Capturing Oracle Guided Hiders

Akshat Tandon
IIIT-Hyderabad, India
akshat.tandon@research.iiit.ac.in

Kamalakar Karlapalem
IIIT-Hyderabad, India
kamal@iiit.ac.in

ABSTRACT

Consider a closed environment with static obstacles and mobile agents moving around. There are hider agents that hide from the seeker agents. The seeker has a limited visibility range, and if a hider comes into the visibility region of a seeker, it is considered caught. The practical applications range from gaming to security. In this work, we focus on deterministic capture of hiders, even if they are guided by an Oracle which knows the future positions of seekers. We develop strategies for seekers, having limited visibility ranges, to catch all hiders and establish minimum bounds on the number of seekers required to catch the hiders, on a per strategy basis. We use spatio-temporal graph models and reasoning to formulate and address the problem.

KEYWORDS

Emergent behavior; Pursuit evasion games; Hide-and-Seek

ACM Reference Format:
Akshat Tandon and Kamalakar Karlapalem. 2020. Capturing Oracle Guided Hiders. In *Proc. of the 19th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2020), B. An, N. Yorke-Smith, A. El Fallah Seghrouchni, G. Sukthankar (eds.), Auckland, New Zealand, May 2020,* IFAAMAS, 9 pages.

1 INTRODUCTION

The game of Hide-and-Seek involves two competing agent teams. The team of seeker agents is concerned with finding all the hider agents, and the team of hider agents is concerned with remaining concealed and hidden from the seeker agents. A pragmatic example of the problem is a warehouse environment with no drones or cameras, and some culprit agents are hiding from the police agents. The hiders may be guided by an external entity (oracle) about the future positions of the seeker, allowing them to evade easily.

In this work, we introduce hiker graphs, which allows encapsulation of spatial features of a landscape environment (city's grid-like arrangement of streets, buildings). Furthermore, we present two seeker strategies - *trap* and *wave*, both of which utilize the hiker graph to deterministically capture all the hiders, even if the hiders are informed by oracles about the future positions of seekers. The hiker graph explicitly represents visibility and connectivity aspects via separate nodes and edges. It permits abstraction of spatial features at a desired level of granularity, in turn allowing strategies to incorporate visibility and connectivity as primitives in their optimization operations. The agent model used by us assumes a limited visibility range, which is in contrast to existing work (Gerkey et al. [9]).

Proc. of the 19th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2020), B. An, N. Yorke-Smith, A. El Fallah Seghrouchni, G. Sukthankar (eds.), , May 2020, Auckland, New Zealand.

1.1 Related Work

Hide-and-Seek has been studied in various forms and under different domains. In operations research, it is studied under search theory. Hide-and-Seek game involving movable hiders was introduced by Isaacs [10] in the form of Princess and Monster game. This was solved for circular boundaries by Zelikin [20], Alpern [1], and Foreman [7] and was eventually solved for the general case by Gal [8]. Search strategies in the case of static hider have also been well explored by Alpern and Gal [2]. The strategies have regular and well-defined environments and use proximity and not visibility as a capture criterion. This is in contrast to our strategies which involve irregular and obstacle-filled environments (city grids) and use visibility as a capture criterion. The search theory strategies are also mostly probabilistic aiming to minimize expected capture time. Our strategies guarantee capture and have deterministic (fixed) upper bound on capture times.

Similar to search theory is the domain of pursuit-evasion games within mobile robotics and multi-agent systems (Chung et al. [5]). Parsons [13], and Megiddo and Hakimi [11] considered the problem of figuring out the minimum number of seekers required to guarantee the capture of a hider hiding in a discrete graph environment. This problem was eventually found out to be NP-Hard by Megiddo et al. [12]. Suzuki and Yamashita [17] extended the above problem to a continuous polygons based environment having visibility based agents embedded with K flashlights. Hider is considered caught if it intercepts one of the flashlights. Gerkey et al. [9] also presented visibility based, guaranteed capture strategies. The agents considered by them have a limited field of view (FOV) instead of having K flashlights. Our strategy is similar to the above pursuit-evasion strategy in the sense of having visibility based agents exhibiting limited FOV, and a guaranteed capture criteria. Visibility range is defined as the maximum distance at which an object can be discerned. The agents used by Gerky et al have an unlimited visibility range whereas our agents have a limited visibility range. Using an unlimited visibility range, a seeker can capture a much larger coverage area, as opposed to a limited range. This impacts how spatial abstractions, and the eventual strategies, are formulated. Tandon and Karlapalem [18] described strategic and coverage points as spatial abstractions for encapsulating a 2-D environment. In our work, we build upon these to arrive at hiker graphs.

Another related field is that of multi-agent security. These involve modeling a real-world domain as a Stackelberg game (Pita et al. [16], Fang et al. [6]) and finding an optimal strategy for the seekers to commit (Paruchuri et al. [15], Paruchuri et al. [14]). The optimal strategies in these cases are optimal in an expected payoff sense. Also, unlike our strategies, they do not guarantee the capture of hiders (or followers).

Hide-and-Seek has also been explored in the domain of multi-agent deep reinforcement learning (Baker et al. [3]). The environment used in their setting had movable obstacles in the form of

blocks and ramps. Using repeated self plays, the hider and seeker agents were able to learn various strategies, such as hiders using blocks to prevent a seeker from entering a region, and seekers using ramps to jump over obstacles to enter the blocked regions. Our work only involves immovable obstacles. Also, our agents follow a fixed strategy and do not learn via repeated trails (Baker et al. [3]).

1.2 Players, Environment, and Obstacles

The game is played in a 2D bounded, continuous, rectangular environment $\mathcal{E}$ (figure 1). The environment contains many obstacles $O = \{o_1, o_2, ..., o_k\}$. For simulation purposes, we consider square and rectangles as basic obstacle blocks and construct any arbitrary polygonal shape as a combination of them. The game comprises of a team of hider agents $\mathcal{H} = \{h_1, h_2, ..., h_n\}$ and a team of seeker agents $\mathcal{S} = \{s_1, s_2, ..., s_m\}$. The game simulation occurs in discrete time steps. At each step, the agents chose an action from one of the sixteen compass directions $\mathcal{A} = \{d_1, d_2, ...d_{16}\}$, each oriented at an angle of multiples of 22.5° from its base axis.

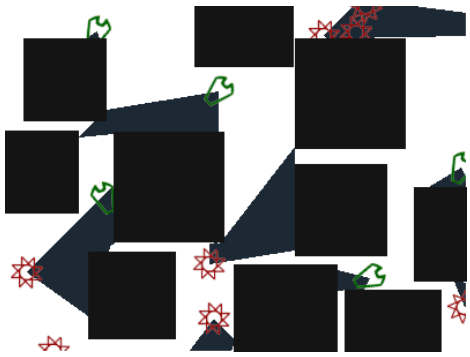

Figure 1: Green agents are seekers and red ones are hiders. Visibility cones emanate from each agent.

The state of an agent is defined by its current position $(x, y) \in \mathcal{E}$ as well as the compass direction $d \in \mathcal{A}$ which the agent is facing. The agent changes the state by taking an action. The speed of all the agents is assumed to be a fixed constant. Thus, at each game step, the agents can only move rectilinearly by a fixed distance in direction d.

1.3 Visibility and Elimination

To approximate the notion of visibility in simulation, we associate a visibility cone with each agent. A hider is visible to a seeker if the hider (associated point (x, y)) lies inside the seeker's visibility cone. The visibility cone depends on the current state of an agent, changes as the agent moves and is constructed by tracing the path of uniformly spaced rays emitted from the agent's current position, spread along some angle to the left and right of the agent's head facing direction. Each agents visibility cone has a limited field of view (viewing angle) and a limited range (maximum distance at which objects are visible).

1.4 Spatial Abstractions

We use graph-based abstractions to encapsulate the spatial features of the 2D landscape environment, such as obstruction, visibility, coverage and, connectivity. Our strategies use these abstractions.

1.4.1 Strategic and Coverage Points. A strategic point is used as an abstraction for a hiding location. It is the midpoint of an edge of an obstacle. Each obstacle yields a set of strategic points and the union of these sets constitute the strategic points set $\mathcal{SP}$, of the environment.

Coverage point is used as an abstraction for a seeking location. It is a point from which one or more strategic points are visible when scanned in all the directions. A coverage point is said to cover the strategic points visible from it. An optimal set of coverage points $\mathcal{CP}$ must satisfy the following criteria

- All the strategic points of the environment must be visible to at least one coverage point in the set.
- Maximal number of strategic points (if possible) must be visible from each coverage point in the set.

We derived an algorithm that finds such a set of optimal coverage points by utilizing an intermediary visibility graph $\mathcal{VG}$, built upon strategic points. The nodes of this graph consist of strategic points of the environment $\mathcal{E}$. There is an edge between any two nodes if there exists a point in $\mathcal{E}$ from which the strategic points corresponding to nodes are visible (figure 2). To reduce the computational cost of constructing $\mathcal{VG}$, a discretized grid cell version $\mathcal{G}$ of environment $\mathcal{E}$ is considered. Discretization of the environment is done by partitioning the continuous 2-D space into contiguous cells via uniformly spaced horizontal and vertical lines. Each cell $c \in \mathcal{G}$ is indexed by its row and column values $[i, j]$ and is represented by its center point coordinate $(c_x, c_y) \in \mathcal{E}$. To compute edges of $\mathcal{VG}$, associate each strategic point with the set of grid cells visible to it if scanned in all the directions. If the intersection of the visible cell set associated with two strategic points is not null, then there exists a point in $\mathcal{E}$ which is visible to both of these. Thus, there exists an edge between those two strategic points in $\mathcal{VG}$. An optimal set of coverage points can be obtained by

(1) Enumerating over all the maximal cliques (Bron and Kerbosch [4]) of the visibility graph $\mathcal{VG}$.
(2) Finding the smallest set of coverage points required for covering the strategic points corresponding to the nodes of enumerated clique.
(3) Taking the union of these coverage point sets.

Maximal cliques of the visibility graph are considered because a maximal clique of $\mathcal{VG}$ encapsulates the set of strategic points that are visible to each other. If strategic points belonging to a set are visible to each other, fewer number of coverage points are required to cover them. It should also be noted that a single coverage point is not always sufficient for covering all the strategic points of a clique.

Lemma 1.1. *A single coverage point is not always sufficient for covering all the strategic points of a visibility graph clique*

Proof. We prove this via a counterexample. Consider an environment in which there are only three strategic points. An obstacle can be present inside the circular perimeter formed by the three

(a) Visibility graph. There must be common point from which 1 and 2 are visible. Similar is the case with (1,3) and (2,3).

(b) There might exist a common coverage point from which all three strategic points are visible.

Figure 2: Visibility Graph and Coverage Points

points in a manner that prevents the three points from being seen by a common point. However, any two of them can be seen by a common point. The resulting visibility graph $\mathcal{VG}$ is a complete graph, comprising of three nodes, each having an edge with the other two (figure 2(a)). Since all three cannot be seen by a common point, there must be at least two coverage points to cover all the nodes of this clique. □

To find the smallest set of coverage points corresponding to a clique $\mathcal{CQ}$, iterate over all grid cells of $\mathcal{G}$, visible from some node (i.e. strategic point) of that clique, and find the cell whose center covers the maximum number of nodes (i.e. strategic point) of that clique. If the found cell's center covers all the strategic points of the clique, return the center as the sole member of the coverage points set, else remove the strategic points covered by the found cell from the clique $\mathcal{CQ}$ and feed the modified clique back to the algorithm to recursively find remaining coverage points.

This coverage point set $\mathcal{CP}$ satisfies our two optimal criteria.

Lemma 1.2. *Each strategic point of the environment is visible to at-least one coverage point in the set $\mathcal{CP}$*

Proof. Assume there exists a strategic point $(s_x, s_y) \in \mathcal{E}$ which is not visible to any coverage point. Since each coverage point is associated with a maximal clique of $\mathcal{VG}$, this implies that (s_x, s_y) does not belong to any maximal clique. Since any node in $\mathcal{VG}$ must belong to one of the maximal cliques, it implies that (s_x, s_y) does not belong to $\mathcal{VG}$ i.e. there is no node in $\mathcal{VG}$ corresponding to (s_x, s_y). This is a contradiction since for each strategic point, there exists a node in $\mathcal{VG}$ (by definition). Thus, our assumption is wrong, and there does not exist any strategic point which is not visible to any coverage point. □

A maximal clique of strategic points can be covered by either a single coverage point or a set of coverage points (refer lemma 1.1). In case of coverage by a single coverage point, each such point generated by the above algorithm is associated with as many strategic points as is possible (maximal). In other words, such a coverage point can't cover more strategic points, than the ones which the algorithm for finding the coverage points has currently associated it with.

Lemma 1.3. *Maximal number of strategic points are visible from each coverage point which covers a maximal clique $\mathcal{CQ} \subseteq \mathcal{VG}$*

Proof. Assume a coverage point (c_x, c_y) is generated by the algorithm corresponding to a maximal clique $q \subseteq \mathcal{VG}$. (c_x, c_y) covers all nodes of q. In addition to these nodes, (c_x, c_y) also covers a strategic point $(s_x, s_y) \notin q$. Since q is a maximal clique, there exists some $(s_{\bar{x}}, s_{\bar{y}}) \in q$ which does not have an edge with (s_x, s_y) in visibility graph $\mathcal{VG}$. Two strategic points not having an edge in $\mathcal{VG}$ implies that there does not exist a common point from which both of the points are visible. This contradicts the assumption that a coverage point covering a maximal clique covers an additional strategic point (s_x, s_y). □

1.4.2 Hiker Graph. A hiker (hider-seeker) graph $\mathcal{HG}$ is used as an abstraction of the 2D landscape environment. It consists of two types of nodes - strategic nodes and coverage nodes, and two types of edges - traversal edges and visibility edges. Corresponding to each strategic point in $\mathcal{SP}$, there exists a strategic node in $\mathcal{HG}$, and corresponding to each coverage point in $\mathcal{CP}$ there exists a coverage node.

Traversal edges connect either two strategic nodes or two coverage nodes. These edges represent the notion that a hider agent can traverse from one strategic point to another, and a seeker agent can traverse from one coverage point to another. Two strategic nodes are connected via a traversal edge if their corresponding strategic points are near to each other, by some metric. In our empirical analysis, we used a proximity-based metric. Any two coverage nodes are always connected by a traversal edge. Traversal connectivity of strategic nodes encapsulate the fact that the hider agents do not move freely in the environment. They always move from one obstacle to another one, or from one side of the obstacle to another side. If they moved freely, they would expose themselves for a significant duration leading to capture by seekers. On the other hand, traversal connectivity of coverage nodes encapsulate the fact that the seekers are free to move anywhere. Two strategic nodes, or two coverage nodes, are said to be traversal adjacent if they are connected via a traversal edge.

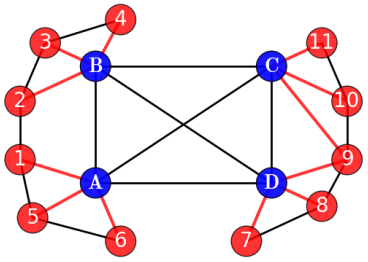

Figure 3: Hiker Graph

Visibility edges connect a strategic node with a coverage node. These represent the notion that a strategic point is visible from a coverage point. A strategic node is connected to a coverage node via a visibility edge if the associated strategic point is visible to the associated coverage point, in the 2D environment $\mathcal{E}$. A coverage and strategic node connected via a visibility edge are said to be visibility adjacent.

Consider figure 3. The red nodes are the strategic nodes and the blue ones are the coverage nodes. The black edges are the traversal edges and the red ones are the visibility edges.

A Hide-and-Seek game can be represented in a hiker graph. Before the start of the game, the hiders occupy some strategic nodes. At the start, seekers occupy some strategic nodes. The game proceeds in turns and at each turn, the agents may choose to move to their traversal adjacent nodes. The *hider agents move to traversal adjacent strategic nodes* and *seeker agents move to traversal adjacent coverage nodes. If a hider occupies a strategic node that is visible adjacent to coverage node occupied by a seeker, then the hider is considered caught and is eliminated from the game.* The game continues until all the hiders are caught.

Note, a *step* in simulation of the hide-and-seek game in 2D environment $\mathcal{E}$ (section 1.2), is different from a *turn* in the abstracted out version of the game played on the hiker graph $\mathcal{HG}$. At a discrete-time step in the simulation, each agent moves by the same fixed Euclidean distance in $\mathcal{E}$, whereas at a turn in the hiker graph, the agents move to adjacent nodes. Performing a 2D traversal in $\mathcal{E}$, across points that correspond to traversal adjacent nodes, requires multiple steps by an agent.

1.4.3 Gaps, Oracles, and Oscillatory Behavior. The hiders which remain, at some turn t of the game, are those that occupy the strategic nodes which are not visible adjacent to any currently occupied coverage node. We classify such strategic nodes as gap nodes. At each turn, depending on the occupancy of coverage nodes, the gap nodes change. The hiders which have continued to evade the seeker till turn t have continued to be on gap nodes at each turn. A hider can continue to evade seekers as long as it manages to occupy gap nodes. *The hiders do not know before a turn, which of the nodes are going to be gap nodes since these are decided by the actions of seekers. However, if they are guided by an oracle agent, who can predict the future movements of seekers, they may continue evading the seekers.* For example, suppose the game is played between a single seeker and a single hider. At the start of the game (refer figure 3), assume that a hider is at node 1 and a seeker is at node B. The hider is guided by an oracle and hence knows the future positions of seeker. When the seeker traverses to node A, the hider traverses to node 2 and when the seeker goes back to node B, the hider also goes back to node 1. *The oscillatory behavior, induced by the presence of an oracle agent prevents the hider from ever getting caught. In the presence of an oracle, a single seeker can never capture such a hider. However, if there are two seekers and one hider, then even if the hider is guided by the oracles, it will be caught.* Assume the hider is at node 1. If both the seekers simultaneously occupy nodes A and B, then irrespective of any oracle, the hider will be caught. *The focus of our paper is on strategies, which capture oracle guided hiders, using minimum possible number of seekers.* Another way of looking at strategies, which aid in capturing oracle guided hiders, is regarding them as deterministic capture strategies. The probability of capturing hiders by such a strategy is always one.

2 STRATEGIES

We start by describing trivial strategies and building upon them to arrive at our final one. All of the strategies which we describe guarantee the capture of all the hiders present in the game played on a hiker graph. Also, each strategy is associated with the minimum number of seekers, required for it to work.

2.1 Trivial Capture

At the start of the game, position a seeker agent at each coverage node. The number of seekers required to do so is equal to the number of coverage nodes in the hiker graph. All the hiders are caught and the game ends at the very start.

Theorem 2.1. *Trivial capture strategy deterministically captures all the hiders*

Proof. Assume that the seeker team plays the trivial capture strategy and there exists a hider that is not caught. This implies that there must be a gap node, further implying that there must be a coverage node which is not occupied by a seeker. This is a contradiction since each coverage node is occupied by a seeker. □

Although this strategy deterministically captures all the hiders, it becomes impractical in larger environments due to its high minimum number of seeker requirements.

2.2 Trap

An oracle guided hider can stay hidden at turn $t+1$ if there exists a gap node, which is traversal adjacent to the strategic point which the hider currently occupies. If there exists no traversal adjacent gap nodes, then the hider cannot evade and will be caught. The trap strategy works on this principle.

Before going into details, we introduce a new type of relationship (or edge) between two coverage nodes in $\mathcal{HG}$, to simplify explanation and analysis. If a coverage node A is visibility adjacent to strategic node 1, the strategic node 1 is traversal adjacent to node 2, and strategic node 2 is visibility adjacent to coverage node B, then coverage nodes A and B are said to be capture adjacent (or joined by a capture edge). As shown in figure 3, the coverage nodes A and B follow the above-described criteria and hence are capture adjacent.

The trap strategy comprises two steps, initial arrangement of seekers and subsequent traversal. At the start of the trap strategy, each coverage point of a hiker graph must follow either of the two trap conditions

- Be occupied by a seeker.
- Not be occupied, but have all of its capture adjacent coverage nodes, occupied by seekers.

As the game proceeds, a seeker, in addition to the ones which already occupy coverage points, goes to each non-occupied coverage point. When this seeker would have traversed to each non-occupied coverage point, all the hiders would have been caught.

The number of seekers required for this algorithm is equal to the number of seekers required to satisfy the above two trap conditions, as well as an additional seeker required for traversal. The problem of finding the minimum number of seekers and the coverage nodes which they need to occupy, for satisfying the trap conditions, is formulated in terms of an integer program.

Variables

x_v	$v \in CN$ (set of all coverage nodes)	x_v denotes seeker's occupancy at node v

Objective

$$\min \sum_{v \in CN} x_v \tag{1}$$

Constraints

$$\sum_{v' \in Adj_{cap}(v)} x_{v'} \geq (1 - x_v)|Adj_{cap}(v)| \qquad \forall v \in CN \tag{2}$$

$$x_v \in \{0, 1\} \qquad \forall v \in CN \tag{3}$$

$Adj_{cap}(v)$ is the set of all the nodes which are capture adjacent to node v, and $|Adj_{cap}(v)|$ is the cardinality of this set. Variable x_v is a binary variable, denoting whether the coverage node v should be occupied by a seeker or not. The optimal solution to the above linear program gives the minimum number of seekers required, as well as the coverage nodes which need to be occupied, for satisfying the above two trap conditions.

The second constraint of the integer program enforces both the trap conditions. If a coverage node v is occupied by a seeker, $x_v = 1$ and the constraint becomes

$$\sum_{v' \in Adj_{cap}(v)} x_{v'} \geq 0$$

The constraint implies that the number of coverage nodes which are capture adjacent to v and are occupied by seekers must be greater than or equal to 0. This is a trivial inequality which will always be satisfied.

If a coverage node v is not occupied by a seeker, $x_v = 0$ and then the constraint becomes

$$\sum_{v' \in Adj_{cap}(v)} x_{v'} \geq |Adj_{cap}(v)|$$

The constraint implies that the number of coverage nodes which are capture adjacent to v and are occupied by seekers must be greater than or equal to the number of all capture adjacent nodes of v. Since LHS cannot be greater than RHS, the only valid solution occurs when both are equal. Therefore, the constraint can be reduced to an equality.

$$\sum_{v' \in Adj_{cap}(v)} x_{v'} = |Adj_{cap}(v)|$$

This is the exact requirement of the second trap condition.

To further reduce the number of seekers required, instead of operating (or analyzing) the entire hiker graph, we operate on its components. The idea is to obtain smaller hiker graphs from the main hiker graph which can be treated as individual games of hide and seek since a hider belonging to one component cannot move to some other component.

To find such smaller hiker components, consider a subgraph of a hiker graph consisting solely of all strategic nodes and their traversal edges. Consider each connected component of this subgraph. For each such connected component, consider all the coverage nodes which are visible adjacent, and the corresponding visibility edges, and add it into the subgraph. This results in a smaller (or equal in size) hiker graph derived from the original hiker graph. The property of this smaller hiker graph is that any hider which occupies a strategic node of this graph cannot leave this graph since all the nodes which this hider can reach are present in the same graph, and all the nodes which are not connected to the occupied node, are not present in the smaller hiker graph. Having obtained these connected strategic node hiker components, we treat each component as a separate Hide-and-Seek game and apply the trap strategy on them, one by one.

Theorem 2.2. *Trap strategy deterministically captures all the hiders*

Proof. Assume that the seeker team plays the trap strategy and there exists a hider that is not captured. Suppose the trap strategy takes few turns to complete. Since the hider remained uncaught, at each turn of the game, the hider would have remained at a gap node. This trivially implies that a hider would have been at a gap node s_i at the very start of the game.
State at turn $t = 0$: The coverage nodes which are visible adjacent to s_i, are unoccupied by seekers since it is a gap node. Let these coverage nodes be represented by $C_{s_i} = \{c_1, c_2, ..., c_n\}$. Due to the second trap condition, the coverage nodes which are capture adjacent to nodes C_{s_i}, are occupied by seekers. This means that any strategic node, traversal adjacent to s_i, is visible adjacent to a coverage node occupied by a seeker. All seekers who occupy such coverage nodes remain static i.e. occupy the same node, throughout the game. This also means that at any turn t, there will be no traversal adjacent strategic nodes which are also gap nodes. If at any turn t, the hider tries to move to its traversal adjacent strategic node, it will be caught. Under the trap strategy, an additional seekers job is to traverse each non-occupied coverage node. At some time T, this seeker will come to a coverage node $\in C_{s_i}$.
State at some turn $0 < t < T$: As proved in the previous paragraph, the hider has to keep on staying at strategic node s_i to stay hidden from seekers.
State at turn $t = T$: Since a seeker occupies one of the coverage points $\in C_{s_i}$, if the hider does not leave the node s_i, it will be caught by this seeker. Thus, to stay hidden, the hider moves to a gap node which is traversal adjacent to s_i. This is a contradiction since such a node does not exist.

□

2.3 Wave

The traversal of hider and seeker agents across the hiker graph can be perceived as flows. The hiders keep occupying gap nodes at each turn, guided by oracles. These gap nodes are created by the non-occupancy of seekers at certain coverage nodes. Thus, seekers influence the flows of hiders. The seekers can force the hiders to traverse to certain gap nodes, eventually capturing them at some point. The wave strategy works on this principle.

The wave strategy comprises two steps, preparation of a plan and subsequent traversal. For plan preparation, a subgraph of hiker graph $\mathcal{HG}$, consisting of only coverage nodes and capture edges (refer trap strategies), is considered. Call this subgraph $\mathcal{HCG}$. A Breadth-first search (BFS) is performed on this subgraph, from some arbitrary node. Performing a BFS partitions the coverage nodes into layers. The nodes which were explored in the first iteration belong to layer 1, those that were explored in the second belong to layer 2, and so on. Each coverage node belongs to a unique layer, $c \in layer(K)$. In other words, $layer(K)$ is a set comprising of all the coverage nodes explored by the BFS traversal algorithm at K^{th}

iteration. For simpler explanation and analysis, we also associate each strategic node $\in \mathcal{HG}$, with one of the layers. A strategic node $s \in layer(K)$, if it is visible adjacent to any coverage node $c \in layer(K)$, and is not visible adjacent to any coverage node $c \in layer(K-1)$ or $layer(K-2)$ ... or $layer(1)$. In other words, the strategic node belongs to the smallest layer (in terms of labeling K) whose coverage node is visible adjacent to it.

At the start of the game, the seekers occupy all the coverage nodes of $layer(1)$ and $layer(2)$. At the next turn, the seekers in $layer(1)$ rearrange to occupy coverage nodes of $layer(3)$, whereas the seekers in $layer(2)$ continue occupying their current nodes. At the next turn, the seekers in $layer(2)$ rearrange to occupy $layer(4)$ and the seekers occupying $layer(3)$ retain their positions. This goes on till the seekers occupy the last two layers.

Before proceeding with the proof of correctness for the strategy, we prove a few lemmas.

Lemma 2.3. *A hider occupying strategic node $s \in layer(K)$ is not traversal adjacent to any strategic node $s' \in layer(K-2)$.*

Proof. Assume there exists a node $s' \in layer(K-2)$ which is traversal adjacent to a node $s \in layer(K)$. Since $s \in layer(K)$, by definition, there must be a coverage node $c \in layer(K)$ which is visible adjacent to s. Similarly, there must be a coverage node $c' \in layer(K-2)$ which must be visible adjacent to s'. By definition, the coverage nodes c and c' must be capture adjacent. Since BFS traversal is performed by considering coverage nodes and capture edges, c and c' must belong to the same layer, or adjacent layers ($layer(K)$, $layer(K-1)$). This is a contradiction since the layers which c and c' belong to ($layer(K)$, $layer(K-2)$) are neither adjacent, nor same. Thus, such a node s' cannot exist. □

Lemma 2.4. *A hider occupying strategic node $s \in layer(K)$ is not traversal adjacent to any strategic node $s' \in layer(K-2)$, or $layer(K-3)$, ..., or $layer(1)$.*

Proof. We have proved the statement for $layer(K-2)$. We can easily prove it for $layer(j)$, $j < K-2$ by replacing $K-2$ with j in the above proof. □

Lemma 2.5. *A hider occupying strategic node $s \in layer(K)$ is not traversal adjacent to any strategic node $s' \in layer(K+2)$, or $layer(K+3)$, ..., or $layer(N)$.*

Proof. We have proved the statement for $layer(K-2)$. We can easily prove it for $layer(j)$, $j \geq K+2$ by replacing $K-2$ with j in that proof. □

Similar to the trap scenario, the number of required seekers can be further reduced by operating, one by one, on the components of the hiker graph having connected strategic node subgraphs.

Theorem 2.6. *Wave strategy deterministically captures all the hiders*

Proof. Before each turn K, the seekers are already occupying all the coverage nodes $\in layer(K-1)$ and $layer(K-2)$. At the start of the turn, the seekers occupying $layer(K-2)$ leave their current nodes and occupy all the nodes of $layer(K)$. For now, assume there exists a sufficient number of seekers for doing this at each turn. We will prove an invariant which will be used to prove the capture criteria.

Invariant: At the start of each turn K of the wave strategy, no hider will occupy any strategic nodes $\in layer(K-1)$, $layer(K-2)$, $layer(K-3)$, ..., $layer(1)$

We will prove this invariant using induction.

Initialization (Base Case): The first turn starts at $K = 3$. The invariant trivially holds for $K = 3$ since all coverage nodes $\in layer(1)$ and $layer(2)$ are already occupied by seekers. Any hider occupying a strategic node $\in layer(1)$ or $layer(2)$ will be caught.

Maintenance (Inductive Step): Assume the invariant holds true for K. As the turn proceeds, the seekers occupying coverage nodes $\in layer(K-2)$ will leave their current nodes and occupy coverage nodes $\in layer(K)$. This traversal is possible since, by definition, any two coverage nodes are traversal adjacent. Now hiders occupying strategic nodes $\in layer(K)$ can either occupy their current nodes or move to a different node. If a hider does not leave its node, it will be captured by some incoming seeker and will be eliminated from the game. If a hider decides to traverse to some different strategic node, it has various choices.

- Can move to another strategic node $\in layer(K)$, in which case it will be caught.
- Can move to a node $\in layer(K-1)$. Since all coverage nodes of this layer are occupied by seekers, the hider will be caught.
- Can move to a node $\in layer(K-2)$, or $layer(K-3)$, ... or $layer(1)$. However, this is impossible by lemma 2.4.
- Can move to a node $\in layer(K+2)$, or $layer(K+3)$, ... or $layer(N)$. However, this is impossible by lemma 2.5.

Thus, a hider occupying a strategic node $\in layer(K)$ can only move to another strategic node $\in layer(K+1)$. In the case of other layers, it is either impossible to occupy their strategic node or the hider will be caught if the hider tries to go there.

Consider a hider occupying a strategic node $\in layer(K+1)$, at the start of turn K. This hider can move to $layer(K)$, but it will be caught. It cannot move to $layer(K-1), layer(K-2), \ldots, layer(1)$ (lemma 2.4). Similarly, a hider occupying $layer(K+2)$ cannot move to $layer(K), layer(K-1), \ldots, layer(1)$. The case for hiders occupying $layer(K+3), layer(K+4), \ldots layer(N)$ is also similar. Thus, hiders occupying $layer(K)$ and beyond, will not be able to occupy $layer(K), layer(K-1), layer(K-2), \ldots layer(1)$, at turn K.

We had assumed that the invariant is true for K, i.e. no hider occupies a strategic node $s \in layer(K-1), layer(K-2), \ldots layer(1)$, and we have shown that any hider occupying a strategic node $s \in layer(K), layer(K+1), \ldots layer(N)$ cannot occupy any strategic node $s' \in layer(K), layer(K-1), \ldots, layer(1)$. This proves the invariant for $K+1$. Since the base and inductive steps both are true, by mathematical induction, the invariant holds true.

Termination At the end of the last turn of the strategy, the seekers would be occupying all the coverage nodes belonging to the last two layers ($layer(N)$, $layer(N-1)$). In the end, no hider will be occupying any strategic node $\in layer(N)$, $layer(N-1)$, ..., $layer(1)$, or in other words, all of the hiders would have been eliminated.

□

The minimum number of seekers required by this strategy, for a particular BFS ordering from some arbitrary node, can be denoted

by the below expression.

$$\max_{j=2:N} \quad \big(|layer(j-1)| + |layer(j)|\big)$$

This expression denotes the maximum number of coverage nodes belonging to two consecutive layers. To find the minimum possible number under this strategy, the minimum number under all the possible BFS orderings needs to be considered.

$$\min_{i}\Big(\max_{j=2:N} \quad \big(|layer_i(j-1)| + |layer_i(j)|\big)\Big)$$

Here i denotes the coverage node from which BFS exploration was started ,and $layer_i(j)$ denotes the set of all nodes explored at the j^{th} iteration when BFS was started from node i.

3 SIMULATION

3.1 Setup

We created a simulator capable of simulating and visualizing 2-D Hide-and-Seek games (Tandon and Karlapalem [19]). It provides a 2-D bounded and continuous playing area, polygonal obstacles and agents embedded with visibility cones. An agent can perceive an opponent agent only if it lies inside its visibility cone. The simulator allows easy interfacing of these game primitives with the spatial abstractions and strategies described in previous sections. The simulator plays the game in discrete time steps. At each step, each agent is supposed to take an action after considering its percepts (visibility cone) and internal state. The simulator accepts the actions and moves the agent in the chosen direction by a fixed distance. In other words, both the hider and seeker agents move with a fixed speed. If a hider agent lies in the visibility cone of the seeker, it is eliminated. The game completion time is the time step at which the last hider is eliminated. The simulator also allows agents of the same type to form teams and coordinate with each other, which is necessary for our trap and wave strategies. We have released the simulator and strategy related code on GitHub (https://github.com/droftware/Medusa).

For empirical analysis and simulations of the spatial abstractions and strategies, we consider four game-maps. Each of these maps has a different arrangement and number of obstacles. We label the maps as 1, 2, 3, 4, in increasing order of number of obstacles, i.e. map 1 has the minimum (figure 1), and map 4 has the maximum number of obstacles (figure 6). Using these maps we empirically evaluate the characteristics of our strategies and spatial abstractions, such as the number of coverage nodes and minimum number of seekers required to play a strategy in a particular map. Furthermore, we also contrast the game completion times across the strategies.

3.2 Resolution and Parameters

Rectangular environments used for simulations have dimensions ranging from 400×300 pixels (map 1) to 1200×700 pixels (map 4). The agents are represented as a single pixel associated with a direction $d \in \mathcal{A}$ (section 1.2). They move at a fixed speed of about 10 pixels/step. Each agent is also embedded with a visibility cone which is constructed by emitting 10 rays in direction d, spread across 90° followed by tracing over the points of obstruction. Limited visibility is enforced by preventing these rays from going outside a bounding square of length 250 pixels, formed around the agent.

Map	Nodes		Trap		
	$\lvert\mathcal{SP}\rvert$	$\lvert\mathcal{CP}\rvert$	$Occupied$	$NonOccupied$	$Min^m Seek^s$
1	22	17	13	4	14
2	41	39	32	7	33
3	58	50	39	11	40
4	121	108	59	49	60

Table 1: Trap strategy's spatial features and minimum seeker requirements

To reduce the computational cost of finding an optimal coverage points set $\mathcal{CP}$ of an environment $\mathcal{E}$, the environment is discretized into grid cells (section 1.4). In all our simulations, we used discretized cells having a resolution of 10×10 pixels. During computation of $\mathcal{CP}$, using a smaller discretization resolution may reduce the number of number of coverage points required for covering the nodes corresponding to a maximal clique of $\mathcal{VG}$ (section 1.4).

3.3 Spatial Features and Minimum Seeker Requirements

We analyze the spatial features on a per strategy basis. Under trap strategy (table 1), for each of the maps, we compute the coverage nodes which ought to be occupied, coverage nodes that do not need to be occupied and the minimum seekers required to play the strategy. Under wave strategy (table 2), we compute the total number of layers, the maximum number of nodes belonging to a layer, and the minimum number of required seekers.

Since the hiker graph depends only on the map and is independent of the strategy used, the number of strategic and coverage nodes is the same for each map, under both the strategies.

Two points are worth noting.

(1) Features such as strategic nodes, coverage nodes, minimum required seekers, occupied nodes (table 1), layers (table 2), increase as the number of obstacles increase (Recall that maps were ordered in increasing order of number of obstacles). However, this is not always true. Map 4 has the maximum number of obstacles, but under wave strategy, features - $MaxNodes$ (maximum number of nodes in a layer) and $MinSeek^s$ (minimum number of seekers required), have smaller values when compared to corresponding features of map 2 and 3.
(2) Minimum number of seekers required to play wave strategy is always less than the number required to play trap strategy. The difference is especially noticeable in map 4. Trap requires 60 seekers whereas wave requires only 22.

3.4 Game Completion Times

We analyze game completion times by simulating both the strategies on maps 1 and 4. Trap and wave strategies assume that a hider would always occupy strategic points. In a simulation, the hiders cannot be forced to occupy only strategic points. In some games, a hider may occupy some non-strategic points and stay hidden for a very long time (or even forever). These situations can be eliminated (or reduced) by increasing the granularity of strategic points, i.e. instead of just mid-points of obstacles, consider an even

Map	Nodes		Wave		
	$\|\mathcal{SP}\|$	$\|\mathcal{CP}\|$	$Layers$	$MaxNodes$	$Min^m Seek^s$
1	22	17	4	8	13
2	41	39	4	19	29
3	58	50	5	20	33
4	121	108	13	11	22

Table 2: Wave strategy's spatial features and minimum seeker requirements

greater number of points which surround obstacles, to eliminate the existence of a point which is not covered by a coverage point. We tackle this, not by increasing the granularity of strategic points, but by restarting the strategy if some hiders exist after the strategy has completed all its steps. The hider agent used in simulation follows a random action strategy. At the start of a game, they occupy a random point which is not necessarily a strategic point. At each step of the game, they take a random action.

For each map (1 and 4), and strategy (trap and wave) combination, we simulated ten Hide-and-Seek games. Consider figure 4, which shows the results for simulations done using Map 1. For both the strategies, the minimum possible number of seekers were used in all the simulations. In the case of wave strategy, the simulations were done using 13 seekers, and in the case of trap they were done using 14. 20 hiders were used in both the scenarios. Both the strategies perform similarly, giving a median game completion time in the range of 200-300 steps. In case of wave strategy, there were two simulations (out of 10) which had much higher game completion times (569, 776). This may be attributed to some hiders not getting captured after the first run of the wave strategy, leading to delay because of subsequent runs of the wave strategy.

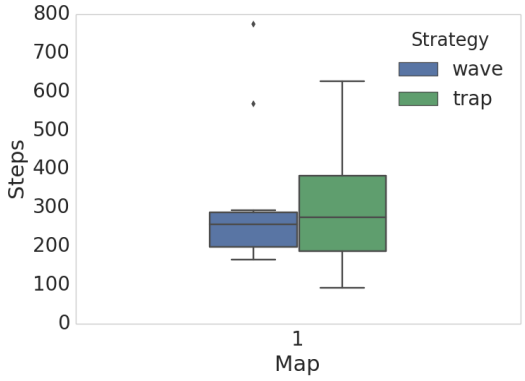

Figure 4: Game completion times for Map 1

Consider figure 5, which denotes the results for simulations done using Map 4. Similar to above, here also we use minimum possible number of seekers, in each simulation (50 when playing trap, and 22 when playing wave). The number of hiders used is 50 in all the simulations. Compared to Map 1, the game completion times are not similar. In the case of wave, the median is 2626.3 game steps, whereas in the case of trap it is 962.4. This can be directly attributed to the greater number of seekers used in the case of trap strategy. The variance exhibited by the wave strategy is greater than that of trap. When compared to Map 1, there is a magnitude of difference in game completion times.

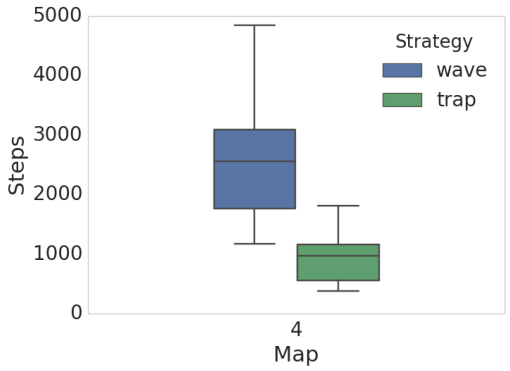

Figure 5: Game completion times for Map 4

Figure 6: Environment corresponding to Map 4

4 CONCLUSION

We have presented two seeker strategies - trap and wave, which guarantee the capture of all hiders, in a Spatio-temporal graph abstraction, under a minimum seeker requirement criteria. The agent model, upon which we have built our strategies, has a limited visibility range. Limited visibility allows an agent to see the environment only till a certain maximum distance. This is in contrast to many guaranteed capture pursuit-evasion methods that assume unlimited visibility range. We proved that both the strategies deterministically capture all the hiders, even in the presence of an oracle. We also formulated the minimum number of seekers required for each of the strategies, to guarantee the capture of all hiders. Lastly, we introduced abstractions for encapsulating the spatial features of a 2-D landscape environment such as obstruction, coverage, and connectivity.

REFERENCES

[1] Steve Alpern. 1974. The search game with mobile hider on the circle. *Differential games and control theory* (1974), 181–200.

[2] Steve Alpern and Shmuel Gal. 2006. *The theory of search games and rendezvous*. Vol. 55. Springer Science & Business Media.

[3] Bowen Baker, Ingmar Kanitscheider, Todor Markov, Yi Wu, Glenn Powell, Bob McGrew, and Igor Mordatch. 2019. Emergent tool use from multi-agent autocurricula. *arXiv preprint arXiv:1909.07528* (2019).

[4] Coen Bron and Joep Kerbosch. 1973. Algorithm 457: finding all cliques of an undirected graph. *Commun. ACM* 16, 9 (1973), 575–577.

[5] Timothy H Chung, Geoffrey A Hollinger, and Volkan Isler. 2011. Search and pursuit-evasion in mobile robotics. *Autonomous robots* 31, 4 (2011), 299.

[6] Fei Fang, Peter Stone, and Milind Tambe. 2015. When security games go green: Designing defender strategies to prevent poaching and illegal fishing. In *Twenty-Fourth International Joint Conference on Artificial Intelligence*.

[7] Joe G Foreman. 1974. The princess and the monster on the circle. *Differential Games and Control Theory* (1974), 231–240.

[8] Shmuel Gal. 1979. Search games with mobile and immobile hider. *SIAM Journal on Control and Optimization* 17, 1 (1979), 99–122.

[9] Brian P Gerkey, Sebastian Thrun, and Geoff Gordon. 2006. Visibility-based pursuit-evasion with limited field of view. *The International Journal of Robotics Research* 25, 4 (2006), 299–315.

[10] RUFUS Isaacs. 1965. Differential Games, SIAM Series in Applied Mathematics. (1965).

[11] Nimrod Megiddo and SL Hakimi. 1978. *Pursuing Mobile Hiders in a Graph*. Technical Report. Northwestern University, Center for Mathematical Studies in Economics and ….

[12] Nimrod Megiddo, S Louis Hakimi, Michael R Garey, David S Johnson, and Christos H Papadimitriou. 1988. The complexity of searching a graph. *Journal of the ACM (JACM)* 35, 1 (1988), 18–44.

[13] Torrence D Parsons. 1978. Pursuit-evasion in a graph. In *Theory and applications of graphs*. Springer, 426–441.

[14] Praveen Paruchuri, Jonathan P Pearce, Janusz Marecki, Milind Tambe, Fernando Ordonez, and Sarit Kraus. 2008. Playing games for security: An efficient exact algorithm for solving Bayesian Stackelberg games. In *Proceedings of the 7th international joint conference on Autonomous agents and multiagent systems-Volume 2*. International Foundation for Autonomous Agents and Multiagent Systems, 895–902.

[15] Praveen Paruchuri, Jonathan P Pearce, Milind Tambe, Fernando Ordonez, and Sarit Kraus. 2007. An efficient heuristic approach for security against multiple adversaries. In *Proceedings of the 6th international joint conference on Autonomous agents and multiagent systems*. ACM, 181.

[16] James Pita, Manish Jain, Janusz Marecki, Fernando Ordóñez, Christopher Portway, Milind Tambe, Craig Western, Praveen Paruchuri, and Sarit Kraus. 2008. Deployed ARMOR protection: the application of a game theoretic model for security at the Los Angeles International Airport. In *Proceedings of the 7th international joint conference on Autonomous agents and multiagent systems: industrial track*. International Foundation for Autonomous Agents and Multiagent Systems, 125–132.

[17] Ichiro Suzuki and Masafumi Yamashita. 1992. Searching for a mobile intruder in a polygonal region. *SIAM Journal on computing* 21, 5 (1992), 863–888.

[18] Akshat Tandon and Kamalakar Karlapalem. 2018. Agent Strategies for the Hide-and-Seek Game. In *Proceedings of the 17th International Conference on Autonomous Agents and MultiAgent Systems*. International Foundation for Autonomous Agents and Multiagent Systems, 2088–2090.

[19] Akshat Tandon and Kamalakar Karlapalem. 2018. Medusa: Towards Simulating a Multi-Agent Hide-and-Seek Game.. In *IJCAI*. 5871–5873.

[20] Mikhail Il'ich Zelikin. 1972. On a differential game with incomplete information. In *Doklady Akademii Nauk*, Vol. 202. Russian Academy of Sciences, 998–1000.