us to compute $t_G(D)$. Note that the computation only uses the values of the form $t_H(D \cap V(H), r)$ for some cograph H whose cotree is a subtree of the cograph of G and for some integer r between 0 and $|V(G)|$. Therefore, the computation can be done in polynomial time, which proves Theorem 26.

[3]Note that this decomposition is not necessarily unique but, for algorithmic purposes, we only need that such a decomposition tree exists and can be computed in polynomial time which is indeed the case.

Proposition 28. *Let G_1, G_2 be two cographs and D_1, D_2 be sets of guards on G_1 and G_2. We have:*

$$t_{G_1 \cup G_2}(D_1 \cup D_2, r) = \min_{0 \leqslant i \leqslant r} (t_{G_1}(D_1, i) + t_{G_2}(D_2, r - i))$$

and

$$t_{G_1 \bowtie G_2}(D_1 \cup D_2, r) = \min\{t_{G_1}(D_1, r + |D_2|), t_{G_2}(D_2, r + |D_1|)\}.$$

The rest of this section is devoted to proving this statement. Each of the two equalities is proved by showing two inequalities. We start with the easiest ones, stated in the two following lemmas.

Lemma 29. *Let G_1, G_2 be two cographs, $r \geqslant 0$ and D_1, D_2 be sets of guards on G_1 and G_2. We have:*

$$t_{G_1 \cup G_2}(D_1 \cup D_2, r) \leq \min_{0 \leqslant i \leqslant r} (t_{G_1}(D_1, i) + t_{G_2}(D_2, r - i)).$$

Proof. Let $G = G_1 \cup G_2$ and r_1, r_2 be two integers. If, for $i = 1, 2$, Attacker wins on (G_i, D_i, r_i) in t_i turns, then he can also win on $(G, D, r_1 + r_2)$ in $t_1 + t_2$ turns by first playing t_1 turns on (G_1, D_1, r_1) (so that Defender is forced to play her r_1-th reservist), and then playing t_2 turns on (G_2, D_2, r_2) so that Defender plays her $(r_1 + r_2)$-th reservist. □

Lemma 30. *Let G_1, G_2 be two cographs, $r \geqslant 0$ and D_1, D_2 be sets of guards on G_1 and G_2. We have:*

$$t_{G_1 \bowtie G_2}(D_1 \cup D_2, r) \leq \min\{t_{G_1}(D_1, r + |D_2|), t_{G_2}(D_2, r + |D_1|)\}.$$

Proof. Let $G = G_1 \bowtie G_2$. Observe that if Attacker plays only on G_1, the board is equivalent to $(G_1, D_1, r+|D_2|)$ since Defender can use the guards of D_2 as reservists. In particular, attacking only in G_1 or only in G_2 gives a winning strategy for Attacker in $\min\{t_{G_1}(D_1, r+|D_2|), t_{G_2}(D_2, r+|D_1|)\}$ turns. □

To prove the remaining inequalities, we consider an arbitrary order $\leq_G$ over the vertices of G and introduce $\mathcal{S}^*$ as the following strategy of Defender: if Attacker attacks the vertex v, then Defender answers (if possible) with a guarded vertex $u \in N(v)$ whose common ancestor $\mathrm{lca}(u, v)$ in the cotree of G is the deepest. If several such vertices exist, she picks the smallest with respect to the order $\leq_G$. If no such vertex exists (*i.e.* if v has no guarded neighbor), Defender calls a reservist. We denote by $t^*_G(D, r)$ the smallest number of turns required for Attacker to win on (G, D, r) against this strategy $\mathcal{S}^*$. Notice that this strategy does not depend on the number of available reservists, nor on the connected components of G that do not contain v.

Observe that if Attacker wins in k turns on (G, D, r), then he also wins in at most k turns against $\mathcal{S}^*$. Therefore, we get the following.

Fact 31. *Every cograph G satisfies $t^*_G \leq t_G$.*

In the following, we will show by induction that this inequality is actually an equality, meaning that $\mathcal{S}^*$ is an optimal strategy for Defender.

For the base case, Fact 27 clearly remains true when Attacker plays against the strategy $\mathcal{S}^*$.

Fact 32. *Let G be a graph containing only an isolated vertex. Then,*

$$t^*_G(D, r) = \begin{cases} 1, & \text{if } D = \emptyset \text{ and } r = 1 \\ +\infty, & \text{otherwise} \end{cases}$$

To prove the inductive step, we will show that the converse inequalities of Lemmas 29 and 30 hold when replacing t_G by t^*_G. More precisely, we show the two following lemmas.

Lemma 33. *Let G_1, G_2 be two cographs, $r \geqslant 0$ and D_1, D_2 be sets of guards on G_1 and G_2. We have:*

$$t^*_{G_1 \cup G_2}(D_1 \cup D_2, r) \geq \min_{0 \leqslant i \leqslant r} (t^*_{G_1}(D_1, i) + t^*_{G_2}(D_2, r - i)).$$

Lemma 34. *Let G_1, G_2 be two cographs, $r \geqslant 0$ and D_1, D_2 be sets of guards on G_1 and G_2. We have:*

$$t^*_{G_1 \bowtie G_2}(D_1 \cup D_2, r) \geq \min\{t^*_{G_1}(D_1, r + |D_2|), t^*_{G_2}(D_2, r + |D_1|)\}.$$

Before proving these results, let us show how to use them to conclude the proof of Proposition 28.

Proof of Proposition 28. We prove by induction on cographs that every cograph G satisfies $t_G(D, r) = t^*_G(D, r)$ for any set of guards D and $r \geqslant 0$. The base case is already provided by Fact 27 and Fact 32.

Now, let G be a cograph defined as the union of two cographs G_1 and G_2. Let $r \geqslant 0$ and D be a set of guards, and $D_i = D \cap V(G_i)$ for $i = 1, 2$. Applying successively Lemma 29, the induction hypothesis on G_1, G_2 and Lemma 33, we get

$$\begin{aligned} t_G(D, r) &\leq \min_{0 \leqslant i \leqslant r} (t_{G_1}(D_1, i) + t_{G_2}(D_2, r - i)) \\ &= \min_{0 \leqslant i \leqslant r} (t^*_{G_1}(D_1, i) + t^*_{G_2}(D_2, r - i)) \\ &\leq t^*_G(D, r) \end{aligned}$$

This concludes using Fact 31. The case of joins is similar, we only use Lemmas 30 and 34 instead. □

It thus remains to prove Lemmas 33 and 34. We start with the case of unions.

Proof of Lemma 33. Let $G = G_1 \cup G_2$, $D = D_1 \cup D_2$ and A be a winning strategy in $t^*_G(D, r)$ turns on (G, D, r) against the strategy $\mathcal{S}^*$. Since the strategy of Defender is prescribed, we may assume that A is just a sequence of moves.

Let A_1 (resp. A_2) be the subsequence of attacks of A played on G_1 (resp. G_2) and r_1 (resp. r_2) be the number of reservists called in G_1 (resp. G_2). Denote by a_1 (resp. a_2) the length of A_1 (resp. a_2), so that $r_1 + r_2 = r$ and $a_1 + a_2 = t^*_G(D, r)$. Since the strategy $\mathcal{S}^*$ of Defender only depends on the connected component of the attacked vertex, for $i = 1, 2$, A_i is a sequence of attacks on (G_i, D_i) that moves r_i reservists when Defender plays according to $\mathcal{S}^*$. In particular, $a_i \geq t^*_{G_i}(D_i, r_i)$.

Consequently, $t^*_G(D, r) = a_1 + a_2 \geq t^*_{G_1}(D_1, r_1) + t^*_{G_2}(D_2, r_2)$, which concludes. □

We end this section with the case of joins. Lemma 34 relies on the following result.

Lemma 35. *Let $G = G_1 \bowtie G_2$ be a cograph, D be a set of guards and $r \geq 1$ be an integer. Then, Attacker can win on (G, D, r) in $t^*_G(D, r)$ steps against the strategy $\mathcal{S}^*$ by only playing on G_1 or by only playing on G_2.*

Lemma 35 states that (up to exchanging G_1 and G_2) Attacker has a winning strategy in $t^*_{G_1 \bowtie G_2}(D, r)$ turns on $(G_1 \bowtie G_2, D, r)$ against the strategy $\mathcal{S}^*$ where he plays only on G_1. Since vertices of G_2 have the highest possible common ancestors with vertices of G_1 in the cotree of $G_1 \bowtie G_2$,

this is equivalent to playing on $(G_1, D_1, r + |D_2|)$ against the strategy $\mathcal{S}^*$ where $D_i = D \cap V(G_i)$ for $i = 1, 2$. In particular, $t^*_G(D, r) \geqslant t^*_{G_1}(D_1, r + |D_2|)$, and we get Lemma 34 by symmetry.

The proof of Lemma 35 is based on this trade-off between increasing the number of reservists and playing only on one side of a join. We first summarize this in the following fact.

Fact 36. *Let G_1 and G_2 be two cographs, each one having a set of guards D_1 and D_2. Assume that Attacker wins in k turns against $\mathcal{S}^*$ on $(G_1 \bowtie G_2, D_1 \cup D_2, r)$ by playing only on G_2. Then $k \geqslant t^*_{G_2}(D_2, r + |D_1|)$.*

Another observation is given below. It formalizes the intuition that defending using a reservist instead of a guard already in a graph does not slow down Attacker.

Claim 37. *Let G be a graph, D a set of guards on G and $r \geq 1$. For every $x \in D$, we have $t^*_G(D, r) \leq t^*_G(D \setminus \{x\}, r + 1)$.*

Proof. We proceed by induction on $t^* = t^*_G(D \setminus \{x\}, r + 1)$. If $t^* = 1$, then we must have $r = 0$ and in that case $t^*_G(D, 0) = 0$, which concludes.

Assume now that $t^* \geq 2$. Let (D, r) be the current configuration and $x \in D$. We denote by $P_1 = (D, r)$ and $P_2 = (D \setminus \{x\}, r + 1)$. Let u be the first vertex played by Attacker in P_2, and denote by P'_2 the configuration obtained after Defender answered using $\mathcal{S}^*$. Similarly, denote by P'_1 the configuration obtained after Defender used $\mathcal{S}^*$ to defend u in P_1. By definition, observe that $t^* = t^*_G(P_2) = 1 + t^*_G(P'_2)$ and that $t^*_G(P_1) \leqslant 1 + t^*_G(P'_1)$. Consider the following cases:

- Assume that Defender moves the guard from x to defend u in P_1, and that u has no guarded neighbor in P_2. So Defender uses a reservist in P_2 and then the resulting instances P'_1 and P'_2 are $(D \cup \{u\} \setminus \{x\}, r)$ so the conclusion follows immediately.

- Assume that Defender moves the guard from x to defend u in P_1 and a guard from a vertex v to defend u in P_2. Then $P'_1 = (D \setminus \{x\} \cup \{u\}, r)$ and $P'_2 = (D \setminus \{x, v\} \cup \{u\}, r + 1)$. Applying the induction hypothesis with D replaced by $D \setminus \{x\} \cup \{u\}$ yields $t^*_G(P'_1) \leqslant t^*_G(P'_2)$ and we get

$$t^*_G(P_1) \leqslant 1 + t^*_G(P'_1) \leqslant 1 + t^*_G(P'_2) = t^*.$$

- Otherwise, Defender defends with strategy $\mathcal{S}^*$ against the attack on u in P_1 without moving the guard from vertex x. In this case, the strategy $\mathcal{S}^*$ performs the same move in P_2, and we may conclude applying induction as in the previous item. □

We are now ready to conclude the proof of Lemma 35.

Proof of Lemma 35. We proceed by induction on $t^* = t^*_G(D, r)$. If $t^* = 1$, the result follows since Attacker wins against $\mathcal{S}^*$ in a single move.

Assume now that $t^* \geq 2$, and let u be the first vertex attacked by Attacker in a winning strategy in t^* turns against $\mathcal{S}^*$. Without loss of generality, assume that u is in G_1 and let v be the answer from Defender according to $\mathcal{S}^*$. On the obtained instance $(G, D \setminus \{v\} \cup \{u\}, r)$ (or $(G, D \cup \{u\}, r - 1)$ if a reservist has been used), Attacker has a winning strategy in $t^* - 1$ turns when Defender uses $\mathcal{S}^*$. So, by induction, Attacker has a winning strategy against $\mathcal{S}^*$ playing only in G_1 or in G_2. We can assume that it is in G_2 (since otherwise we are done), and we may thus apply Fact 36.

Setting $D_i = D \cap V(G_i)$, we consider three cases depending on v:

- If v is a reservist, then $t^* - 1 \geqslant t^*_{G_2}((D \cup \{u\}) \cap V(G_2), r - 1 + |(D \cup \{u\}) \cap V(G_1)|) = t^*_{G_2}(D_2, r + |D_1|)$.
- If v is in G_1, we similarly get $t^* - 1 \geqslant t^*_{G_2}(D_2, r + |D_1|)$.
- Otherwise v is in G_2 and we get $t^* - 1 \geqslant t^*_{G_2}(D_2 \setminus \{v\}, r + |D_1| + 1)$. By Claim 37, this is at least $t^*_{G_2}(D_2, r + |D_1|)$.

In each case, there is a winning strategy for Attacker on $(G_2, D_2, r + |D_1|)$ against $\mathcal{S}^*$ in at most $t^* - 1$ turns. But this is also a strategy for Attacker on (G, D, r) against $\mathcal{S}^*$ where he plays only on G_2. This is a contradiction by definition of t^*, which concludes the proof. □

7. Parameterized complexity

In this section, we study the parameterized complexity of Fast-Strategy. We consider two parameters: t the number of turns and g the number of guards. We first consider generic results, then study restricted classes of graphs.

7.1. Generic graphs

As a first result, we will show the following theorem.

Theorem 38. Fast-Strategy *parameterized by g is in* XP.

Proof. Let $G = (V, E)$ be a graph. We will compute all the values $t_G(D)$ for each subset $D \subset V$ of size g in time $O(n^{2g+2})$.

We first build an auxiliary directed graph $\mathcal{G}$ that models the moves available for each player. For every subset D of g vertices of G, and for every vertex $v \in V$, $\mathcal{G}$ has a vertex labeled by D, and one labeled by (D, v). Moreover, $\mathcal{G}$ contains the arcs:

- $D \to (D, v)$ for every set D and $v \in V$, and
- $(D, v) \to (D \cup \{v\}) \setminus \{u\}$ for every set D, $v \in V \setminus D$ and $u \in D \cap N(v)$.

We now label all vertices (D, v) such that $D \cap N(v) = \emptyset$ with 0. Then we apply the following rules while it is possible.

- We label each non-labeled vertex D with $\ell+1$ where ℓ is the minimum label of its out-neighbors (if at least one such out-neighbor is labeled).
- If every out-neighbor of an non-labeled vertex (D, v) is labeled, we label it with the maximum label of its out-neighbors.

At the end, all non-labeled configurations are labeled $+\infty$.

Note that each rule can be applied in linear time with respect to $\mathcal{G}$. Moreover, each application of these rules labels at least one new vertex (and these labels do not change afterward), hence the total procedure is at most quadratic in $\mathcal{G}$. Since $\mathcal{G}$ has $O(n^{g+1})$ vertices and $O(n^{g+1})$ arcs, this yields an XP algorithm.

We now claim that afterward, each vertex D is labeled with $t_G(D)$. Indeed, if D is labeled with $+\infty$, then so are all its out-neighbors and in particular, for every $v \notin D$, (D, v) has an out-neighbor

$(D \cup \{u\}) \setminus v$ labeled with $+\infty$. Therefore, Defender can eternally defend by answering to Attacker in such a way the set of guards stays labeled with $+\infty$.

Similarly, one can easily see by induction on ℓ that if a set D of guards is labeled with ℓ, then Attacker wins in at most ℓ turns and Defender can resist to $\ell - 1$ attacks on D, hence $t_G(D) = \ell$. The main point of the induction is that at any moment of the game, Attacker has a move available from a position D to a position (D, v) which decreases the label by one, while Defender has a move available from a position (D, v) to a position D' which has the same label. □

Theorem 39. Fast-Strategy *is* W[1]*-hard when parameterized by* $t + g$.

Proof. Let (G, k) be an instance of Independent-Set. We build an instance of Fast-Strategy as follows. Let G' be the graph obtained from G by adding a set D of $k - 1$ guarded vertices $u_1 \ldots, u_{k-1}$, each of them connected to all vertices of G. We claim that (G, k) is a positive instance of Independent-Set if and only if (G', D, k) is a positive instance of Fast-Strategy.

If G admits an independent set I of size k, then Attacker wins in k turns by attacking successively the vertices in I. Indeed, at each turn, Defender has to move a guard from D (she cannot move a guard twice since I is independent). Therefore, she loses at the k-th turn since no guard is available in D.

Conversely, assume that Attacker can win in k turns. If Defender manages to move the same guard twice during the game, then either she can answer the last attack with a guard in G, or one guard is still on D at the beginning of the k-th turn, and she can use it to answer the last attack, a contradiction. Therefore, the k vertices played by Attacker must induce an independent set of G, which concludes. □

7.2. Bipartite graphs

It is easily seen that the reduction in Theorem 2 is an FPT-reduction when parameterized by t (but not by g). Since Independent-Set is $W[1]$-complete when parameterized by the size of the independent set, we obtain the following result.

Theorem 40. Fast-Strategy *parameterized by* t *is co-*$W[1]$*-hard on bipartite graphs.*

The behavior of Fast-Strategy becomes quite different when parameterized by g instead, as shown by the following result.

Theorem 41. Fast-Strategy *parameterized by* g *is FPT on bipartite graphs.*

Proof. Let (G, D, t) be an instance of Fast-Strategy where G is a bipartite graph $(A \cup B, E)$. By Lemma 3 and Lemma 4, we can assume that $B = D$ and that Attacker plays only in A. In particular, $t_G(D) \leqslant |B| + 1 = g + 1$.

We build a kernel (G', D, t) of (G, D, t) as follows: for every set of at least $g + 1$ twins in A, delete all but $g + 1$ of them. Observe that G' has at most $g + (g + 1)2^g$ vertices. We claim that $t_G(D) = t_{G'}(D)$.

Recall that Attacker always has a shortest winning strategy where he plays only on initially unguarded vertices by Lemma 3. In particular, if he wins in t turns on (G', D), he can just use the same strategy to win in t turns on (G, D).

Conversely, if Attacker wins in t turns on (G, D), then mimicking his strategy on (G', D) is also making him win in at most t turns (note that each time he plays on a vertex v of G that was deleted, there is at least one unguarded twin of v in G'). □

Finally, we prove that Fast-Strategy is FPT when it is parameterized by $n - g$, the number of non-guarded vertices. First, we state a useful lemma inspired by crown decompositions [14].

Lemma 42. *Let $G = (A \cup B, E)$ be a bipartite graph. Assume that there is no isolated vertex in B and $|A| < |B|$. Then one can compute in polynomial time a non-empty set $A' \subseteq A$ and a set $B' \subseteq B$ such that*

- *there is a matching M between A' and B' that covers A';*
- *there is no edge between B' and $A \setminus A'$.*

Proof. By Kőnig's theorem [15], there exist a matching M and a vertex cover C such that $|C| = |M| \leq |A|$. Set $A' = C \cap A$ and $B' = B \setminus C$. Observe that $A' \neq \varnothing$, otherwise $C \subset B$, and since B has no isolated vertex, $C = B$, a contradiction since $|A| \geq |C| = |B| > |A|$.

Now observe that every edge e in M has exactly one endpoint in C, hence the edges of M intersecting A' have their other endpoint in B'. Thus, these edges form a matching between A' and B' that covers A'.

Finally, by definition of a vertex cover, $(A \cup B) \setminus C$ is an independent set of G. In particular, there is no edge between $B' = B \setminus C$ and $A \setminus A' = A \setminus C$. □

Using Lemma 42, we can prove the following theorem.

Theorem 43. Fast-Strategy *admits a kernel of size $2(n - g)$ on bipartite graphs.*

Proof. Let $G = (A \cup B, E)$ be a bipartite graph and D be a set of guards. By Lemma 3 and Lemma 4, we can assume $B = D$ and Attacker plays only in A. We use the following two reduction rules.

- Remove isolated vertices in B.
- If $|A| < |B|$, let A', B' the sets obtains by Lemma 42. Then remove A' and B' from G and remove B' from D.

After iterating these two rules while we can, we end up with a kernel (G', D') with less guarded vertices than unguarded, hence it has size at most $2(n - g)$, as requested. It remains to show that these rules are correct. The first rule clearly preserves $t_G(D)$ since guarded isolated vertex cannot be attacked or used to defend another vertex.

Consider now the second rule, and assume that $|A| < |B|$. Let A', B' the set obtained by applying Lemma 42, and (G', D') the new instance. We claim that $t_G(D) = t_{G'}(D')$.

If Attacker has a winning strategy in t turns on (G', D'), then he can apply the same strategy on (G, D) since, by construction, Defender has the same available moves on (G, D) and on (G', D'). Therefore, $t_G(D) \leq t_{G'}(D')$.

Conversely, assume that Defender has a strategy to resist to t attacks on (G', D'). Then she can resist t attacks on (G, D) using the following: if Attacker plays on a vertex in $A \setminus A'$, then she uses the same strategy as on (G', D'). Otherwise, let M be a matching between A' and B' saturating A'. When Attacker plays on a vertex v in A', Defender answers with the guard on B' matched with v in M. □

7.3. First-order definability

In this subsection we show that Fast-Strategy can be expressed with a first-order formula. This directly implies complexity upper bounds using well-known meta-theorems. The first one comes from the complexity of FO-model-checking in the generic case [17].

Theorem 44. Fast-Strategy *parameterized by t is in* AW[$*$].

It is well-known that this complexity drops when considering restricted classes of graphs, for example nowhere-dense graph classes [21] and bounded twin-width graph classes [7]. Notice that nowhere-dense graph classes include planar graphs, bounded degree graphs, and graphs with an excluded minor [29].

Theorem 45. Fast-Strategy *parameterized by t is* FPT *on nowhere-dense graph classes.*

One can note that the same holds for bounded twin-width graphs when we are given a contraction sequence.

This result relies on the following.

Lemma 46. *For every $t > 0$, there is a first-order formula $\varphi_t(X)$ such that $G \models \varphi_t(D)$ if and only if $t_G(D) \leq t$.*

Proof. Let X be a set of guards and $k < t$. Our goal is to define a formula $\psi_{t,k}(X, a_1, d_1, \ldots, a_k, d_k)$ that holds if Attacker can win in at most t turns when the first k turns were already played on $a_1, d_1, \ldots, a_k, d_k$. In particular, we will have $\varphi_t = \psi_{t,0}$.

First observe that the set X_k of guards obtained from X after playing these k turns can be defined in FO. Indeed, the formula guards_k defined by $\text{guards}_0(X, x) = x \in X$ and

$$\text{guards}_k(X, a_1, d_1, \ldots, a_k, d_k, x) = (x = a_k) \vee (\text{guards}_{k-1}(X, a_1, d_1, \ldots, a_{k-1}, d_{k-1}, x) \wedge x \neq d_k)$$

holds if and only if $x \in X_k$.

Observe that Attacker wins in at most t turns after $k < t$ turns are played if either X_k is not a dominating set or $k+1 < t$ and there is a position a_{k+1} such that for every answer b_{k+1} of Defender, $\psi_{t,k+1}$ holds. Therefore, denoting by

$$\text{dom}_k(X, a_1, d_1, \ldots, a_k, d_k) = \forall x \exists y (x - y) \wedge \text{guards}_k(X, a_1, d_1, \ldots, a_k, d_k, y)$$

the formula stating that X_k dominates G, we have $\psi_{t,t-1} = \neg\text{dom}_t$ and

$$\psi_{t,k}(X, a_1, d_1, \ldots, a_k, d_k) = \neg\text{dom}_k(X, a_1, d_1, \ldots, a_k, d_k)$$
$$\vee \exists a_{k+1} \forall b_{k+1} (a_{k+1} - b_{k+1} \wedge \text{guards}_k(X, a_1, d_1, \ldots, a_k, d_k, b_{k+1})) \Rightarrow \psi_{t,k+1}(X, a_1, d_1, \ldots, a_{k+1}, b_{k+1}),$$

which concludes. □

8. Conclusion

We give a summary table of the different complexity results.

	Eternal-Domination-Number	Eternal-Dominating-Set	Fast-Strategy
Trees	P[19]	P(Th 1)	P(Th 11)
Cographs	P[19]	P(Th 26)	P(Th 26)
Bipartite	P[19]	P(Th 1)	coNP-hard (Th 2)
Split	P[19]	?	coNP-hard (Th 6)
Perfect	P[19]	?	PSPACE-hard (Th 8)
General	coNP-hard [3]	?	PSPACE-hard (Th 8)

In addition, we obtained several results on the parameterized complexity of Fast-Strategy.

class of graphs	parameter	complexity
General	t	AW[$*$] (Th 44)
General	g	XP(Th 38)
General	$g+t$	W[1]-hard (Th 39)
Bipartite	t	co-W[1]-hard (Th 40)
Bipartite	g	FPT(Th 41)
Bipartite	$n-g$	linear kernel (Th 43)
Nowhere dense	t	FPT(Th 45)
Bounded twin-width	$t+tww$	FPT(Th 45)

Quite often in the literature, game problems are either in PTime or PSpace-hard. Motivated by Theorem 2, we propose the following conjecture.

Conjecture 47. Fast-Strategy *is* PSpace*-complete on bipartite graphs.*

Besides Conjecture 47, we enumerate some open questions for future work.

1. Are there other graph classes where Fast-Strategy is polynomial? Good candidates are unit interval graphs. We think that the notion of arena (defined for trees) can be adapted to these graphs.
2. In this paper, we have only considered the case where there is at most one guard per vertex. One can allow multiple guards on the same vertex *i.e.* the guards now form a multiset.
   There is a generic reduction to the case considered here, roughly consisting in adding twins. More precisely, given a graph G and a multiset of guards D, consider the graph G' and set of guards D' obtained by replacing k guards on a vertex v into a set of k true twins, all of them being guarded. One can see that $t_G(D) = t_{G'}(D')$. Cographs are preserved by this transformation since they are closed under adding true twins. Thus, Theorem 26 still holds. However, this is not true for trees and adapting the proof of Theorem 11 does not seem straightforward.
3. For perfect graphs, we know that Eternal-Dominating-Number is in Pand Mobile-Domination-Game is PSPACE-hard. What is the complexity of Eternal-Dominating-Set on such graphs?
4. From a parameterized complexity perspective, we showed that Mobile-Domination-Game parameterized by the number of guards is FPT on bipartite graphs. A natural extension consists in looking for a polynomial kernel.

5. Can our results be adapted in the "all guards move" variant of eternal domination (where Defender can move more than one guard at each turn)? An upper bound on the number of moves for Attacker to win has been given in [5].

References

[1] M. Anderson, C. Barrientos, R. Brigham, R. Carrington, J. Vitray, and J. Yellen. Maximum demand graphs for eternal security. *Journal of Combininatorial Mathematics and Combinatorial Computing*, 61:111–128, 2007.

[2] V. Anuradha, C. Jain, J. Snoeyink, and T. Szabó. How long can a graph be kept planar? *Electronic Journal of Combinatorics*, 15(1), 2008.

[3] G. Bagan, A. Joffard, and H. Kheddouci. Eternal dominating sets on digraphs and orientations of graphs. *Discrete Applied Mathematics*, 291:99–115, 2021.

[4] S. Bard, C. Duffy, M. Edwards, G. Macgillivray, and F. Yang. Eternal domination in split graphs. *Journal of Combinatorial Mathematics and Combinatorial Computing*, 101:121–130, 2017.

[5] V. Blažej, J. M. Křišťan, and T. Valla. Efficient attack sequences in m-eternal domination. *arXiv:2204.02720*, 2022.

[6] V. Blažej, J. M. Křišťan, and T. Valla. Computing m-eternal domination number of cactus graphs in linear time. *arxiv:2301.05155*, 2023.

[7] E. Bonnet, E. J. Kim, S. Thomassé, and R. Watrigant. Twin-width I: Tractable FO model checking. *Journal of the ACM*, 69(1), 2021.

[8] A. Braga, C. C. de Souza, and O. Lee. The eternal dominating set problem for proper interval graphs. *Information Processing Letters*, 115(6):582 – 587, 2015.

[9] B. Brešar, P. Dorbec, S. Klavžar, G. Košmrlj, and G. Renault. Complexity of the game domination problem. *Theoretical Computer Science*, 648:1–7, 2016.

[10] A. Burger, E. Cockayne, W. Grundlingh, C. Mynhardt, J. Van Vuuren, and W. Winterbach. Finite order domination in graphs. *Journal of Combinatorial Mathematics and Combinatorial Computing*, 49:159–176, 2004.

[11] A. Burger, E. Cockayne, W. Gründlingh, C. Mynhardt, J. V. Vuuren, and W. Winterbach. Infinite order domination in graphs. *Journal of Combinatorial Mathematics and Combinatorial Computing*, 50:179–194, 2004.

[12] N. Cohen, N. A. Martins, F. Mc Inerney, N. Nisse, S. Pérennes, and R. Sampaio. Spy-game on graphs: Complexity and simple topologies. *Theoretical Computer Science*, 725:1–15, 2018.

[13] D. G. Corneil, Y. Perl, and L. K. Stewart. A linear recognition algorithm for cographs. *SIAM Journal on Computing*, 14(4):926–934, 1985.

[14] M. Cygan, F. V. Fomin, L. Kowalik, D. Lokshtanov, D. Marx, M. Pilipczuk, M. Pilipczuk, and S. Saurabh. *Parameterized Algorithms*. Springer, 2015.

[15] R. Diestel. *Graph Theory: Springer Graduate Text GTM 173*. Springer Graduate Texts in Mathematics (GTM). Springer, 2012.

[16] E. M. Eschen and X. Wang. Algorithms for unipolar and generalized split graphs. *Discrete Applied Mathematics*, 162:195–201, 2014.

[17] J. Flum and M. Grohe. *Parameterized Complexity Theory*. Texts in Theoretical Computer Science. An EATCS Series. Springer, 2006.

[18] V. Gledel, V. Iršič, and S. Klavžar. Fast winning strategies for the maker-breaker domination game. *Electronic Notes in Theoretical Computer Science*, 346:473–484, 2019. LAGOS 2019.

[19] W. Goddard, S. Hedetniemi, and S. Hedetniemi. Eternal security in graphs. *Journal of Combinatorial Mathematics and Combinatorial Computing*, 52, 2005.

[20] J. L. Goldwasser and W. F. Klostermeyer. Tight bounds for eternal dominating sets in graphs. *Discrete Mathematics*, 308(12):2589–2593, 2008.

[21] M. Grohe, S. Kreutzer, and S. Siebertz. Deciding first-order properties of nowhere dense graphs. *Journal of the ACM*, 64(3), 2017.

[22] D. Hefetz, M. Krivelevich, M. Stojaković, and T. Szabó. Fast winning strategies in maker–breaker games. *Journal of Combinatorial Theory, Series B*, 99(1):39–47, 2009.

[23] F. M. Inerney, N. Nisse, and S. Pérennes. Eternal domination: D-dimensional cartesian and strong grids and everything in between. *Algorithmica*, 83(5):1459–1492, 2021.

[24] A. V. Iyer, H. D. Ratliff, and G. Vijayan. Optimal node ranking of trees. *Information Processing Letters*, 28(5):225–229, 1988.

[25] W. Klostermeyer and G. MacGillivray. Eternal security in graphs of fixed independence number. *Journal of Combinatorial Mathematics and Combinatorial Computing*, 63, 11 2007.

[26] W. Klostermeyer and G. MacGillivray. Eternal dominating sets in graphs. *Journal of Combinatorial Mathematics and Combinatorial Computing*, 68:97–111, 02 2009.

[27] W. F. Klostermeyer and C. Mynhardt. Eternal and secure domination in graphs. In *Topics in Domination in Graphs*, pages 445–478. Springer, 2020.

[28] I. Lamprou, R. Martin, and S. Schewe. Eternally dominating large grids. *Theoretical Computer Science*, 794:27–46, 2019.

[29] J. Nešetřil and P. Ossona de Mendez. On nowhere dense graphs. *European Journal of Combinatorics*, 32(4):600–617, 2011.

[30] M. Rinemberg and F. Soulignac. The eternal dominating set problem for interval graphs. *Information Processing Letters*, 146, 06 2019.

[31] R. Samal and T. Valla. The guarding game is E-complete. *Theoretical Computer Science*, 521:92–106, 2014.

[32] T. J. Schaefer. On the complexity of some two-person perfect-information games. *Journal of Computer and System Sciences*, 16(2):185–225, 1978.