Randomized Pursuit-Evasion with Limited Visibility

Volkan Isler[*] Sampath Kannan[†] Sanjeev Khanna[‡]

Abstract

We study the following pursuit-evasion game: One or more hunters are seeking to capture an evading rabbit on a graph. At each round, the rabbit tries to gather information about the location of the hunters but it can see them only if they are located on adjacent nodes. We show that two hunters suffice for catching rabbits with limited visibility with high probability. We distinguish between reactive rabbits who move only when the hunter is visible and general rabbits who can employ more sophisticated strategies. We present polynomial time algorithms that decide whether a graph G is hunter-win, that is, if a single hunter can capture a rabbit of either kind on G.

1 Introduction

Pursuit-evasion games are problems of fundamental interest in many diverse fields such as computer-science, operations-research, game theory and control theory. The goal of a pursuit-evasion game is to find a strategy for a pursuer trying to catch an evader who, in turn, tries to avoid capture indefinitely. The game has many variations based on the environment in which it is played (E.g. plane, grid, graph), the information available to the the players (Do they know each others' positions all the time? Does the pursuer know evader's strategy?), the controllability of their motion (Is there a bound on their speed? Can they turn whenever they want?), and the meaning of a *capture* (whether the evader must be intercepted, seen or surrounded).

Earlier studies of pursuit-evasion were motivated by control tasks such as intercepting missiles [4]. The problem is addressed in the robotics community for its applications in collision avoidance, search and rescue, and air-traffic control [10, 9]. In these models typically the motion of the evader is modeled by a stochastic process. However, recently there has been increasing interest in modeling games where the evader is more "intelligent" and has certain sensing capabilities [18]. Pursuit-evasion games on graphs [17, 15, 13, 12, 6, 1] have been studied not only for their applications in network security and protocol design (e.g. [3, 11]) but also for their relations to fundamental properties of graphs such as vertex separation [7]. A remark about the terminology: In the literature, the names pursuer-evader, cop-robber, monster-princess, hunter-rabbit, sheriff-thief have been used somewhat synonymously. We adopt the hunter-rabbit term for it emphasizes the discrete nature of the game [5, 1].

In this paper, we address a different aspect of the problem that has not received much attention so far. We study the relationship between the information available to the rabbit and the conditions to capture it. The basic model of our game is as follows: The players are located on the nodes of a graph. At every time step, they move to nodes in their neighborhoods (which includes the current node) simultaneously. We say a rabbit is *caught* or *captured* if at the beginning of a time step it occupies the same node as a hunter. We associate the information available to the rabbit with its visibility. If the rabbit has complete information about the location of the hunter(s) during the entire game, we say the rabbit has *full visibility.* On the contrary, if the rabbit has no information about the hunters, then we say it has *no visibility.*

In our present work, we study the game when the rabbit has *limited visibility.* That is, it can only see the nodes that are adjacent to its current location. When the hunter is located at an adjacent node, the rabbit has complete information about his location. However, if the hunter is not visible, then the rabbit must infer the hunter's location based on the time and location of their last encounter. Note that this model is different from the "visibility-based pursuit evasion" work [9, 16], where the goal is to eventually "see" an evader which has complete visibility and unbounded speed.

Recently, Adler et al. studied this game when the rabbit has no visibility [1]. They showed that a single hunter can catch the rabbit on any (connected) graph.

[*]Department of Computer and Information Science, University of Pennsylvania, Philadelphia, PA, USA. Email: isleri@cis.upenn.edu. Supported in part by NSF-IIS-0083209, NSF-IIS-0121293 and MURI DAAH-19-02-1-03-83

[†]Department of Computer and Information Science, University of Pennsylvania, Philadelphia, PA, USA. Email: kannan@cis.upenn.edu. Supported in part by NSF Grants CCR0105337 and CCR9820885.

[‡]Department of Computer and Information Science, University of Pennsylvania, Philadelphia, PA, USA. Email: sanjeev@cis.upenn.edu. Supported in part by an Alfred P. Sloan Research Fellowship and an NSF Career Award CCR-0093117.

The full visibility version has also been studied [15, 6]. It is known that under the full-visibility model, the class of graphs on which a single hunter suffices is the class of dismantlable graphs. The number of hunters necessary to capture the rabbit on a graph G is known as the cop (hunter) number of G. It is known that [2] the cop number of planar graphs is at most 3 but the cop number of general graphs is still an open question [14, 8].

We will focus on randomized strategies. The previous body of work for the full visibility case [15, 6, 14, 8, 2] derandomized the game by forcing the players to move in turns, rabbit followed by the hunter. Note that when the players move simultaneously, the game is not well-defined for deterministic strategies: Suppose the game is played on a complete graph. In this case it is easy to see that a single hunter can catch the rabbit simply by guessing its location in the next turn. However, if the hunter's strategy is deterministic, knowing it, the rabbit would never get caught. Similarly, the hunter could always catch the rabbit in a single move if he knew its strategy.

Our results and techniques: Our main result is an algorithmic characterization for the limited visibility case. We show that two hunters always suffice on general graphs and present a polynomial time procedure that decides whether a single hunter is sufficient to capture the rabbit on an input graph G. In order to obtain an efficient decision procedure, we establish that the uncertainty in rabbit's knowledge of hunter's location satisfies an interesting monotonicity property. This monotonicity property turns out to be crucial for obtaining a polynomial time characterization.

In the winning strategy for two hunters, a central component is to have one hunter mainly focus on keeping the rabbit on the move. This motivated us to study a natural class of *reactive* rabbit strategies, where the rabbit moves only when the hunter is in its sight. We show that the class of hunter-win graphs (i.e graphs on which a single hunter suffices) against general rabbits is strictly smaller than the class of hunter-win graphs against reactive rabbits. We present a characterization algorithm for reactive rabbits as well.

The characterization algorithms presented mark pairs of vertices according to certain rules, where the pairs correspond to players' positions. To understand the corresponding hunter strategies on hunter-win graphs, we first present a hunter strategy for the full visibility case. Next, we show that omitting one of the rules from the characterization algorithms yields an algorithm that recognizes graphs that are hunter-win against rabbits with full visibility. Using these two results, we show how the hunter exploits the limited visibility if the game is played on a graph G such that on G, the hunter can win against a rabbit with limited visibility but not against a rabbit with full visibility.

We note that when the rabbit's visibility is extended to distance 2, there exist graphs for which $\tilde{\Omega}(\sqrt{n})$ hunters are necessary.

Organization of the paper: The paper is organized as follows: In Section 2, we review necessary concepts that will be used throughout the paper. In Section 3 we present a winning strategy for two hunters on general graphs. Next, we study the graphs on which a single hunter suffices, both for reactive (Section 4.1) and general rabbits (Section 4.2). Section 5 is dedicated to the study of hunter strategies on hunter-win graphs. A gap example distinguishing the power of the two types of rabbit strategies is also presented in Section 5. We conclude the paper with a discussion on extensions of our work.

2 Preliminaries

Throughout the paper, we use the notation $N(v)$ to denote the set of vertices that are adjacent to v and we always assume that $v \in N(v)$. Unless otherwise stated, n denotes the number of vertices.

The game we study is formally defined as follows: It is played in rounds. In the beginning of a round, suppose a player (either a hunter or a rabbit) is located at vertex v. First, the player checks $N(v)$ and if there is another player located at a vertex $u \in N(v)$, this information is revealed to the player. In this case we say the two players *see* each other. Next, all the players make a decision about where to move and choose a vertex in their neighborhoods. At the end of the round, all players move to their chosen vertex simultaneously. A hunter *catches* the rabbit if they are located on the same vertex.

A *reactive rabbit strategy* is a rabbit strategy where the rabbit is not allowed to move from a vertex v unless the hunter is in $N(v)$. A rabbit strategy is *general* if it is not reactive. In other words, the rabbit can move even if the hunter is not visible. A *(resp. non-)reactive rabbit* is a rabbit that employs a (resp. non-)reactive strategy. A graph G is *hunter-win against reactive rabbits* if there exists a hunter strategy that catches any reactive rabbit on G with non-zero probability for all possible starting configurations. A graph that is *hunter-win against general rabbits* is defined similarly.

Configuration versus state: For a single hunter game, a *configuration* refers to an ordered pair (h, r) which corresponds to the locations of the hunter and the rabbit respectively. Note that this information may not be available to the rabbit at all times due to its limited visibility. A configuration (h, r) is *adjacent* if $h \in N(r)$. We use the notation $< H, r >$ to denote the *state* of the game where r is the location of the rabbit and H corresponds to the set of vertices where the hunter can possibly be located. For the full visibility case, if the current configuration is (h, r), the state is $< \{h\}, r >$. For the zero visibility case, the state is either $< G - \{r\}, r >$ or $< \{r\}, r >$. For the limited visibility case that we study, state has a more complex structure, and it evolves over time even when neither the hunter nor the rabbit is in motion.

Suppose u and v are two nodes of a graph G such that $N(u) \subseteq N(v)$. Then, the operation of deleting u from G is called a *folding* of G and we say *u folds onto v*. A graph is called *dismantlable* if there is a sequence of folds reducing it to a single vertex. We say *u eventually folds onto v*, if there is a sequence $u_0 = u, u_1, \ldots, u_k = v$ such that u_i folds onto u_{i+1}, $0 \leq i < k$. Let G be a dismantlable graph and ψ be a folding sequence reducing G to a single vertex v. We can visualize ψ as a tree T whose vertices are the vertices of G such that when rooted at v every vertex in T is folded onto its immediate parent.

If a graph G is not dismantlable, this means that after a sequence of foldings ψ it reduces to a graph H which can not be folded any further. We refer to the graph H as the *residual graph* of G, or just the *residual*, if G can be inferred from the context. It is known that the residual is unique up to isomorphism [6]. We can visualize the folding process for non-dismantlable graphs as a forest of trees T_h hanging from each vertex $h \in H$ (see Figure 2). T_h is composed of vertices that eventually fold onto h and each vertex is folded onto its parent. We define $\psi(u) = w$ if and only if $u \in T_w$, $w \in H$. We note that the tree representation depends on the folding sequence ψ and in general it is not unique.

3 A winning strategy with two hunters

In this section, we present a strategy with two hunters that catches the rabbit on any graph. In general, a single hunter can not always capture the rabbit. This can be seen by considering a cycle of of length at least 4 as the input graph: The rabbit's strategy is to wait until the hunter becomes visible and move to its neighbor which does not contain the hunter. This strategy guarantees that it will never get caught.

The strategy of the two hunters is divided into epochs that are comprised of two phases. An epoch starts with the hunters located at a predetermined vertex. The first phase starts at time $t = 1$.

In *Phase One*, two hunters move together and their goal is to see the rabbit. To achieve this, the hunters generate a random vertex label $v \in \{1 \ldots n\}$ and move together to v. Afterwards, they wait at v until either $(t \bmod n) = 0$ or the rabbit becomes visible. If the rabbit becomes visible at any time, the first phase is over and the second phase starts. Otherwise, the hunters repeat the same process by generating a new label v.

We claim that the first phase lasts only $n^2 \log n$ steps with high probability. To see this, let $r_1, r_2, \ldots$ be the location of the rabbit at times $n, 2n, 3n, \ldots$ Suppose the hunters have not seen the rabbit until time $i \times n$. At that time, the probability that they generate a label in $N(r_{i+1})$ is at least $\frac{1}{n}$. Since they generate a label after every n steps, the first phase will be over in $n^2 \log n$ steps with high probability.

In *Phase Two* the hunters try to catch the rabbit as follows: Suppose the second phase starts at time $t = t_1$ and let $t_i = t_1 + (i - 1)$. At that time both hunters H_1 and H_2 are at vertex h and the rabbit is at vertex r, with $r \in N(h)$. For the rest of the second phase, let r_i denote the position of the rabbit at time $t = t_i$ and let us define $r_0 = h$.

The strategy of H_1 is as follows: At time $t = t_i$, he is located at r_{i-1}. With probability $p_1 = \frac{1}{n^2}$, he attacks the rabbit by generating a random neighbor of r_{i-1} and going there in the next step. With probability $1 - p_1$, he chases the rabbit by going to r_i in the next step. The second phase ends with failure if H_1 attacks and misses the rabbit.

The strategy of H_2 is based on the following observation: If H_1 chases the rabbit for more than n steps, the rabbit must revisit a vertex by the pigeonhole principle. Let u be the first vertex revisited and suppose at time t_r, the rabbit visits a vertex $v \in N(u)$ for the first time before revisiting u. The goal of H_2 is to enter v at the same time with the rabbit. To achieve this, first he guesses u, v and t_r. In order to reach u, he chases H_1 by moving to his location in the previous time step until u. Afterwards, H_2 waits until time $t = t_r - 1$ and goes to v from u. We say H_2 is in *chasing mode* if he is following H_1 and he is in *attacking mode* after he arrives at u. The second phase ends with failure if H_2 misses the rabbit when it arrives at v. To summarize, at time $t = t_1$, the hunters are at r_0 and the rabbit is at r_1. When the hunters are chasing, the locations of the rabbit, H_1 and H_2 at time t_i are r_i, r_{i-1}, r_{i-2} respectively. The phase ends when either hunter attacks. If no hunter attacks within n^2 steps, they end the phase and move to the predetermined vertex to start a new epoch.

Next, we state the crucial property of the strategy of the hunters.

LEMMA 3.1. *During Phase Two, the rabbit can not distinguish between the modes of hunter H_2.*

Proof. If the attacking mode starts at time $t = t_1$, the location of H_2 is the same for both modes. If it starts afterwards, we show that if the rabbit sees H_2, it will get caught with non-zero probability.

Suppose the rabbit sees H_2 at time $t = t_2$ which implies $r_2 \in N(r_0)$. In this case, with probability at least $\frac{p_1}{n}$, H_1 can decide to attack from r_0 to r_2 at time $t = t_1$ and catch the rabbit.

Next, suppose the rabbit sees H_2 at time $t > t_2$. If H_2 was in chasing mode at that time, the fact that rabbit sees H_2 implies $r_i \in N(r_{i-2})$. In this case as well, H_1 could decide to attack in the previous step and catch the rabbit with probability $\frac{p_1}{n}$. Therefore H_2 must be invisible to the rabbit during the chasing mode. But, H_2 will also be invisible in the attacking mode because as soon as the rabbit enters a vertex v where it can see H_2, H_2 can catch it by guessing v and the arrival time correctly.

Therefore in order to avoid getting caught, the rabbit must avoid seeing H_2. But then the information available to the rabbit will be same, no matter which mode H_2 is in: H_2 is out of its sight since the beginning of the second phase.

Lemma 3.2. *During Phase Two, the hunters succeed with non-zero probability.*

Proof. As discussed previously, after the start of the second phase, the rabbit must revisit a vertex u at time $k \leq n$. If the rabbit does not see H_2 until $t = k$, H_2 can catch it with probability $\frac{1}{n^3}$ at least by guessing $t_r, u, v \leq n$. Note that H_1 will still be chasing the rabbit with probability at least $1 - \frac{k}{n^2} \geq 1 - \frac{1}{n}$. On the other hand, if the rabbit sees H_2, it is caught with probability at least $\frac{1}{n^3} = \min\{\frac{p_1}{n}, \frac{1}{n^3}\}$, by Lemma 3.1.

The length of an epoch is $O(n^2 \log n)$: Phase One lasts $O(n^2 \log n)$ time with high probability and Phase Two lasts $\Theta(n^2)$ steps. We have established that in Phase Two, the rabbit is caught with probability at least $\frac{1}{n^3}$. Therefore after $n^3 \log n$ epochs, each of which last $O(n^2 \log n)$ steps at most, the rabbit will be caught, yielding our main result.

Theorem 3.1. *Two hunters can catch a rabbit with limited visibility on any graph with high probability.*

4 Hunter-win graphs

In this section, we start the study of graphs on which a single hunter suffices. An interesting feature of the strategy of two hunters is that one hunter makes the rabbit move constantly, therefore forces it into making mistakes. This suggests that moving when a hunter is not visible may be a disadvantage for the rabbit.

To study this phenomenon we introduce reactive strategies where the rabbit moves only when the hunter is visible and ask the question whether the class of hunter-win graphs against reactive graphs is equivalent to the class of hunter-win graphs against general rabbits. The answer turns out to be negative: The graph in Figure 1 is hunter-win against reactive rabbits. The input graph consists of a cycle and the gadget shown in the figure. It is easy to see that any rabbit can be driven into the gadget by simply chasing it along the cycle. It can also be verified that, once the rabbit is in the gadget, the hunter can reach a vertex whose neighborhood dominates rabbit's neighborhood without being seen. In this case the reactive rabbit would never leave the gadget and get caught. However, a general rabbit can keep moving in the opposite direction of where it saw the hunter last until it leaves the gadget. If the cycle is big enough, the hunter can not reach the other entrance of the gadget before the rabbit and therefore a general rabbit is safe on this graph.

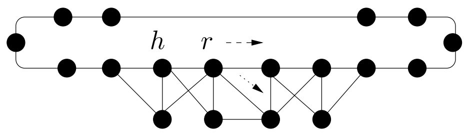

Figure 1: This graph is hunter-win against reactive rabbits but not against general rabbits.

4.1 Characterization of hunter-win graphs against reactive rabbits

In this section, we describe an algorithm that recognizes hunter-win graphs against reactive rabbits. The algorithm marks configurations (h, r) according to the following rules.

Algorithm Mark-Reactive:
Mark all configurations (v, v) for every vertex v. (Initialization)
Repeat
    Mark (h, r) if for all $r' \in N(r)$, there exists a vertex $h' \in N(h)$ with (h', r') marked. *(Stride Rule)*
    For all (h', r) that are marked, for all $h \in N(h') \setminus N(r)$, mark (h, r). *(Stealth Rule)*
Until no further marking is possible.

Next, we prove the *soundness* (if all configurations are marked, then the graph is hunter-win) and *completeness* (if the graph is hunter-win, then all configurations will be marked) properties of the marking algorithm.

Soundness: The proof is by induction on the round k in which a configuration is marked.

When $k = 1$ only the configurations (v, v) are marked and the hunter trivially wins the game in these configurations.

Suppose the configurations marked in the first k rounds are sound and consider the configuration (h, r)

marked during step $k+1$. If (h, r) was marked using the stride rule, during the execution of the game, the hunter can force a configuration marked during the k^{th} step with non-zero probability. Hence these configurations are sound. If, on the other hand, the configuration (h, r) is marked by the stealth rule, we observe that the rabbit will remain at vertex r since the hunter is out of its sight and hence hunter can reach the configuration (h', r) which has been marked during the previous steps. Therefore the stealth rule is also sound by the inductive hypothesis.

Completeness: Clearly, if the rabbit is captured the game ends at a marked configuration. Otherwise, we show that the rabbit can always stay in an unmarked configuration and hence never get caught. Suppose there is an unmarked configuration (h, r) and the hunter and the rabbit are at vertices h and r respectively. There are two cases: If $h \in N(r)$, the rabbit must have a move to a vertex r' such that there exists no $h' \in N(h)$ with (h', r') is marked. Otherwise (h, r) would be marked by the stride rule. On the other hand, if $h \notin N(r)$, no matter which vertex h' the hunter moves, (h', r) is unmarked. Otherwise (h, r) would be marked by the stealth rule.

We can now state the result of this section which follows from the soundness and completeness of the marking algorithm.

Theorem 4.1. *A graph G is hunter-win against reactive rabbits if and only if the algorithm* Mark-Reactive *marks all configurations.*

4.2 Characterization of hunter-win graphs against general rabbits For reactive rabbits, it is easy to see that on a hunter-win graph every rabbit walk can be intercepted (i.e. the rabbit gets caught) by the hunter in $O(n^3)$ steps. However, it is far from being clear that such a polynomial length intercepting walk (i.e a *witness*) exists for non-reactive rabbits. The difficulty is that at any point in time, the rabbit can infer a subset $H \subseteq V$ of possible hunter locations and plan its motion accordingly. This suggests that the state of the game may require specifying arbitrary subsets of vertices, potentially leading to exponential witnesses. Fortunately, we can establish a monotonicity property to establish once again polynomial-size witnesses.

Let $< H, r >$ be the state of the game where H is the set of possible hunter locations when the rabbit is at r. When the rabbit and the hunter are at adjacent vertices r and h respectively, the rabbit knows the hunter's position with certainty and therefore $H = \{h\}$. Now suppose the game starts at configuration (h, r).

Proposition 4.1. *The hunter can reach an adjacent configuration from any starting configuration (h, r).*

The proof of Proposition 4.1 is implicit in the strategy presented in Section 3. During Phase One, the two-hunters act as one and we showed that their strategy ensures that the hunters and the rabbit will end up in adjacent vertices in n steps with non-zero probability. This means that, no matter which path rabbit takes, there exists a hunter-path of length at most n that leads to an adjacent configuration.

Proposition 4.2. *A graph G is hunter-win if and only if the hunter wins starting from any adjacent configuration.*

Proof. If the graph is hunter-win, the hunter must win from all starting configurations including the adjacent ones. Conversely, if the hunter can win from any adjacent configuration, then starting from any configuration he can reach an adjacent configuration by Proposition 4.1, and win the game from here on.

Therefore by Proposition 4.2, on a hunter-win graph, we can assume that the game starts from an initial configuration where the players see each other. In addition, without loss of generality, we assume that the rabbit moves so as to maximize the time taken for capture and the hunter moves so as to minimize it.

We can view any hunter-win game as a sequence of rounds $R_1, \ldots, R_p$ where each round starts with the players located at adjacent vertices. Hence, the rabbit has full knowledge of the hunter's position. Clearly, there are at most n^2 rounds and the rounds do not repeat.

Lemma 4.1. *The length of each round is bounded by n^2.*

Proof. Partition the round into segments of length n each. The rabbit must revisit a vertex r within the same segment. Let $< H_1, r_1 >$ and $< H_2, r_2 >$ be the state of the game during the first and second visits. First, we show that $H_1 \subseteq H_2$. This is because, between r_1 and r_2, the rabbit can not visit any vertex u with $u \in N(h)$, $h \in H_1$: If the hunter is at h, the rabbit would be captured. Next, if $H_1 = H_2$, then the part of the hunter strategy between r_1 and r_2 is redundant and hence the hunter can shorten the game. Therefore as the rabbit keeps visiting the same vertex, its uncertainty is monotonically increasing and after at most n revisits the state of the game becomes $< G - N(r), r >$. In this case, either the rabbit gets caught if it moves or the hunter reveals himself, ending the round. Since the rabbit has to revisit a vertex every n steps and there are at most n revisits, the lemma follows.

Since the length of a round is bounded by n^2 and there are n^2 rounds, we conclude that the total length of a hunter-win game is $O(n^4)$.

Our characterization algorithm for general rabbits is based on the existence of such a polynomial size witness. *We will mark only adjacent configurations:* if the adjacent configurations are all marked, by Proposition 4.2 the hunter wins from all starting configurations. A general rabbit can move even if the hunter is not visible. In order to capture this capability we need to generalize the stealth moves, described next.

4.2.1 Stealth Moves A *k-stealth move* from configuration (h, r) with $h \in N(r)$ to a marked configuration (h', r') is defined as follows: For every rabbit path $P_r = \{r, r_1, \ldots, r_k = r'\}$ of length k, the hunter has a path $P_h = \{h, h_1, \ldots, h_k = h'\}$ such that $h_i \notin N(r_i)$ for $i = 1, \ldots, k-1$, $h_k \in N(r_k)$ and (h_k, r_k) is marked. We refer to P_h as the *stealth path* corresponding to P_r. A configuration (h, r) is marked by the *Stealth Rule* if for all $r' \in N^k(r)$, there exists a k-stealth move to a marked configuration (h', r'). Note that the Stealth Rule for $k = 1$ subsumes the Stride Rule.

LEMMA 4.2. *The markings corresponding to stealth moves are sound.*

Proof. Suppose all previously marked adjacent configurations are sound and consider the next adjacent configuration (h, r) marked by a stealth move of length k. At time $t = 0$ the rabbit is located at r. Since we mark only the adjacent configurations, the state of the game is $< \{h\}, r >$. Take any rabbit path of length k, and suppose at time $t = i$ the rabbit is at vertex r_i. Let $r'_1, \ldots, r'_p$ be the vertices accessible from r_i in the remaining $k - i$ steps and $P_1, \ldots P_p$ be the corresponding stealth paths such that at the end of k steps, P_i ends at vertex h'_i and (h'_i, r'_i) is marked. Let E_i be the event that the hunter has chosen path P_i, $i = 1, \ldots, p$ and let h_i be the i^{th} vertex on P_i. The claim follows from the observation that no matter which path P_i the hunter chooses, the information available to the rabbit is the same, namely hunter was not visible for the last i steps. Therefore the state of the game is $< H, r >$ where $\{h_i | 1 \leq i \leq p\} \subseteq H$. Since the rabbit can not distinguish between the events E_i, no matter which final destination r'_j it chooses, the hunter can be at the corresponding vertex h_j and arrive at the already marked configuration (h'_j, r'_j).

The stealth moves starting from configuration (h, r) and ending at configuration (h', r') can be computed efficiently by dynamic programming.

We will need an intermediate look-up table T, with $T[h, r, h', r', k]$ = TRUE if and only if for any rabbit path $\{r, r_1, \ldots, r_k = r'\}$ of length k there is a stealth path of length k that starts from h and ends at h'.

The entries of the Table T are filled as follows:

(i) $T[h, r, h', r', 0]$ = TRUE iff $h = h'$, $r = r'$ and $h' \in N(r')$.

(ii) $T[h, r, h', r', 1]$ = TRUE iff $h' \in N(h)$, $r' \in N(r)$ and $h' \in N(r')$.

(iii) $T[h, r, h', r', k+1]$ = TRUE iff for all $u \in N(r)$ there is a vertex $v \in N(h) \setminus N(u)$ with $T[u, v, h', r', k]$ = TRUE, for $1 \leq k \leq n^2$.

We now present a marking algorithm that uses the look-up table T to compute the stealth moves.

```
Algorithm Mark-General:
Mark all configurations $(v, v)$ for every vertex $v$. (Initialization)
Repeat
    For all configurations $(h, r)$ with $h \in N(r)$, mark $(h, r)$ if there exists an index $k \leq n^2$ such that $\forall r' \in N^k(r)$, there exists a vertex $h'$ with $T[h, r, h', r', k]$ = TRUE and $(h', r')$ is marked. (Stealth Rule).
Until no further marking is possible.
```

LEMMA 4.3. *If the graph is hunter-win, then the marking algorithm Mark-General will mark all adjacent configurations.*

Proof. Let (h, r) be an adjacent configuration left unmarked after the execution of algorithm Mark-General. We claim that the rabbit can get to an adjacent configuration (h', r') that is unmarked. Suppose not. This means that for any rabbit path $r, r_1, r_2, \ldots, r_k$ there is a hunter path $h, h_1, h_2, \ldots, h_k$ with $h_k \in N(r_k)$ and (h_k, r_k) is marked. By Lemma 4.1, we have $k \leq n^2$. This implies that (h, r) would be marked by the stealth rule, which gives us the desired contradiction.

Therefore, starting from any unmarked adjacent configuration (h, r), the rabbit can reach another unmarked adjacent configuration. This means that the rabbit will never get caught, since a capture implies that the game enters the configuration (v, v) for some vertex v which is a marked adjacent configuration.

THEOREM 4.2. *A graph G is hunter-win against general rabbits if and only if the algorithm* Mark-General *marks all adjacent configurations.*

Proof. If all the configurations are marked, G is hunter-win due to the fact that the stealth rule is sound (Lemma 4.2). Conversely, if there is an unmarked configuration, the rabbit is never caught by Lemma 4.3.

5 Complete visibility and dismantlable graphs

When the rabbit has full visibility, the stealth rule does not make sense. In fact, we will show that the stride rule against reactive rabbits is sound and complete against rabbits with full visibility.

```
Algorithm Mark-FullVisibility:
Mark all configurations $(v, v)$ for every vertex $v$.
Repeat
    Mark $(h, r)$ if for all $r' \in N(r)$, there exists a vertex $h' \in N(h)$ with $(h', r')$ marked. (Stride Rule)
Until no further marking is possible.
```

It turns out that the algorithm Mark-FullVisibility recognizes hunter-win graphs against rabbits with full visibility.

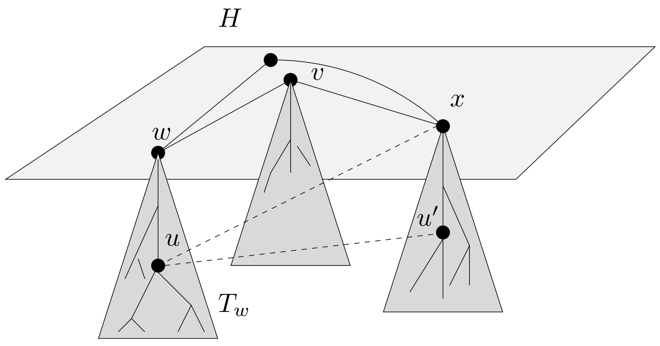

Figure 2: Visualization of the folding procedure for a non-dismantlable graph. The vertices w,v and x are in the residual H. Since there is no edge from w to x, the edges shown with dashed lines can not exist.

We will need the following property of non-dismantlable graphs:

Proposition 5.1. *Let G be a non-dismantlable graph, ψ be a folding sequence and H be the residual. Let x and w be two distinct vertices in H and T_x and T_w be the corresponding folding trees (see Figure 2). If there exist a vertex $u \in T_w$ that is adjacent to a vertex $u' \in T_x$, then $x \in N(w)$.*

Proof. Without loss of generality, suppose u was folded before u'. This implies that the parent of u must be adjacent to u'. We replace u with its parent and continue this process of propagating the edge between u and u', which must eventually reach the roots w and x of the corresponding trees.

Theorem 5.1. *The algorithm Mark-FullVisibility marks all configurations if and only if the input graph is dismantlable.*

Proof. Suppose the input graph G is dismantlable. We can prove that all configurations will be marked by induction on the order of G. Since G is dismantlable, it must have two vertices u and v with $N(u) \subseteq N(v)$. Let $G' = G - \{u\}$ and run algorithm Mark-FullVisibility on G'. Suppose, inductively, that all configurations in G' are marked. Consider the marking algorithm for G which marks (u, u) first and simulates the marking algorithm on G' afterwards. In addition, whenever (x, v) is marked for a vertex $x \in G'$, we also mark (x, u). This is possible since (x, v) is marked implies that for all $v' \in N(v)$, there exists a vertex $x' \in N(x)$ with (x', v') marked and $N(u) \subseteq N(v)$. Next, we show that all the configurations (x, y) in G' will also get marked in G. Suppose there exists a configuration (x, y) that is marked in G' but not in G. Consider the first such configuration that is discovered in the marking of G. It must be that $u \in N(y)$ and that for all $x' \in x$, (x', u) is not marked at this point. Also, $v \in N(y)$ since $N(u) \subseteq N(v)$. Now using the fact that (x, y) gets marked at this stage in G', we know that there exists $x'' \in N(x)$ such that (x'', v) is already marked. But then (x'', u) must also be marked at this point according to the modified marking rule. A contradiction! Thus, any (x, y) marked in G' will also be marked in G. It follows that for any x such that (x, v) is marked in G', we can mark (x, u) in G. It is easy to see that for any x, the configuration (u, x) will also be marked in G since u is adjacent to v and, by the argument above, for all $x' \in N(x)$, (v, x') is marked.

Now suppose the input graph is not dismantlable. Let ψ be a sequence of folds reducing G to a residual graph H. For any two vertices $u \in G$ and $v \in H$, we claim that (u, v) is unmarked if $\psi(u) \neq v$. Suppose this is not true and let (u, v) be the first marked configuration such that $\psi(u) \neq v$ (Figure 2). Let $w = \psi(u), w \neq v$. Note that v must have a neighbor x such that $x \notin N(w)$, otherwise v would fold onto w. When (u, v) gets marked, there must be a vertex $u' \in N(u)$ such that (u', x) is marked. If $\psi(u') = x$, this would imply $x \in N(w)$ by Proposition 5.1. So it must be the case that $\psi(u') \neq x$. But then, the fact that (u', x) is marked contradicts with the fact that (u, v) is the first configuration marked with $\psi(u) \neq v$. Therefore, we conclude that if the graph is not dismantlable, the marking process will not mark all configurations.

As stated earlier, it has been shown that the class of graphs that are hunter-win against rabbits with full visibility are precisely the class of dismantlable graphs [6]. Therefore we obtain:

Corollary 5.1. *A graph G is hunter-win against rabbits with full visibility if and only if the algorithm Mark-*

FullVisibility marks all configurations.

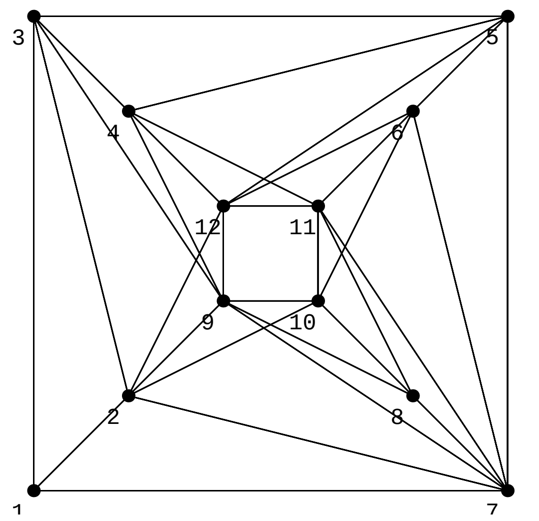

Figure 3: This graph is hunter-win against rabbits with limited visibility. However, a rabbit with full visibility never gets caught.

We know that there are non-dismantlable graphs that are hunter-win against rabbits with limited visibility. An example is shown in Figure 3. The labels on the vertices indicate their folding order: First, vertex 1 folds onto vertex 2, afterwards vertex 2 folds onto vertex 9, etc. After folding vertices 1 to 8, vertices 9 to 12 can not be folded, leaving a four-cycle as the residual. Therefore this graph is not dismantlable and consequently it is not hunter-win against rabbits with full visibility. To see that the hunter wins against rabbits with limited visibility, let us define the mapping $p : V \to V$ where V is the set of vertices. For $v \in V$ with $1 \leq v \leq 8$, $p(v)$ is the vertex which v folds onto. We define $p(9) = 2$, $p(10) = 8$, $p(11) = 6$ and $p(12) = 4$. The first observation is that the hunter wins the game if he can force the rabbit to go to vertex 1 while he is at vertex 2. Next, we observe that if the rabbit is at vertex $v \neq 1$ and the hunter is at $p(v)$, the rabbit must move to a lower numbered vertex. Now suppose the rabbit is reactive. In this case, it can be verified that for any rabbit location r and for any hunter location $h \notin N(r)$, the hunter has a path to $p(r)$ that does not enter $N(r)$. Therefore, by visiting $p(r)$ repeatedly the hunter can force a reactive rabbit to eventually move to vertex 1 and win the game afterwards.

Hence, the rabbit must have a non-reactive strategy, meaning that it must move when the hunter is not visible. Consider the first time this happens: Suppose the hunter and the rabbit are at vertices h and r with $h \in N(r)$ and the rabbit takes the path $r \to r' \to r''$ such that the hunter is not visible from r'. It can be shown, by enumeration, that for any such vertices h, r, r' and r'', the hunter has a path $h \to h' \to r''$ that captures the rabbit. Therefore the rabbit can not have a non-reactive strategy either and the graph is hunter-win against both types of rabbits.

We conclude this section with an interpretation of Theorem 5.1: If G is a graph that is hunter-win against rabbits with limited visibility but not against rabbits with full-visibility, the hunter captures the rabbit with limited visibility using the stealth moves.

5.1 Hunter strategy for dismantlable graphs

Given a folding tree T rooted at vertex v, consider the vertex r rabbit is located. We say the hunter is an *ancestor* of the rabbit if he is located on the path from r to v. Suppose the vertices of T are ordered by their deletion times. The hunter strategy is based on the following two lemmas.

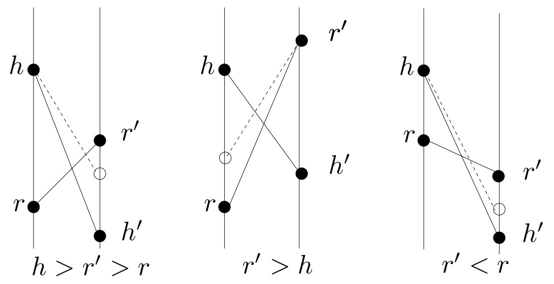

Figure 4: Hunter can always stay above the rabbit. The height of a vertex is proportional to its label.

LEMMA 5.1. *Hunter can always maintain ancestry.*

Proof. Suppose the hunter is at vertex h and is an ancestor of the rabbit who is located at vertex r. Let r' be the rabbit's location in the next round. If h is a common ancestor of r and r' on the folding tree T, then the lemma is trivially true. Otherwise, since h is an ancestor of r and (r, r') is an edge, using basic properties of foldings it can be shown that h is adjacent to a vertex on the path that connects r' to the root of T. We show that there is always such a vertex h' with $h' \geq r'$ by a case analysis on r' (See Figure 4). Suppose for contradiction $h' < r'$. We will show that h must be adjacent to r' thus allowing the hunter to catch the rabbit in one step.

Case ($h > r' > r$): In this case all the ancestors of h' deleted before h (including r') must have edges to h.

Case ($r' > h$): All the ancestors of r deleted before r' (including h) must have an edge to r'.

Case ($r' < r$): All the ancestors of h' deleted before r (including r') must have an edge to h.

In fact, not only the hunter can maintain ancestry, but also he can reduce his height in the tree gradually and therefore get closer and closer to the rabbit.

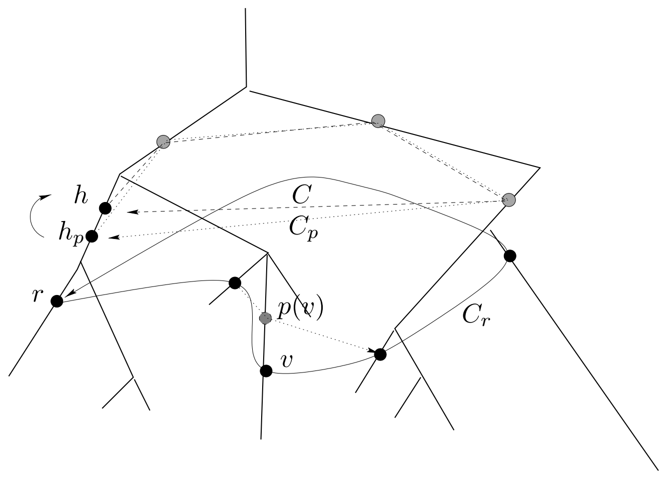

Figure 5: Hunter can make progress every time the rabbit revisits a vertex.

Lemma 5.2. *Every time the rabbit revisits a vertex, the hunter can reduce its height in the tree while maintaining ancestry .*

Proof. Fix any rabbit cycle C_r and let v be the vertex with the lowest label on this cycle and $p(v)$ be its parent (see Figure 5). Since v was deleted first, $p(v)$ must have edges to the neighbors of v on the cycle, so we can make a new cycle by replacing v with $p(v)$. We continue this process until the cycle reaches h, the location of the hunter (this must happen since hunter is an ancestor at all times). Let us call this cycle C. Let C_p be the cycle just before C which contains h's child h_p, instead of h. Consider the path $P = \{h\} \cup (C \cap C_p) \cup \{h_p\}$. If the rabbit follows the cycle C_r, hunter can follow the path P and end up at h_p which is lower than h.

We are now ready to present the hunter strategy on a dismantlable graph G. First, the hunter builds the folding tree T for any folding sequence ψ. Afterwards, he simply guesses the vertex the rabbit will jump to and jumps to the lowest possible ancestor of this vertex (see Figure 5). By Lemma 5.1 he can always remain an ancestor of the rabbit. Further, he can reduce his height in T every time the rabbit revisits a vertex (Lemma 5.2). Since the tree has a finite height, he can eventually catch the rabbit.

5.2 Extension to non-dismantlable graphs For non-dismantlable graphs, we can extend the notion of ancestry as follows. Suppose the rabbit is at r and the hunter is at h. We say hunter is an ancestor of the rabbit if there is a folding of the vertices such that in the corresponding forest representation, h is located on the path from r to the root of the tree that contains r. Once the hunter establishes ancestry, it is easy to see that Lemma 5.1 and Lemma 5.2 still hold –both for reactive and general rabbits. Therefore the hunter can win the game afterwards. Note that the hunter can trivially establish ancestry on dismantlable graphs.

In addition, if we define each vertex as its trivial parent, it is clear that the rabbit wins the game if the hunter can never become an ancestor. Therefore the class of hunter-win graphs is precisely the class of graphs on which the hunter can become an ancestor. One can view the stealth moves as giving the hunter the power to become an ancestor on non-dismantlable but hunter-win graphs such as the one in Figure 3.

6 Concluding Remarks

Let us define rabbits with *i-visibility* as the rabbits who can see all vertices within distance i. It is known that one hunter always suffices to catch rabbits with 0-visibility [1]. In this paper, we studied rabbits with 1-visibility and established that 2 hunters always suffice to catch such rabbits. A natural question is how many hunters suffice when the rabbit has i-visibility. Surprisingly, the number of hunters required for 2-visibility is unbounded: using the probabilistic method, one can show that there exist random bipartite graphs where $\tilde{\Omega}(\sqrt{n})$ hunters are needed.

Another aspect of the game is the time required to catch the rabbit. For 0-visibility, one hunter succeeds in time $O(n \log n)$ [1]. For 1-visibility we showed that two hunters succeed in $\tilde{O}(n^5)$ time. However, it is not clear whether a single hunter can catch a rabbit on a hunter-win graph in polynomial time. We leave this as a direction for future work.

References

[1] M. Adler, H. Räcke, N. Sivadasan, C. Sohler, and B. Vöcking. Randomized pursuit-evasion in graphs. *Proceedings of the International Colloquium on Automata, Languages and Programming (ICALP)*, 2002.

[2] M. Aigner and M. Fromme. A game of cops and robbers. *Discrete Applied Math*, 8:1–12, 1984.

[3] T. Basar and P. R. Kumar. On worst case design strategies. *Computers and Mathematics with Applications: Special Issue on Pursuit-Evasion Differential Games*, 13(1-3):239–245, 1987.

[4] T. Basar and G. J. Olsder. *Dynamic Noncooperative Game Theory.* SIAM, 1998.

[5] P. Bernhard, A.-L. Colomb, and G. P. Papavassilopoulos. Rabbit and hunter game: two discrete stochastic

formulations. *Comput. Math. Appl.*, 13(1-3):205–225, 1987.

[6] G. Brightwell and P. Winkler. Gibbs measures and dismantlable graphs. *J. Comb. Theory (Series B)*, 78, 2000.

[7] 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.

[8] S. Fitzpatrick and R. Nowakowski. Copnumber of graphs with strong isometric dimension two. *ARS COMBINATORIA*, 59:65–73, 2001.

[9] L. J. Guibas, J.-C. Latombe, S. M. LaValle, D. Lin, and R. Motwani. A visibility-based pursuit-evasion problem. *International Journal of Computational Geometry and Applications*, 9(4/5):471–, 1999.

[10] J. P. Hespanha, G. J. Pappas, and M. Prandini. Greedy control for hybrid pursuit-evasion games. In *Proceedings of the European Control Conference*, pages 2621–2626, Porto, Portugal, September 2001.

[11] I.Chatzigiannakis, S.Nikoletseas, and P.Spirakis. An efficient communication strategy for ad-hoc mobile networks. In *Proc. of 15th Symposium on Distributed Computing (DISC'2001)*, pages 285–299, 2001.

[12] A. S. LaPaugh. Recontamination does not help to search a graph. *J. ACM*, 40:224–245, 1993.

[13] N. Megiddo, S. L. Hakimi, M. R. Garey, D. S. Johnson, and C. H. Papadimitriou. The complexity of searching a graph. *J. ACM*, 1988.

[14] S. Neufeld and R. Nowakowski. A game of cops and robbers played on products of graphs. *DISCRETE MATHEMATICS*, 186:253–268, 1998.

[15] R. Nowakawski and P. Winkler. Vertex-to-vertex pursuit in a graph. *Discrete Math*, 43:235–239, 1983.

[16] S.-M. Park, J.-H. Lee, and K.-Y. Chwa. Visibility-based pursuit-evasion in a polygonal region by a searcher. *Proceedings of the International Colloquium on Automata, Languages and Programming (ICALP)*, 2076, 2001.

[17] T. D. Parsons. Pursuit evasion in a graph. In Y. Alavi and D. R. Lick, editors, *Theory and Application of Graphs*, pages 426–441. Springer Verlag, 1976.

[18] R. Vidal, O. Shakernia, J. Kim, D. Shim, and S. Sastry. Probabilistic pursuit-evasion games: Theory, implementation and experimental evaluation. *IEEE Transactions on Robotics and Automation*, 18:662–669, 2002.