Recontamination Helps a Lot to Hunt a Rabbit

Thomas Dissaux
Université Côte d'Azur, Inria, CNRS, I3S, France

Foivos Fioravantes
Université Côte d'Azur, Inria, CNRS, I3S, France
Department of Theoretical Computer Science, FIT, Czech Technical University in Prague, Czech Republic

Harmender Gahlawat
Ben-Gurion University of the Negev, Beersheba, Israel

Nicolas Nisse
Université Côte d'Azur, Inria, CNRS, I3S, France

Abstract

The Hunters and Rabbit game is played on a graph G where the Hunter player shoots at k vertices in every round while the Rabbit player occupies an unknown vertex and, if it is not shot, must move to a neighbouring vertex after each round. The Rabbit player wins if it can ensure that its position is never shot. The Hunter player wins otherwise. The hunter number $h(G)$ of a graph G is the minimum integer k such that the Hunter player has a winning strategy (i.e., allowing him to win whatever be the strategy of the Rabbit player). This game has been studied in several graph classes, in particular in bipartite graphs (grids, trees, hypercubes...), but the computational complexity of computing $h(G)$ remains open in general graphs and even in more restricted graph classes such as trees. To progress further in this study, we propose a notion of monotonicity (a well-studied and useful property in classical pursuit-evasion games such as Graph Searching games) for the Hunters and Rabbit game imposing that, roughly, a vertex that has already been shot "must not host the rabbit anymore". This allows us to obtain new results in various graph classes.

More precisely, let the monotone hunter number $mh(G)$ of a graph G be the minimum integer k such that the Hunter player has a monotone winning strategy. We show that $pw(G) \leq mh(G) \leq pw(G)+1$ for any graph G with pathwidth $pw(G)$, which implies that computing $mh(G)$, or even approximating $mh(G)$ up to an additive constant, is NP-hard. Then, we show that $mh(G)$ can be computed in polynomial time in split graphs, interval graphs, cographs and trees. These results go through structural characterisations which allow us to relate the monotone hunter number with the pathwidth in some of these graph classes. In all cases, this allows us to specify the hunter number or to show that there may be an arbitrary gap between h and mh, i.e., that monotonicity does not help. In particular, we show that, for every $k \geq 3$, there exists a tree T with $h(T) = 2$ and $mh(T) = k$. We conclude by proving that computing h (resp., mh) is FPT parameterised by the minimum size of a vertex cover.

2012 ACM Subject Classification Mathematics of computing → Graph algorithms

Keywords and phrases Hunter and Rabbit, Monotonicity, Graph Searching

Digital Object Identifier 10.4230/LIPIcs.MFCS.2023.42

Related Version *Full Version*: `https://hal.science/hal-03995642v3` [12]

Funding This research was partially funded by the project UCA JEDI (ANR-15-IDEX-01) and EUR DS4H Investments in the Future (ANR-17-EURE-004) and the ANR Digraphs and the IFCAM project Applications of graph homomorphisms (MA/IFCAM/18/39) and ERC grant titled PARAPATH and the CTU Global postdoc fellowship program.

1 Introduction

The Hunters and Rabbit game is played on a graph G and with a positive integer k (the number of hunters), where the Hunter player shoots at k vertices in every round while the Rabbit player occupies an unknown vertex and, if it is not shot, must move to a neighbouring vertex after each round. The Rabbit player wins if he can ensure that its position is never shot. The Hunter player wins otherwise. The Hunters and Rabbit game was first introduced in [7], in the case $k = 1$, where it was shown that the Hunter player wins in a tree T if and only if T does not contain as subgraph any tree obtained from a star with 3 leaves by subdividing each edges twice. This result was also observed in [19], where the authors also consider the minimum number of rounds needed for the Hunter player to win. The version where $k > 1$ was first considered in [1]. Observe that, if $k = |V(G)| - 1$, the Hunter player can win in any graph G (in two rounds) by shooting twice a subset of k vertices of G. Hence, let the *hunter number* of G, denoted by $h(G)$, be the minimum integer k such that k hunters can win in G whatever be the rabbit strategy.

In [1], it is shown that the hunter number is closed under taking subgraphs. Moreover, this result trivially extends to the case when the starting positions of the rabbit are restricted. The exact value of $h(G)$ has been determined for several specific families of graphs G. For any $n \geq 2$, $h(P_n) = 1$ where P_n is the path with n vertices [7] (because the rabbit is forced to move at every round, $h(P_1) = 0$). For any $n \geq 3$, $h(C_n) = 2$ and $h(K_n) = n - 1$, where C_n and K_n are the cycle and complete graph on n vertices respectively [1]. Moreover, $h(G_{n\times m}) = \lfloor \frac{\min\{n,m\}}{2} \rfloor + 1$ [1] and $h(Q^n) = 1 + \Sigma_{i=0}^{n-2} \binom{i}{\lfloor i/2 \rfloor}$ [5], where $G_{n\times m}$ is the $n \times m$ grid and Q^n is the hypercube with dimension n. The case of bipartite graphs has been particularly studied because it was proved in [1] that we may constrain the rabbit to start in a fixed part of the bipartition, without decreasing the hunter number. By taking advantage of the bipartiteness of trees, it was proved that, for any tree T, $h(T) \leq \lceil \frac{1}{2} \log_2(|V(T)|) \rceil$ [17]. Surprisingly, the computational complexity of the problem that takes a graph G and an integer k as inputs and aims at deciding whether $h(G) \leq k$ is still open, even if G is restricted to be a tree.

In this paper, we progress further in this research direction by exhibiting new classes of graphs G where $h(G)$ can be determined in polynomial time.

Graph Searching games. The Hunters and Rabbit game takes place in the larger class of Graph Searching games initially introduced in [6, 26]. In these pursuit-evasion games, one player plays with a team of searchers (or hunters, etc.) that must track a fugitive (or robber, rabbit, etc.) moving in a graph. Multiple games fall under this framework, each one specifying its own rules on, for example, the available moves of the searchers, the speed of the fugitive, whether the fugitive is visible or not, etc. Several variations of Graph Searching games have been studied in the literature due to their numerous applications in artificial intelligence [21], robot motion planning [9], constraint satisfaction problems and database theory [16], and distributed computing [25]. The study of such games also leads to significant implications in graph theory and algorithms. Indeed, many variants of these games provide algorithmic interpretations of width measures of graphs like treewidth [27], pathwidth [26], tree-depth [15], hypertree-width [2], cycle-rank [15], and directed tree-width [22]. Central to the connection between Graph Searching games and such structural parameters is the notion of *monotonicity* [3, 27, 24, 20]. In short, a searchers' strategy is *monotone* if it ensures that the fugitive can never "recontaminate" a vertex, i.e., it can never access a vertex that has already been "visited" by a searcher. The main question is then, given a game, whether "recontamination does not help in this game" [23], i.e., whether there always exists, in this game, an optimal (in terms of number of searchers) monotone winning strategy for the searchers. In particular, the monotonicity played a central role in the proof that the minimum number of searchers to capture an invisible (resp., visible) fugitive in the node-searching game played in a graph G equals its pathwidth plus one [3] (resp., treewidth plus one [27]).

Unsurprisingly, the Hunters and Rabbit game also has a close relationship with the pathwidth of graphs. Precisely, the hunter number of any graph is at most its pathwidth plus one [1]. In this paper, we investigate further this relationship and, for this purpose, we define, and study, a notion of monotonicity adapted to the Hunters and Rabbit game.

Our contribution. In Section 2, we give the main notations used throughout this paper. In Section 3, we introduce the notion of monotonicity for the Hunters and Rabbit game which is not straightforward; let $mh(G)$ denote the *monotone hunter number of* G. Then, we prove that $mh(G) \in \{pw(G), pw(G)+1\}$ in any graph G. Along with implying that it is NP-complete to compute $mh(G)$ for a graph G, this result also implies that it is NP-hard to approximate $mh(G)$ up to an additive error of $|V(G)|^{\varepsilon}$, for $0 < \varepsilon < 1$. On the positive side, in Section 4, we give polynomial time algorithms to determine $h(G)$ and/or $mh(G)$ in cographs, split and interval graphs. In Section 5, we adapt the Parsons' Lemma [26] to the case of the monotone Hunters and Rabbit game which leads to a polynomial time algorithm that computes $mh(T)$ for any tree T. Then, we investigate the monotonicity property in the case of the "bipartite" variant of the Hunters and Rabbit game (see [1, 17]) in trees. In particular, we show that, for any $k \in \mathbb{N}$, there exist trees T such that $h(T) = 2$ and $mh(T) \geq k$. That is, "recontamination helps a lot" in the Hunters and Rabbit game. Finally, in Section 6, we show as a general positive result that the problem of deciding if $h(G) \leq k$ or $mh(G) \leq k$, for some given integer k, is in FPT when parameterized by the vertex cover number of G. This is done through kernelization. We close our study by providing directions for further research in Section 7.

2 Preliminaries

Unless mentioned otherwise, in this paper we will always deal with graphs $G = (V, E)$ that are non empty, finite, undirected, connected and simple. For any two adjacent vertices $x, y \in V$, let $xy \in E$ denote the edge between x and y. Given a set $S \subseteq V$, let $G[S]$ denote the subgraph of G induced by (the vertices in) S and let $G \setminus S$ denote the subgraph $G[V \setminus S]$. For any $v \in V$ and $X \subseteq V$, let $N_X(v) = \{u \in X \mid uv \in E\}$ be the *open neighbourhood* of v in X and let the *closed neighbourhood* of v in X be $N_X[v] = (N_X(v) \cup \{v\}) \cap X$. If $X = V$, we simply write $N(v)$ and $N[v]$ respectively. For any $S \subseteq V$, let $N(S) = \bigcup_{v \in S} N(v) \setminus S$ and $N[S] = N(S) \cup S$. The degree $d(v) = |N(v)|$ is the number of neighbours of v and let $\delta(G) = \min_{v \in V} d(v)$. An *independent set* of a graph $G = (V, E)$ is a subset I of V such that, for every $u, v \in I$, $uv \notin E$. A graph is *bipartite* if its vertex-set can be partitioned into two independent sets.

Hunters and Rabbit game. The Hunters and Rabbit game is played between two players, Hunter and Rabbit, on a graph. Let $k \in \mathbb{N}^*$. The Hunter player controls k hunters and the Rabbit player controls a single rabbit. First, the Rabbit player places the rabbit at a vertex $r_0 \in V$. The rabbit is *invisible*, i.e., the position of the rabbit is not known to the hunters. Then, the game proceeds in *rounds*. In each round $i \geq 1$, first, the Hunter player selects a non empty subset $S_i \subseteq V$ of at most k vertices of G (we say that the vertices in S_i

are *shot* at round i). If the current position r_{i-1} of the rabbit is shot, i.e., if $r_{i-1} \in S_i$ (we say that the rabbit is shot), then the Hunter player wins, and the game stops. Otherwise, the rabbit must move from its current position r_{i-1} to a vertex $r_i \in N(r_{i-1})$, and the next round starts. The Rabbit player wins if the rabbit avoids being shot forever.

A *hunter strategy* in $G = (V, E)$ is a finite sequence $\mathcal{S} = (S_1, \ldots, S_\ell)$ of non empty subsets of vertices of G. Let $h(\mathcal{S}) := \max_{1 \leq i \leq \ell} |S_i|$ and let us say that $\mathcal{S}$ *uses* $h(\mathcal{S})$ hunters. A *rabbit trajectory in G starting from $W \subseteq V$* (W will always be assumed non empty) is any walk $(r_0, \ldots, r_\ell)$ such that $r_0 \in W$ and $r_i \in N(r_{i-1})$ for every $1 \leq i \leq \ell$. A hunter strategy is *winning with respect to W* if, for every rabbit trajectory $(r_0, \ldots, r_\ell)$ starting from W, there exists $0 \leq j < \ell$ such that $r_j \in S_{j+1}$, i.e, the rabbit is eventually shot whatever be its trajectory starting from W. Given a hunter strategy $\mathcal{S} = (S_1, \ldots, S_\ell)$, a rabbit trajectory $(r_0, \ldots, r_\ell)$ starting from W is *winning against $\mathcal{S}$* if $r_i \notin S_{i+1}$ for every $0 \leq i < \ell$. A *winning hunter strategy* is any winning hunter strategy with respect to V and a *rabbit trajectory* is any rabbit trajectory starting from V.

The *hunter number of $G = (V, E)$ with respect to $W \subseteq V$*, denoted by $h_W(G)$, is the minimum integer k such that there exists a winning hunter strategy with respect to W and using k hunters. Let $h(G) = h_V(G)$ be the *hunter number* of G. The Rabbit player has a *strategy $\mathcal{R}$ starting from $W \subseteq V$ against $k \geq 1$ hunters* if, for every hunter strategy $\mathcal{S}$ using k hunters, there exists a rabbit trajectory $\mathcal{R}(\mathcal{S})$ that is winning against $\mathcal{S}$. If such a strategy $\mathcal{R}$ exists, then $h_W(G) > k$.

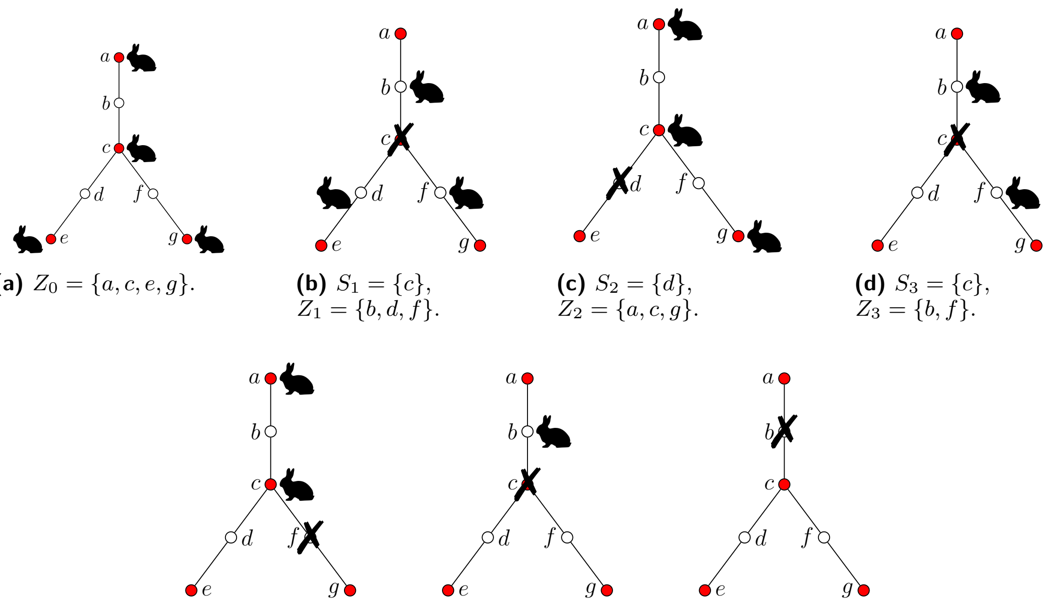
For any hunter strategy $\mathcal{S} = (S_1, \ldots, S_\ell)$, it will be convenient to identify the potential positions of a rabbit (starting in $W \subseteq V$) after each round. Precisely, let $\mathcal{Z}^W(\mathcal{S}) = (Z_0^W(\mathcal{S}), \ldots, Z_\ell^W(\mathcal{S}))$ be defined as follows. Let $Z_0^W(\mathcal{S}) = W$ and, for every $0 < i \leq \ell$, let $Z_i^W(\mathcal{S})$ be the set of vertices v such that there exists a rabbit trajectory $(r_0, r_1, \ldots, r_i = v)$ such that $r_0 \in W$ and, for every $0 \leq j < i$, $r_j \notin S_{j+1}$. Formally, for any $1 \leq i \leq \ell$, let $Z_i^W(\mathcal{S}) = \{x \in V(G) \mid \exists y \in (Z_{i-1}^W(\mathcal{S}) \setminus S_i) \wedge (xy \in E(G))\}$. Intuitively, $Z_i^W(\mathcal{S})$ is the set of vertices that the rabbit (starting from some vertex in W) can have reached at the end of the i^{th} round without having been shot. We will refer to the vertices in $Z_i^W(\mathcal{S})$ as the *contaminated* vertices after round i. Note that, if $\mathcal{S}$ is winning, then $Z_\ell^W(\mathcal{S}) = \emptyset$. In what follows, we write Z_i (resp., $Z_i(\mathcal{S})$) instead of $Z_i^W(\mathcal{S})$ when $\mathcal{S}$ and W (resp., when W) are clear from the context.

To reduce the search space for the possible hunter strategies, we establish that we can have winning hunter strategies that shoot only on a subset of contaminated vertices in each round, without increasing the number of hunters required. More precisely, a hunter strategy $\mathcal{S} = (S_1, \ldots, S_\ell)$ is said to be *parsimonious* if, for every $1 \leq i \leq \ell$, $S_i \subseteq Z_{i-1}(\mathcal{S})$.

3 Monotonicity

In classical graph pursuit-evasion games, an important notion is that of *monotonicity.* On a high level, a strategy is *monotone* if the area reachable by the fugitive never increases. In the particular case of Graph Searching games, a strategy is monotone if, once a searcher is removed from some vertex, it is never necessary to occupy this vertex during a subsequent round. Monotone strategies have been widely studied [3, 29, 24] because it is generally easier to design them and because they have length polynomial in the size of the graph and, so, corresponding decision problems can be proven to be in NP.

It is clear that such a definition is not suitable to the Hunters and Rabbit game. Indeed, consider the graph that consists of a single edge uv: the hunter must shoot at some vertex, say u, and, if the rabbit was at v, it will move to u, i.e., the vertex u is "recontaminated".



(a) $Z_0 = \{a, c, e, g\}$.

(b) $S_1 = \{c\}$, $Z_1 = \{b, d, f\}$.

(c) $S_2 = \{d\}$, $Z_2 = \{a, c, g\}$.

(d) $S_3 = \{c\}$, $Z_3 = \{b, f\}$.

(e) $S_4 = \{f\}$, $Z_4 = \{c, a\}$.

(f) $S_5 = \{c\}$, $Z_5 = \{b\}$.

(g) $S_6 = \{b\}$, $Z_6 = \emptyset$.

Figure 1 Example of a bipartite graph (where $V_r = \{a, c, e, g\}$ corresponds to the red part of the bipartition, illustrated by the red vertices in the figures) and of a parsimonious winning strategy with respect to V_r, such that no vertex in $\{a, e, g\}$ is ever shot. Each subfigure depicts the situation at the end of the corresponding round. The cross indicates the shot of the hunter, and all the possible positions of the rabbit are shown by a rabbit next to the corresponding vertex.

Therefore, we propose to define monotonicity in the Hunters and Rabbit game as follows (see the formal definition below): once a vertex has been "cleared", if the rabbit can access it in a subsequent round, then the vertex must be shot immediately.

In classical Graph Searching games, a vertex being cleared at some round means that the searcher's strategy ensures that the fugitive cannot occupy this vertex at this round. A recontaminated vertex can be intuitively defined as a vertex that can be reached by the fugitive while having been cleared in a previous round. This intuitive definition does not make any sense in the Hunters and Rabbit game. For example, it is shown in [1] that, in bipartite graphs, $h(G) = h_{V_r}(G) = h_{V_w}(G)$, where (V_r, V_w) denotes the bipartition of V. In other words, it is sufficient to consider winning hunter strategies with respect to one of the independent sets of the bipartition; we call this the *red variant* of the game. In this case, every vertex of V_r is cleared at every odd round (since the rabbit can only occupy vertices of V_w at odd rounds) and so looking for a strategy without recontamination would be meaningless.

A related difficulty comes from the fact that, contrary to classical Graph Searching games, a vertex may be "cleared" without having been shot during the game. As a concrete example, consider a star with three leaves whose edges have been subdivided once each. Then, assuming that the leaves and the centre are red, in the red variant, it is possible for one hunter to win without shooting any of the leaves. Indeed, consider the strategy illustrated in Figure 1.

To overcome these difficulties, we propose to define the clearing of a vertex at some round by the fact that the actions of the hunters ensure that this vertex cannot be occupied by the rabbit at this round. Precisely, two actions of the hunters may clear a vertex: either a hunter shoots a vertex v at round i and does not shoot the rabbit (meaning that $v \neq r_{i-1}$), or the hunters shoot on every contaminated vertex in the neighbourhood of v. In this case, either v was occupied and the rabbit has to leave v, or v was not occupied and cannot be occupied after the move of the rabbit. In both cases, $v \notin Z_i$. This discussion motivates the following definition for the monotonicity of hunter strategies.

▶ **Definition 1** (Monotone strategy). *Let G be a graph and $\mathcal{S}$ be a winning hunter strategy with respect to $W \subseteq V(G)$. We say that a vertex v is cleared at round i if either $v \in S_i$ or, $N(v) \cap Z_{i-1} \neq \emptyset$ and $N(v) \cap Z_{i-1} \subseteq S_i$. A strategy $\mathcal{S} = (S_1, \dots, S_\ell)$ is monotone if, for every vertex $v \in V$, if there exists an i such that v is cleared at round i, then for every $j > i$ such that $v \in Z_j$, the strategy ensures that $v \in S_{j+1}$. A vertex v is recontaminated at round j if there exists $i \leq j$ such that v is cleared at round i and $v \in Z_j \setminus S_{j+1}$.*

The *monotone hunter number* of a graph G with respect to $W \subseteq V(G)$, denoted by $mh_W(G)$, is the minimum number k such that k hunters have a monotone winning hunter strategy in G with respect to W. Let us denote the *monotone hunter number* $mh_V(G)$ of G by $mh(G)$. Note that, by definition, $h_W(G) \leq mh_W(G) \leq mh(G)$.

We can prove that monotone strategies have many interesting properties that are used in most of the proofs of our results. The proofs of these properties, which are omitted due to lack of space, are not trivial. The most intuitive one, is that, when the hunters follow a monotone strategy $\mathcal{S}$, the set of possible positions of the rabbit cannot increase. That is, $Z_\ell(\mathcal{S}) \subseteq \cdots \subseteq Z_1(\mathcal{S})$. Moreover, $mh(G)$ is closed under taking subgraphs. A crucial property is that there exists an optimal (using $mh(G)$ hunter) parsimonious monotone strategy in any graph G. Finally, in any monotone strategy, once a vertex is shot, it has to be continuously shot until the rabbit can no longer reach it. In particular, this last property is used in the proof of upcoming Theorem 2 and it implies that the problem of computing mh is in NP.

3.1 Monotone hunter number and pathwidth

Here, we relate the monotone hunter number of a graph to its pathwidth. Our result might be surprising since the pathwidth of a graph G is equivalent to the number of searchers required to (monotonously) capture an arbitrary fast invisible fugitive [3] while, in our case, the invisible rabbit seems much weaker than the fugitive: the rabbit is “slow” (it moves only to neighbours) and constrained to move at every round. In this view, one might guess that the monotone hunter number of a graph could be arbitrary smaller than its pathwidth. On the contrary, we show that these parameters differ by at most one.

A *path-decomposition* of a graph $G = (V, E)$ is a sequence $P = (X_1, \dots, X_p)$ of subsets of vertices, called *bags*, such that (1) $\bigcup_{i \leq p} X_i = V$; (2) for every $uv \in E$, there exists $i \leq p$ with $\{u, v\} \subseteq X_i$; and (3): for every $1 \leq i \leq j \leq q \leq p$, $X_i \cap X_q \subseteq X_j$. The *width* $w(P)$ of P is the size of a largest bag of P minus one, i.e., $w(P) = \max_{i \leq p} |X_i| - 1$. The *pathwidth* $pw(G)$ of G is the minimum width of its path-decompositions. A path-decomposition of G of width $pw(G)$ is said to be *optimal*.

▶ **Theorem 2.** *For any graph $G = (V, E)$, $pw(G) \leq mh(G) \leq pw(G) + 1$.*

Sketch of proof. Let $P = (X_1, \dots, X_\ell)$ be a path-decomposition of G with width k. Then, P is a monotone hunter strategy in G using $k + 1$ hunters.

To show the other inequality, let $\mathcal{S} = (S_1, \dots, S_\ell)$ be a parsimonious winning monotone hunter strategy in G using at most $k \geq mh(G)$ hunters. Observe that $\mathcal{S}$ is almost a path-decomposition due to the fact that it is parsimonious and monotone. Indeed, any vertex v that is shot by $\mathcal{S}$, will be shot during some consecutive rounds (will belong to consecutive bags of the decomposition). It remains to take care of unshot vertices. Such vertices are cleared by shooting at all of their neighbours during a round of $\mathcal{S}$. So, starting from $\mathcal{S}$, we build a path-decomposition $\mathcal{P}$ of G as follows: we start with a bag B_i corresponding to S_i, for each i. Then, for each vertex u that has never been shot, all the neighbours of which are shot during the round $i+1$ for the first time, we create an intermediate bag, between B_i and B_{i+1}, containing all the vertices of B_i and the vertex u. ◀

Theorem 2 has important consequences, following from the inapproximability of the pathwidth of a graph [4]. Moreover, using a result in [17], this implies that recontamination may help in the Hunters and Rabbit game.

▶ **Corollary 3.** *Given an n-node graph G and $k \in \mathbb{N}$, it is NP-complete to decide whether $mh(G) \leq k$. Moreover, it is NP-hard to approximate $mh(G)$ up to an additive error of n^{ε}, for $0 < \varepsilon < 1$.*

▶ **Corollary 4.** *There exists $\varepsilon > 0$ such that, for any $k \in \mathbb{N}$, there exists a tree T with $h(T) \geq k$ and $mh(T) \geq (1+\varepsilon)h(T)$.*

4 (Monotone) hunter number of some graph classes

In this section, we characterise the (monotone) hunter number of several graph classes such as cographs, split and interval graphs. In all these cases, our results lead to a polynomial time algorithm to compute the monotone hunter number.

4.1 Split and interval graphs

A graph $G=(V,E)$ is a *split graph* if $V = C \cup I$ can be partitioned into two sets C and I, inducing an inclusion-maximal clique and an independent set, respectively. Given a split graph G, a partition (C,I) of $V(G)$ can be computed in linear time [18]. The following theorem fully characterises the hunter number of split graphs. It also allows us to show that the hunter number and the pathwidth of split graphs coincide.

▶ **Theorem 5.** *Let $G = (C \cup I, E)$ be a split graph. Then, $h(G) = |C|$ if and only if for every two distinct vertices $x, y \in C$, there exists a vertex $z \in N_I(x) \cap N_I(y)$. Otherwise, $h(G) = |C| - 1$.*

Sketch of proof. If every two vertices of C have a common neighbour in I, there exists a rabbit strategy against $|C|-1$ hunters. The idea is to take advantage of the fact that not all the vertices of C can be shot during a same round. Thus, the rabbit will remain as much as possible on C, and if it is unable to do so, it will go to I and then return to C, which is possible due to the hypothesis. For the reverse direction, let x and y be two vertices of C without common neighbour in I. The strategy $\mathcal{S} = (S_1, S_2, S_3, S_4, S_5)$, where $S_1 = S_2 = S_5 = C \setminus \{y\}$ and $S_3 = S_4 = C \setminus \{x\}$, is a winning hunter strategy using $|C|-1$ hunters. ◀

Recall that a vertex in a graph G is *simplicial* if its neighbourhood induces a clique. Recall also that an *interval graph* is the intersection graph of a set of intervals in the real line. Let $\omega(G)$ denote the maximum size of a clique of G.

▶ **Theorem 6.** *Let G be a interval (resp. split) graph. Then, $h(G) = mh(G) = \omega(G) - 1$ (resp. $mh(G) = \omega(G) - 1$) if every maximum clique has a simplicial vertex. Otherwise, $mh(G) = \omega(G)$.*

Sketch of proof. For the first direction, in case $G = (C \cup I, E)$ is a split graph with v being a simplicial vertex of C, $S = (C \setminus v, C \setminus v)$ is a monotone winning hunter strategy using $|C| - 1$ hunters. In case G is an interval graph, recall that $pw(G) = \omega(G) - 1$ and G admits an optimal path-decomposition where each bag induces a complete graph. We adapt such an optimal path-decomposition, to a hunter strategy using $\omega(G) - 1$ hunters, by removing a simplicial vertex from each bag containing a maximum clique and shooting twice at each of these bags.

For the reverse direction, let C be any maximum clique of G without simplicial vertex. Assume that there exists a monotone hunter strategy $\mathcal{S}$ using $|C| - 1$ hunters. Then there exists at least one round such that C hunters shoot on vertices of C (otherwise the rabbit can survive by staying on C). Consider the first such round and let u be the vertex of C that is not shot. Using the fact that C has no simplicial vertex, we prove that there exists $w \in N(u) \setminus C$ such that both u and w have never been shot until this round. Thus, the rabbit can oscillate between u and w either until the end, or until the first round where at least one of u or w is shot, where it recontaminates a vertex of C. This is a contradiction. ◀

From the above, we can show that there exist split and interval graphs G for which $mh(G) \neq h(G)$, i.e., recontamination helps in these graph families. Note also that, even knowing that $\omega(G) - 1 \leq h(G) \leq \omega(G)$ for interval graphs, computing $h(G)$ when some maximum clique has no simplicial vertex is challenging.

4.2 Cographs

The class of *cographs* can be defined recursively as follows [10]. One vertex is a cograph. Given two cographs A and B, their disjoint union $A \cup B$ is a cograph, and their join $A \bowtie B$ (where all edges between A and B are added) is a cograph. A decomposition of a cograph (*i.e.*, a building sequence of unions and joins performed from single vertices) can be computed in linear time [10].

▶ **Theorem 7.** *$mh(G)$ can be computed in linear time in the class of cographs.*

Sketch of proof. It suffices to calculate $mh(G)$ in the case where $G = A \bowtie B$. It is easy to see that $mh(A \bowtie B) \leq m = min(mh(A) + |V(B)|, |V(A)| + mh(B))$. Assume that there exists a winning monotone hunter strategy $\mathcal{S} = (S_1, \ldots, S_\ell)$ using at most $m - 1$ hunters. Let v be the first vertex that is no longer available for the rabbit and assume, w.l.o.g., that $v \in V(A)$. Then, all the vertices in $N_B(v) = V(B)$ were shot. Thus, $V(B)$ must be shot during every subsequent round. This means that $\mathcal{S}$ can clear A using $m - |V(B)| - 1 < mh(A)$ hunters, a contradiction. ◀

Once again, the case of the hunter number seems more challenging. In particular, the following lemma shows that recontamination may help in cographs.

▶ **Lemma 8.** *For every $k \geq 1$, there exists a cograph G such that $h(G) \geq k$ and $mh(G) \geq \frac{3}{2}h(G) - 1$.*

Sketch of proof. Let A and B be two (isomorphic) cographs consisting of the disjoint union of a complete graph with $a \geq 3$ vertices and a independent vertices. The graph $G = A \bowtie B$ verifies that $mh(G) = 3a - 1$ and $h(G) = 2a$. ◀

5 (Monotone) hunter number of trees

This section is devoted to showing that mh can be computed in polynomial-time in the class of trees. Then, we show that recontamination helps a lot in trees.

5.1 Monotone hunter number in trees

Let T be a tree and $v \in V(T)$. A *branch* at v is any connected component of $T - v$. A *star* is any tree with at least two vertices and at most one vertex with degree at least three. Roughly, Parsons' Lemma [26] states that, for any tree T and $k \in \mathbb{N}$, $pw(T) \geq k+1$ if and only if there exists a vertex v such that at least three branches at v have pathwidth at least k. Here, we adapt this lemma in the case of the monotone hunter number of trees.

▶ **Lemma 9** (Parsons' like lemma). *Let $T = (V, E)$ be any tree.*

- *$mh(T) = 0$ if and only if $|V| = 1$;*
- *$mh(T) = 1$ if and only if T is a star;*
- *$mh(T) = 2$ if and only if T is not a star and contains a path P such that $T \setminus P$ is a forest of stars and isolated vertices;*
- *For every $k \geq 3$, $mh(T) \geq k$ if and only if there exists a vertex $v \in V$ such that at least three branches at v have monotone hunter number at least $k-1$.*

Taking advantage of Lemma 9, we design a dynamic programming algorithm to compute $mh(T)$ of a given tree T. Our algorithm is heavily inspired by the polynomial time algorithm computing the pathwidth of T [13].

▶ **Theorem 10.** *For a tree T, $mh(T)$ can be computed in polynomial time.*

5.2 Monotone hunter number in the red variant in trees

So far, we have investigated the monotone Hunters and Rabbit, since monotone strategies are often easier to deal with. Previous works on the Hunters and Rabbit game in bipartite graphs $G = (V_r \cup V_w, E)$ have shown that studying the *red variant* of the Hunters and Rabbit game, i.e., when the rabbit is constrained to start in a vertex in V_r, could be very fruitful. For instance, $h(G) = h_{V_r}(G)$ holds for every bipartite graph $G = (V_r \cup V_w, E)$, which helped to get many results on the Hunters and Rabbit game [1, 5, 17]. Therefore, it is interesting to consider the monotonicity constraint when restricted to the red variant of the Hunters and Rabbit game. We now focus on investigating the difference between $h_{V_r}(T)$ and $mh_{V_r}(T)$, for any tree T. Thankfully, all the useful properties we observed for the monotone version can be adapted for the monotone red variant as well.

By Corollary 4, there exists $\varepsilon > 0$ such that, for any $k \in \mathbb{N}$, there exists a tree T with $h(T) \geq k$ and $mh(T) \geq (1+\varepsilon)h(T)$. In the following theorem we improve this, by showing that there exists an infinite family of trees T such that the difference between $mh(T)$ and $h(T)$ is arbitrarily large. This follows from the next theorem since $h_{V_r}(T) = h(T)$ and $mh(T) \geq mh_{V_r}(T)$ for any tree T. Its proof follows from Lemmas 12 and 13, presented below.

▶ **Theorem 11.** *For every $i \geq 3$, there exists a tree T such that $mh_{V_r}(T) \geq i$ and $h_{V_r}(T) = 2$.*

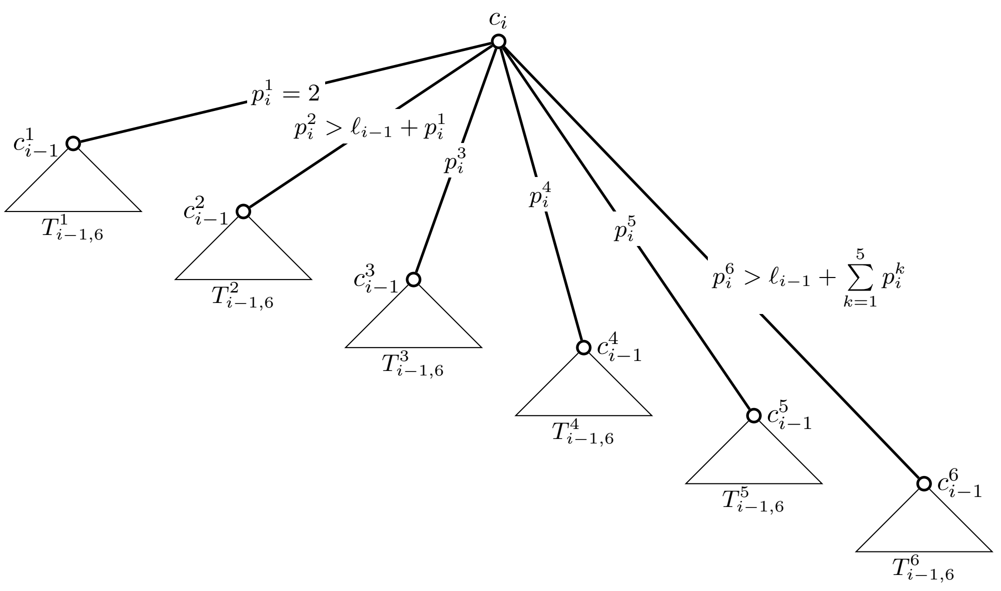


Figure 2 The graph $T_{i,6}$. The labels on the edges are used to represent their respective lengths. In particular, for every $2 \leq j \leq 6$, we have that $p_i^j > \sum_{1 \leq k \leq j-1} p_i^k + \ell_{i-1}$, where $p_i^1 = 2$ and ℓ_{i-1} is equal to the number of turns needed to clear any copy of the $T_{i-1,6}$ graph.

The construction of the tree $T_{i,q}$. Let $S_{k,q}$ be the rooted tree obtained from $q \geq 6$ paths of length $k \geq 3$ (with k edges) by identifying an endpoint of each path into a common vertex called the *root* of $S_{k,q}$ and denoted by c. From now on, let (V_r, V_w) be the bipartition of $V(S_{k,q})$ and assume that $c \in V_r$. Let $\mathcal{S}_1$ be a winning hunter strategy such that $mh_{V_r}(S_{3,q}) = 2$ [7], and let ℓ_1 be the smallest even integer greater or equal to the length of $\mathcal{S}_1$.

For every $i \geq 2$ and $q \geq 6$, let $T_{i,q}$ be the tree recursively built as follows (see Figure 2). First, $T_{1,q} = S_{3,q}$. Then, for $i > 1$, we assume that $T_{i-1,q}$ has been defined recursively and that there exists a winning hunter strategy, of length ℓ_{i-1}, using 2 hunters in the red variant in $T_{i-1,q}$. $T_{i,q}$ is obtained from q vertex disjoint copies $T_i^1, \dots, T_i^q$ of $T_{i-1,q}$ and from a vertex c_i, the root of $T_{i,q}$. Then, for every $1 \leq j \leq q$, add a path P_i^j of length p_i^j between the root c_i^j of T_i^j and c_i (that is, c_i and c_i^j are at distance p_j^i in $T_{i,q}$). The lengths p_i^j are defined recursively as follows. Let $p_i^1 = 2$ and, for every $1 < j \leq q$, let p_i^j be the minimum even integer greater or equal to $\ell_{i-1} + \sum_{1 \leq k < j} p_i^k$. Finally, let us assume that $c_i \in V_r$ and note that, since p_i^j is even, this implies that $c_i, c_i^1, \dots, c_i^q$ all belong to V_r.

▶ **Lemma 12.** *For any $i \in \mathbb{N}^*$ and $q \geq 6$, $h_{V_r}(T_{i,q}) = 2$.*

Sketch of proof. It suffices to give a winning hunter strategy using 2 hunters. The strategy $\mathcal{S}^i$ is defined recursively, and consists of q phases. During the j^{th} phase, for $1 \leq j \leq q$, one hunter shoots at c_i, while the other "pushes" the rabbit toward the subtrees T_i^q, then T_i^{q-1}, and so on, until T_i^j. Then, the two hunters clear the subtree T_i^j, following the strategy $\mathcal{S}^{i-1}$ (without the rabbit being able to leave T_i^j if it was there). The lengths of the paths linking the roots of the subtrees to c_i guarantee that the rabbit cannot reach c_i before T_i^j has been cleared. ◀

As a consequence, the hunter number is not closed under taking a minor.

▶ **Lemma 13.** *For $i \geq 3$ and $q \geq 2i$, $mh_{V_r}(T_{2i,q}) \geq i$.*

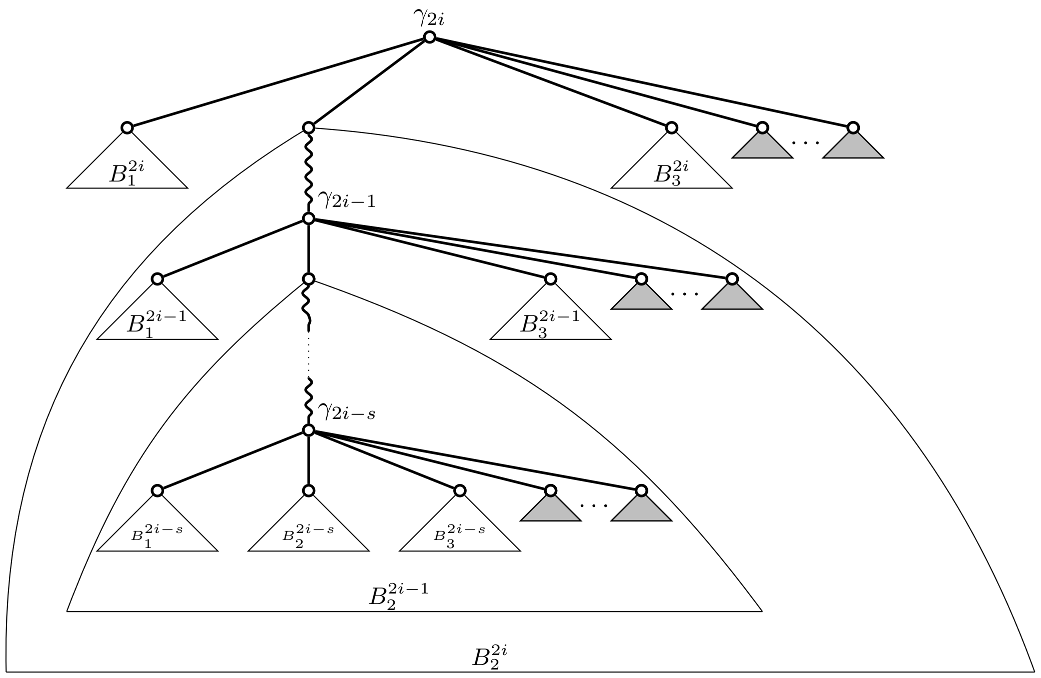


Figure 3 A representation of the tree $T_{2i,d}$. Wiggly edges represent paths whose internal vertices have degree 2.

Sketch of proof. Let γ_{2i} denote the root of $T_{2i,q}$ (see Figure 3) and assume that $mh_{V_r}(T_{2i,q}) \leq i-1$. Let $B_1^{2i}, \dots, B_q^{2i}$ denote the branches of γ_{2i}. The main ingredient of our proof is that q is large enough so that when, say during round j_{2i}, the first branch of γ_{2i}, say B_1^{2i}, is definitively cleared according to any monotone hunter strategy (i.e. the hunters will never shoot a vertex in B_1^{2i} for the remaining of the game), then there are at least two other branches of γ_{2i}, say B_2^{2i} and B_3^{2i}, whose vertices have never been shot. Note that γ_{2i} must be shot during the round j_{2i} or $j_{2i}+1$. Then, the same arguments can be stated recursively on the tree $T_{2i-1,q}$, contained in the first branch among B_2^{2i} and B_3^{2i} that will be definitively cleared. In addition, we prove that $j_{2i} < \cdots < j_1$ and every γ_q (or some vertex of B_2^q or B_3^q), for $1 \leq q \leq 2i$, must be shot during the round j_1 or j_1+1. Thus, we obtain that at least i vertices are shot during the round j_1 or j_1+1, a contradiction. ◀

6 Kernelization by vertex cover

Two instances I and I' are *equivalent* when I is a Yes-instance if and only if I' is a Yes-instance. A *kernelization algorithm* of a parameterized problem Π is a polynomial time algorithm that maps each instance (I,k) of Π to an equivalent instance (I',k') of Π such that the size of (I',k') is bounded by $g(k)$ for some computable function $g(\cdot)$. Here, (I',k') is said to be a *kernel*. It is well-known that a parameterized problem Π is FPT if and only if it admits a kernel. We refer to the book [11] for details on parameterized complexity. We remark that similar techniques were used to obtain kernelization algorithms for several variants of Cops and Robber game parameterized by $vc(G)$ [14].

Recall that a set U is a *vertex cover* of a graph $G=(V,E)$ if $G[V \setminus U]$ is an independent set. For the rest of this section, let U be a vertex cover of size t and I be the independent set $G[V \setminus U]$. If no such vertex cover is given, we can compute a vertex cover using a 2-approximation algorithm [28].

▶ **Theorem 14.** *Let G be a graph having a vertex cover U of size t, and let $k \in \mathbb{N}$. Deciding whether $h(G) \leq k$ (resp., $mh(G) \leq k$) is* FPT *parameterized by t. More specifically, these problems admit a kernel with at most $4^t(t+1) + 2t$ vertices.*

Sketch of proof. First, observe that $mh(G) \leq t$ since the hunters have a monotone winning strategy by shooting the vertices of U twice. Second, we argue that the standard rule of removing twins from I leads to an exponential kernel for these questions. More specifically, we have the following reduction rule along with the rule that if $k \leq t$, return a Yes-instance: If there is some $I' \subset I$ such that $|I'| > k+1$ and for any two vertices $x, y \in I'$, $N(x) = N(y)$, then delete an arbitrary vertex, say v, from I' to get G' and let $k' = k$. It is not difficult to see that this will give us a kernel with at most $2^t(t+1) + t$ vertices.

Next, we give an intuition regarding the safeness of our reduction rule. It is easy to see that if (G, k) is a Yes-instance, then (G', k) is also a Yes-instance as both $h(G)$ and $mh(G)$ are closed under taking subgraphs. For the reverse direction, and towards a contradiction, we will assume that (G, k) is a no-instance but (G', k) is a yes-instance. Since (G', k) is a yes instance, there exists a winning hunter strategy $\mathcal{S}'$ using at most k hunters in G'. Since (G, k) is a no-instance, there exists a rabbit strategy R winning against $\mathcal{S}'$ in G. From R, we can design R', an equivalent rabbit strategy winning against $\mathcal{S}'$ in G', contradicting that (G', k) was a yes-instance. ◀

7 Some Future Directions

In this paper, we studied the Hunters and Rabbit game by defining the notion of monotonicity for this game. Using this notion of monotonicity, we characterised the monotone hunter number for various classes of graphs. Moreover, we established that, unlike several Graph Searching games, the monotonicity helps in this game, i.e., $h(G)$ can be arbitrary smaller than $mh(G)$.

There are still several challenging open questions in this area. The most important among them is the computational complexity of Hunters and Rabbit. Although our results establish that computing $mh(G)$ is NP-hard, the computational complexity of computing/deciding $h(G)$ remains open, even if G is restricted to be a tree.

We also established that both Hunters and Rabbit, as well as its monotone variant, are FPT parameterised by $vc(G)$ by designing exponential kernels. It is not difficult to see that both of these variants admit AND Composition parameterised by the solution size (by taking the disjoint union of the instances). Thus, since computing $mh(G)$ is NP-hard and $pw(G) \leq mh(G) \leq pw(G) + 1$, it is unlikely for Monotone Hunters and Rabbit parameterized by $k + pw(G)$ to admit a polynomial compression. Note that the same cannot be argued about Hunters and Rabbit since it is not yet proved to be NP-hard. Moreover, since $mh(G)$ is closely related to $pw(G)$ and pathwidth admits a polynomial kernel with respect to $vc(G)$ [8], it might be interesting to see if deciding $mh(G) \leq k$ (resp., $h(G) \leq k$) also admits a polynomial kernel when parameterised by $vc(G)$. Moreover, another interesting research direction is to study the parameterised complexity of both these games by considering parameters such as solution size, treewidth, and pathwidth.

Finally, we propose some open questions concerning the computation of $h(G)$ for various graph classes including trees, cographs, and interval graphs. Specifically, it will be interesting to design a polynomial time algorithm to compute $h(T)$ for a tree T, a challenge that was already proposed in [1]. The natural way that one could tackle this question is through the notion of monotonicity, which we defined and studied in this paper. Unfortunately, Theorem 11 implies that such an approach will not work. This means that a positive answer to this question (if any) would require the introduction of new tools and techniques. Moreover, it would be interesting to know the monotone hunter number of grids.

References

1 T. V. Abramovskaya, F. V. Fomin, P. A. Golovach, and M. Pilipczuk. How to hunt an invisible rabbit on a graph. *European Journal of Combinatorics*, 52:12–26, 2016.

2 I. Adler. Marshals, monotone marshals, and hypertree-width. *Journal of Graph Theory*, 47(4):275–296, 2004.

3 D. Bienstock and P. D. Seymour. Monotonicity in graph searching. *Journal of Algorithms*, 12(2):239–245, 1991.

4 H. L. Bodlaender, J. R. Gilbert, H. Hafsteinsson, and T. Kloks. Approximating treewidth, pathwidth, frontsize, and shortest elimination tree. *Journal of Algorithms*, 18(2):238–255, 1995.

5 J. Bolkema and C. Groothuis. Hunting rabbits on the hypercube. *Discrete Mathematics*, 342(2):360–372, 2019.

6 R. L. Breisch. An intuitive approach to speleotopology. *Southwestern cavers*, 6(5):72–78, 1967.

7 J. R. Britnell and M. Wildon. Finding a princess in a palace: a pursuit-evasion problem. *The Electronic Journal of Combinatorics*, 20(1):25, 2013.

8 M. Chapelle, M. Liedloff, I. Todinca, and Y. Villanger. Treewidth and pathwidth parameterized by the vertex cover number. *Discrete Applied Mathematics*, 216:114–129, 2017.

9 T. Chung, G. A. Hollinger, and V. Isler. Search and pursuit-evasion in mobile robotics: A survey. *Autonomous Robots*, 31(299):299–316, 2011.

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

11 M. Cygan, F. V. Fomin, L. Kowalik, D. Lokshtanov, D. Marx, M. Pilipczuk, M. Pilipczuk, and S. Saurabh. *Parameterized Algorithms*. Springer Publishing Company, Incorporated, 1st edition, 2015.

12 T. Dissaux, F. Fioravantes, H. Galhawat, and N. Nisse. Further results on the Hunters and Rabbit game through monotonicity. Technical report, Inria - Sophia Antipolis, February 2023. URL: `https://hal.science/hal-03995642`.

13 J. A. Ellis, I. H. Sudborough, and J. S. Turner. The vertex separation and search number of a graph. *Information and Computation*, 113(1):50–79, 1994.

14 H. Gahlawat and M. Zehavi. Parameterized analysis of the cops and robber game. In Bérôme Leroux, Sylvain Lombardy, and David Peleg, editors, *48th International Symposium on Mathematical Foundations of Computer Science (MFCS 2023)*, Leibniz International Proceedings in Informatics (LIPIcs), pages 47:1–40:15, Dagstuhl, Germany, 2023.

15 A. C. Giannopoulou, P. Hunter, and D. M. Thilikos. Lifo-search: A min-max theorem and a searching game for cycle-rank and treedepth. *Discrete Applied Mathematics*, 160(15):2089–2097, 2012.

16 G. Gottlob, N. Leone, and F. Scarcello. Robbers, marshals, and guards: game theoretic and logical characterizations of hypertree width. *Journal of Computer and System Sciences*, 66(4):775–808, 2003.

17 V. Gruslys and A. Méroueh. Catching a mouse on a tree. *arXiv preprint*, 2015. `arXiv:1502.06591`.

18 P. L. Hammer and B. Simeone. The splittance of a graph. *Combinatorica*, 1:275–284, 1981.

19 J. Haslegrave. An evasion game on a graph. *Discrete Mathematics*, 314:1–5, 2014.

20 D. Ilcinkas, N. Nisse, and D. Soguet. The cost of monotonicity in distributed graph searching. *Distributed Computing*, 22(2):117–127, 2009.

21 A. Isaza, J. Lu, V. Bulitko, and R. Greiner. A cover-based approach to multi-agent moving target pursuit. In *proceedings of the Fourth Artificial Intelligence and Interactive Digital Entertainment Conference*, pages 54–59. AAAI Press, 2008.

22 T. Johnson, N. Robertson, P. D. Seymour, and R. Thomas. Directed tree-width. *Journal of Combinatorial Theory Series B*, 82(1):138–154, 2001.

23 A. S. LaPaugh. Recontamination does not help to search a graph. *Journal of the ACM*, 40(2):224–245, 1993.

24 F. Mazoit and N. Nisse. Monotonicity of non-deterministic graph searching. *Theoretical Computer Science*, 399(3):169–178, 2008.

25 N. Nisse. Network decontamination. In Paola Flocchini, Giuseppe Prencipe, and Nicola Santoro, editors, *Distributed Computing by Mobile Entities, Current Research in Moving and Computing*, volume 11340 of *Lecture Notes in Computer Science*, pages 516–548. Springer, 2019.

26 T. D. Parsons. Pursuit-evasion in a graph. In *Theory and Applications of Graphs, LNCS*, volume 642, pages 426–441. Springer, 1978.

27 P. D. Seymour and R. Thomas. Graph searching and a min-max theorem for tree-width. *Journal of Combinatorial Theory, Series B*, 58(1):22–33, 1993.

28 D. P. Williamson and D. B. Shmoys. *The Design of Approximation Algorithms*. Cambridge University Press, 2011. URL: `http://www.cambridge.org/de/knowledge/isbn/item5759340/?site_locale=de_DE`.

29 B. Yang, D. Dyer, and B. Alspach. Sweeping graphs with large clique number. *Discrete Mathematics*, 309(18):5770–5780, 2009.