CAPTURING HIDERS WITH MOVING OBSTACLES

Ayushman Panda
IIIT-Hyderabad
Hyderabad, India
ayushman.panda@research.iiit.ac.in

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

ABSTRACT

The classic hide-and-seek game serves as an abstraction for many real-world scenarios like capturing intruders in a closed space, locating objects, patrolling an area, etc. Since most of the present work is based on static obstacles, we address solutions for the hide-and-seek game in an environment where the obstacles are not static. We design strategies that would facilitate seekers to capture hiders in an environment with moving obstacles. We have three strategies: *Baseline strategy, Set-cover strategy, and Sweep strategy*, which use different surveillance techniques to be followed by the seekers. We simulate the methods and compare their performance in different scenarios. While the baseline strategy demands many seekers in large environments, the other two strategies, set-cover, and sweep, are ideal for applying in large environments as they require fewer seekers in the same environment.

1 Introduction

In this study, we created a simulation environment for the game of hide-and-seek, in which we included agents and obstacles. The agents were divided into two groups: hiders and seekers. Hiders were tasked with using hiding spots around obstacles to stay hidden, while seekers tried to find and eliminate the hiders. The obstacles independently moved around the environment, making it difficult for the seekers to find hiders hiding near them. All agents and obstacles followed specific properties that guided their interactions with each other and with the environment.

The seekers used an Oracle to guide them towards the last known position of hiders. The properties of the environment, agents, and obstacles were defined in section 2 of the paper. We implemented the hide-and-seek game and ran it for a set number of steps before announcing the results.

We also developed two new strategies for seekers to capture hiders, in addition to a baseline strategy. We simulated these strategies and compared their performance based on the number of steps required to complete a game round in a large environment with different types of obstacle movement, as discussed in section 3.1.

2 Agents and Environmental Properties

We are given an environment **E.** The environment consists of obstacles and agents, both of which can move in the environment. **E** is divided into a *M X N* grid. Every obstacle and agent in the environment is in a specific cell (i, j) in the grid, $0 \leq i < M$ & $0 \leq j < N$.

2.1 Agents

Agents are mobile autonomous individuals which belong to the environment. They are classified into hiders and seekers category. Each type of agent works with a common interest towards an end goal. Every movement or action taken by an agent is in best interest with respect to the target they want to achieve. An agent takes decision individually based on the common strategy they are following.
The *field of vision,* **v** of an agent is the number of cells an agent can track from the current position. If an obstacle comes in the visibility range of any agent, the agent cannot track cells beyond the obstacle. The field of vision of an agent is hence blocked by obstacles. A cell which falls under the field of vision of an agent is called *visible adjacent cell.*

2.1.1 Hiders

Hiders (H) are intruders in the environment. The goal of hiders is to stay in the environment E, as long as possible.

$$H_j \, \epsilon \, E, \; 0 \leq j < NH \tag{1}$$

where, NH is number of hiders present in E

The hiders have to escape from the seekers *field of vision* in order to avoid getting caught by any seeker agent. Once a hider agent is caught by a seeker, the hider is expelled from the environment. The hiders may hide from seekers field of vision by hiding behind obstacles. The default movement of hiders in the environment is to keep switching obstacles to stay hidden in the environment. Hiders are completely exposed in the environment if they are not hiding behind any particular obstacle (occupying an empty grid cell) in any time instance.

2.1.2 Seekers

Seekers (S) are guardians of the given environment **E,** who protect it from intruders (hiders).

$$S_k \, \epsilon \, E, \; 0 \leq k < NS \tag{2}$$

where, NS is number of seekers present in E

The goal of seekers is to keep the environment clear of all hiders as soon as possible. The seekers keep patrolling (traversing) in the environment looking for hiders. They eliminate the caught hider agents from the environment. An hider agent is caught by a seeker agent when it comes in the *field of vision* of any seeker agent. The visibility of seekers is blocked by obstacles. The seekers have information on the last visited obstacle position of each hider in the environment E from the oracle, but cannot eliminate them unless they come in their *field of vision.* Seekers follow strategies to eliminate all hiders from the environment. We provide strategies which defines traversal path for seekers to eliminate hiders from the environment.

2.2 Obstacles

Obstacles (O) are abstraction for objects/blocks which are arbitrarily placed in the environment.

$$O_l \, \epsilon \, E, \; 0 \leq l < NO \tag{3}$$

where, NO is number of obstacles present in E

Obstacles are classified into *static obstacles* and *dynamic obstacles*. *Static Obstacles* remain associated with the same cell in the grid indefinitely, while *dynamic obstacles* occupy different empty cells in the grid through the course of the hide-and-seek game. An obstacle may move to an adjacent empty cell. When more than one empty cell is available for an obstacle to move, it may occupy any of the available cells randomly. Obstacles block the field of vision of agents. Moving obstacles are helpful to the hiders in increasing their hiding time in the environment, while it is a trouble for the seekers since the moving of obstacles increases the elimination time of hiders by sheltering them. Obstacles are 2-dimensional blocks in the environment E. Since an obstacle blocks the field of vision of agents, a seeker can never keep track of all edges of the obstacle at once. Hence, hiders associated with the exposed surface of an obstacle can only be caught and eliminated by a seeker. Seekers may however form strategy to surround an obstacle from all sides and put all corresponding edges into their combined field of vision, ultimately capturing all hiders associated with that obstacle.

2.2.1 Strategic Points

Strategic Points (SP) are hiding points for hider agents. It is defined on the mid point of the edge of obstacles. It can be assumed as physical hiding spaces available for the hiders in the environment.

$$SP_j \, \epsilon \, O_i, 0 \leq i < NO, 1 \leq j \leq 4 \tag{4}$$

where, NO is number of obstacles present in E

Hiders remain associated with one of the SPs to take advantage of obstacle and remain out of seeker's visibility range.

2.3 Coverage Points

Coverage points (CP) are ideal seeking points for seekers in the grid, where seekers can take position to capture hiders.

$$CP_j \, \epsilon \, E, j \geq 1$$

The required number of CP is calculated based on the strategy we decide. The position of CP can be changed by the seeker teams throughout the course of the game based on the need of the strategy they are following. Unlike SPs, CPs are logical positions in the environment, which means that coverage points are imaginary cell positions where the seekers can be placed. The position of CPs are known only to the seeker agents.

3 Hide-and-Seek Game

The game starts with dividing the Environment *E* into an MXN grid. Obstacles (O) are arbitrarily placed in the grid. Hiders are allowed to take position in the environment first. Hiders ideally occupy all SPs initially. If number of hiders are greater than the number of SPs, the additional hiders occupy random unsafe cells. After hiders take position, the seeker team is alarmed. The seekers gather information about the position of hiders from the Oracle. The seeker team chooses a strategy and decides the required number of seekers (NS). The seekers then marks the position of the ideal coverage points in the environment, which may be updated during the course of the game based on the chosen strategy. The game continues until completion [see section 3.2].

3.1 Hider-Seeker Obstacle Movement

Although the movement of obstacle is arbitrary to a large extent, but the overall movement of obstacles can be:

- *Hider-friendly* : While following hider-friendly movement, the obstacles move away from the overall visibility region of the seekers. Hider-friendly obstacle movement would demand determining the position of CPs for the seekers again, to get all SPs within the visibility region. Hider-friendly obstacle movement help hiders to remain hidden in the corresponding SPs for longer duration.
- *Seeker-friendly* : While following seeker-friendly movement, the obstacles prioritise moving to a cell which is already within the field of vision of the seeker-team. Seeker-friendly obstacle movement helps seekers to continue their surveillance in the environment with minimum change in the position of CPs.
- *Completely-random* : When obstacles follow a completely-random movement, the environment is not friendly to either agents and the selection of cell is completely random.

We use different obstacle movement techniques while evaluating strategies and concluding results [see section 5.1].

3.2 Game Completion

We define a limit ***L,*** which denotes the maximum number of game steps allowed for both the agents. The verdict of the game is guaranteed at the end of *L* game steps.

- *Seekers victory* : If the seekers eliminate all hiders from the environment before *L* game steps, the seekers declare victory and the game terminates.
- *Hiders Victory* : If all hiders are not caught at the end of *L* game steps, the seekers lose. The hiders declare victory and the game terminates.

3.3 Strategies for Static Obstacles

The *Trap* and *Wave* strategies [Tandon and Karlapalem(2020)] does not work in case of moving obstacles. The reason for it being the demand for a consistent hiker graph [Tandon and Karlapalem(2020)]. The hiker graph maps the position of SPs and CPs in the form of a graph which remains the basis for the Trap and Wave strategy. However, in the case of moving obstacles, the position of SPs is not fixed since it changes based on the position of the obstacle.

Lemma 1.1 : We cannot get a consistent hiker graph in case of moving obstacles

Proof: We are given a 4 x 4 environment with 2 obstacles and 4 strategic points. Each obstacle has 2 strategic points associated with it.
If we construct a hiker graph for a given instance of the environment, we expect it to be unchanged until the strategy is

completed. Since the probability of an obstacle moving in a given time t is non-zero (by definition), we are likely to encounter a situation where the position of an obstacle changes from one cell to another in the grid. With the change in the position of the obstacle, the position of associated SPs changes too. With changes in the environment, we have to reconstruct the hiker graph since the association of nodes and edges in the original hiker graph is no more valid. Hence, it is proved that we cannot obtain a consistent hiker graph if there are moving obstacles in the environment.

4 Strategies for Moving Obstacles

4.1 Baseline Strategy

As an initial solution, we assign a seeker to guard each *strategic point.* The hiders are caught as soon as the game starts since each of the SPs falls in the field of vision of the corresponding seeker guarding it. It takes the seekers only one game step to eliminate all hiders from the environment and declare *Seekers victory.*
Lemma 2.1: Initial Strategy captures all hiders in the environment in one game step. **Proof:** We will prove this with contradiction. Suppose each hider arbitrarily chooses any of the available SPs to hide and the seeker team decides to play the *Initial strategy*. If we assume that there is a hider left uncaught after the first game step, this would mean that there was a strategic point that was not in the field of vision of a seeker, which allowed the hider to escape or stay hidden. This is a contradiction since each SP is guarded by a seeker by definition. It is impossible for a hider to stay hidden or escape from a strategic point that is actively being observed by a seeker. Hence, it is proved that no hider can escape if the seeker-team follows the *Initial strategy*.

4.2 Set-Cover Strategy

We use the earlier defined *offset, v* [see section 2.1] to represent the *field of vision* of agents in the environment. The set-cover strategy works on the principle of using the field of vision property to minimize the number of coverage points and determine their respective positions such that each coverage point can track as many strategic points as possible.

For the given environment, and offset set to 1, we can track all cells which are adjacent to the agent. The 4 strategic points are present at (1,1), (1,3), (3,3), and (3,1). We see that if we place a seeker at cell *(2,2)*, we can track the following cells : (1,1), (1,2), (1,3), (2,1), (2,3), (3,1), (3,2), (3,3). We can observe that a single coverage point can track all the strategic points in this environment from Figure 1.

4.2.1 Finding Coverage Points

Given an environment with S strategic points and offset set to v, we design an algorithm to compute the minimum number of coverage points to track S strategic points.

- We find the visible adjacent cells for each strategic point in the given environment **E**. The visible adjacent cells for a given strategic point is determined by **v**.
- ***Visible Adjacent List (VAL)*** keeps track of strategic points which are visible adjacent to any other strategic point. This information helps us find intersecting cell positions to place seeker agents.
- We note all the cell positions which can track one or more strategic points in the given environment. The cells which track more strategic points are given a higher rank. We derive all the possible positions of coverage points with the help of *Visible Adjacent List*
- We get the minimum number of cell positions which would cover all the strategic points from ***Possible_CP***. This becomes a classic *set-cover problem*, which is a known NP-hard problem [Akhter(2015)].
- After finding the minimum number of sets (or, *Coverage Points*) from the above step, we find the ***position of each Coverage point*** by taking intersection of all strategic points in each set.

4.2.2 Analysis on the number of Coverage Points.:

Lemma 2.2: Set-cover strategy computes the minimum set of CPs in a given instance of the environment **E.**

Proof:

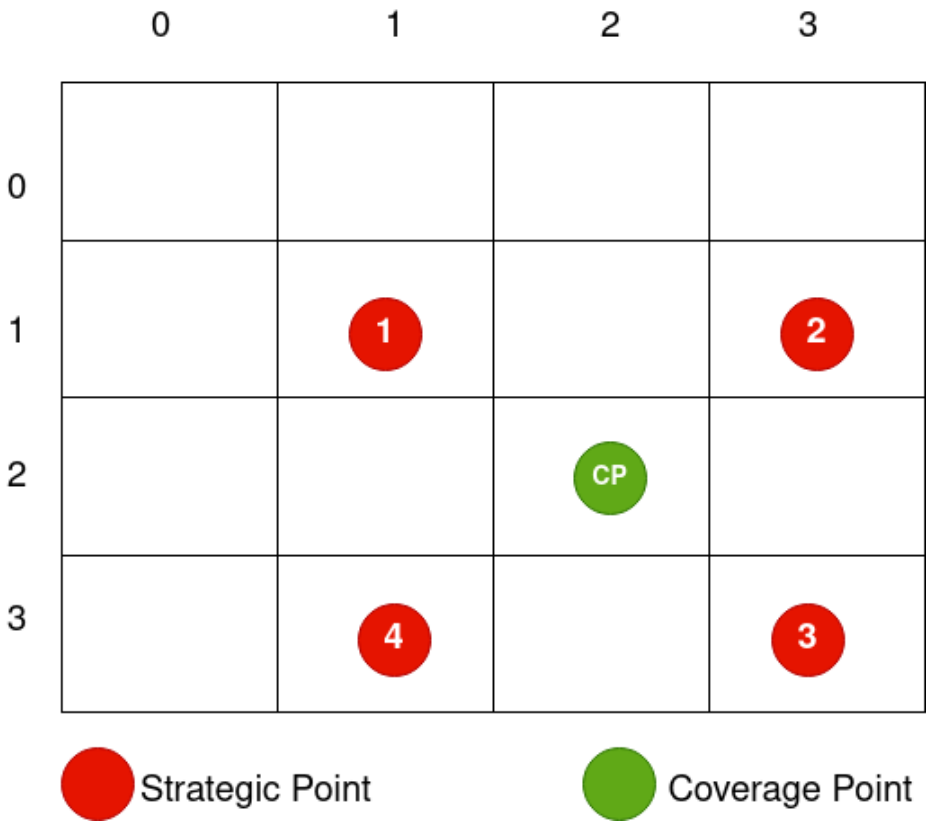

Figure 1: A 4x4 grid, where a single coverage point tracks 4 strategic points.
A 4x4 grid, where a single coverage point tracks 4 strategic points.

Algorithm 1 Computing Visible Adjacent List (VAL)

$S \leftarrow Number\ of\ Strategic\ Points$
$v \leftarrow Field\ of\ Vision$
$VAL \leftarrow [\quad]$ ▷ Initialising VAL
$i \leftarrow 1$
while $i \leq S$ **do**
  $VAL[\,i\,] \leftarrow \phi$
  $j \leftarrow i + 1$
  while $j \leq S$ **do**
    if $pos(i) + v \geq pos(j)$ **then**
      VAL[i] . insert(j) ▷ j is visible adjacent to i
    end if
    $j \leftarrow j + 1$
  end while
  $i \leftarrow i + 1$
end while

In an environment (*m* x *n*) with offset set to *v*, the maximum number of cells that can be tracked by a seeker from a single cell can be given by : $8 * \frac{v*(v+1)}{2}$

- The number of cells being tracked gets reduced in case of edge cells, since the area outside the defined grid is not considered as part of the environment **E**.
- While there might be overlaps in the field of vision of different seeker agents and more than one seeker tracking a single cell, we try to minimise visibility overlaps so that more cells in the environment can be tracked by the same set of coverage points.

Algorithm 2 Computing Possible_CP

$VAL[\ ] \leftarrow Visible\ \ Adjacent\ \ List$
$Possible\_CP \leftarrow [\,]$ ▷ Initialising Possible_CP
$i \leftarrow 1$
while $i \leq S$ **do**
  $t \leftarrow \{\}$ ▷ stores common cells for i-th strategic point
  if $VAL[\ i\ ] = \phi$ **then**
    $t\ .\ insert(\ i\ )$
  else
    $j \leftarrow 1$
    while $j \leq S$ **do**
      if $VAL[\ i\ ] \cap VAL[\ j\ ] \neq \phi$ **then**
        $t\ .\ insert(\ j\ )$ ▷ i, j visible adjacent from a common cell
      end if
    end while
  end if
  $Possible\_CP\ .\ insert(t)$
end while

- Hence, coverage points are ideally placed at a (Manhattan) distance of *v*, whenever possible. In an ideal environment, there are no visibility overlaps. The total number of coverage points (*n(CP)*) required to cover the entire environment (*m* x *n* cells) can be represented as : $n(CP) * (8 * \frac{v*(v+1)}{2}) \geq m * n$

  Therefore, the minimum number of CPs required to cover all the cells in the environment is : $n(CP) = * \frac{(m*n)*2}{8*(v*(v+1))}$
- While the above expression gives us the minimum number of coverage points to cover the entire environment, we do not need keep the entire grid under the visibility range of seekers. We aim to keep the strategic points in the visibility range of the seeker team.
- Hence, in the worst case we might encounter a situation where the strategic points are significantly more and we need to place our seekers such that the whole environment grid is covered. In any case, the number of required seekers would not be more than what needed to cover the entire environment grid.

$$n(CP) \leq \frac{(m*n)*2}{8*(v*(v+1))}$$

Once we find the minimum number of coverage points and their respective position in the environment, we check whether the same set of coverage points still have the strategic points under the seeker team's combined field of vision even after obstacles move. The number and position of coverage points remain the same if all the strategic points are still within the field of view of the seekers, else we run the above algorithm again to compute a new set of coverage points. It is to be noted that if we place a seeker at every coverage point computed through a set-cover strategy, the game would end as soon as it starts since all the strategic points would be guarded by a seeker agent. Since multiple strategic points are tracked instead of just one whenever possible, the number of seekers required would still be very less than the baseline strategy [see section4.1]. We simulate the set-cover strategy with partial seekers too and discuss the observation in various scenarios in section 5.1 of the paper.

4.2.3 Cell Ranking

While calculating the minimum number of CPs and their respective position, we will often encounter more than one cells that cover the required strategic point. We cannot keep both CPs as we have to minimize the overlap of the tracking area. To solve this issue, we rank cells in the case of ties. The ranking is done based on the number of new cells tracked by each of the possible CPs, in addition to existing tracked cells by already chosen CPs.

The cell which adds more tracked cells into the seeker's field of view gets a higher rank. Adding CPs that increase the coverage area would result in less re-arrangement of CPs when the obstacles move in subsequent steps. This will also ensure avoiding overlaps to a great extent and facilitate seekers to keep more grid cells in their field of vision. This results in less number of cells that are not tracked by the seekers, and hence less chance of obstacles to move outside the seeker's field of vision. We hence maximize the overall seeker's vision with existing Coverage Points through *Cell Ranking*.

Visible Adjacent Set (V)	
V_1	[(0,0), (0,1), (0,2), (1,0), (1,2), (2,0), (2,1), (2,2)]
V_2	[(0,3), (1,3), (1,4)]
V_3	[(2,2), (2,4), (3,2), (3,4), (4,2), (4,3), (4,4)]
V_4	[(1,2), (1,3), (1,4), (2,2), (2,4), (3,2), (3,4)]
V_5	[(3,0), (3,1), (4,1)]

Table 1: Visible adjacent set

Visible Adjacent List(VAL)	
VAL[1]	[3, 4]
VAL[2]	[4]
VAL[3]	[4]
VAL[4]	[ ]
VAL[5]	[ ]

Table 2: **Visible Adjacent List**

4.2.4 Example

Consider a 5 x 5 grid with 5 obstacles. Let us assume one strategic point is associated with an obstacle. So, there are five strategic points at a random position in the environment. The offset is set to 1, which allows an agent to have a visibility region of 1 cell in any direction within the grid. Each cell in figure 2 represents the strategic points that are visible adjacent to it.

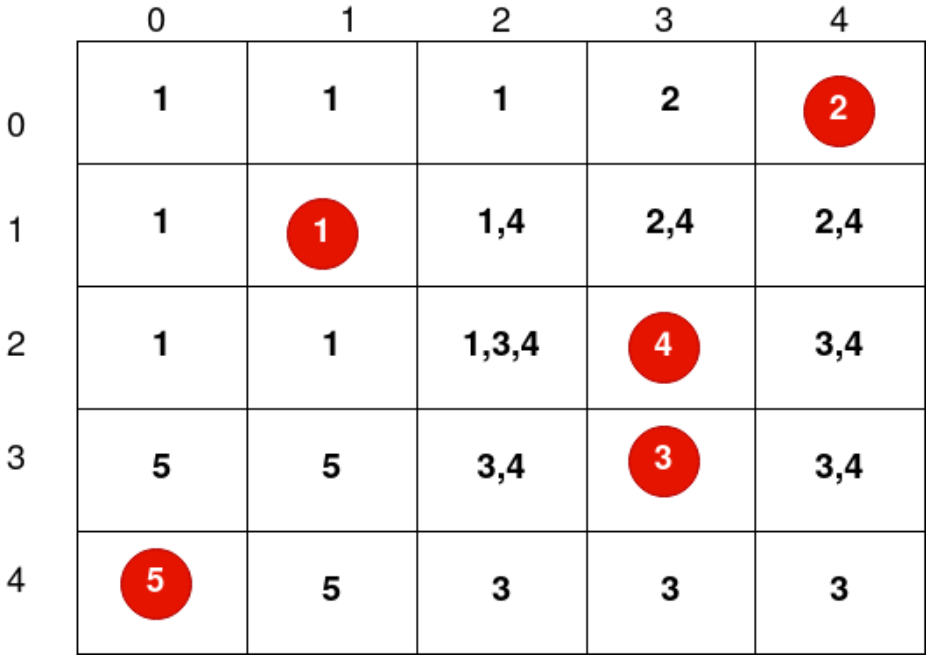

Figure 2: A 5x5 grid, where each cell is marked with the strategic points that can be tracked from it.
A 5x5 grid, where each cell is marked with the strategic points that can be tracked from it.

Set V_i contains the list of coordinates of cells visible adjacent to the i^{th} strategic point [see Table 1]. The Visible Adjacent List contains list of strategic points which share at least 1 common visible adjacent cell with the i^{th} Strategic Point [see Table 2].

Possible_CP stores all possible combinations of strategic points which can be tracked from a single cell. This is determined with the help of *VAL*. For example, VAL[1] contains [3, 4], which gives the following information

- strategic point 1 is visible adjacent to some cell in the grid which is also visible adjacent to strategic point 3.
- strategic point 1 is visible adjacent to some cell in the grid which is also visible adjacent to strategic point 4.
- We also derive that strategic point 4 is visible adjacent to strategic point 3 (from VAL[3]) , so we conclude that there exists some cell ((2,2)) which is visible adjacent to strategic points 1 , 3 and 4
- So we add the above information to *Possible_CP*.After the above iteration, Possible_CP = [(1,3), (1,3,4), (1,4)]

If VAL[i] is NULL , we add *i* to *Possible_CP*. For this simulation, after considering all values in the list ***VAL[]***,

Possible_CP = [(1,3), (1,3,4), (1,4), (2,4), (3,4), (4), (5)]

From **Possible_CP**, we find the minimum number of cells where we can place a Coverage Point, so that all the strategic points are tracked. In this example we conclude from *Possible_CP* that there exists a cell from where strategic points 1,3 and 4 can be tracked. Now, we need to find a cell from where strategic point 2 and 5 can be tracked. From *Possible_CP*, we derive that there exists a cell in the grid from where strategic point 2 and 4 can be tracked, but there is no common visible adjacent cell for strategic point 5 with any other Strategic Points. To track coverage point 5, we have to place a coverage point in one of the cell which is visible adjacent to it. So , with **3 coverage points**, we can ensure that all the 5 strategic points in the given environment can be tracked.
To find the co-ordinate of cell where the 2 coverage point would be placed, we take intersection of respective list from V_i

- *Coverage Point 1* : (1,3,4) : $V_1 \cap V_3 \cap V_4$ = {(2,2)}
- *Coverage Point 2* : (2,4) : $V_2 \cap V_4$ = {(1,3), (1,4)}
- *Coverage Point 3* : (5) : $V_5 \cap$ =

When we do not get any common cell to track a strategic point, we may choose a cell from visible adjacent set [see Table 1]. Hence we may consider any of the cell corresponding to V_5 for a possible coverage point. If we get multiple cells satisfying the condition to become a coverage point, we choose a cell with the highest rank [see section 4.2.3]. In case of more than one cell with the highest rank, we can conclude that all the cells with the same rank add same number of cells to seeker's vision. Hence, we may pick any of the cells as a Coverage Point. In the above case, we can choose either cell (1,3) or (1,4) to place *Coverage Point 2*. But, (1,3) covers 5 new cells while (1,4) covers only 3 new cells. Hence (1,3) is chosen according to rank. Similarly, *Coverage Point 3* is placed at (3,1) as it covers more new cells than other cells in consideration.

4.3 Sweep Strategy

Sweep strategy works on the principle of *Trap Strategy [Tandon and Karlapalem(2020)]*, but on individual obstacles instead of *hiker graph*. Sweep strategy aims at converging the available environment for hiders through a surveillance technique, which limits the available hiding places for the hiders.

The sweep strategy works on the principle of covering all the grid cells of the environment, starting from one corner of the given environment. While traversing through the grid, the seekers also ensure to not leave any gap nodes for the hiders to escape to an area which has already been inspected by the seekers. Any obstacle, static or moving, which comes in the field of vision of the seekers while surveillance is scanned for hiders and marked safe after eliminating all hiders associated with it.

Taking the example of an environment with side *s*, and offset *v* set to 0 :

- The seekers start searching for hiders from one of the corners of the environment.
- The seekers capture all the hiders associated with any obstacle which falls under their visibility range. Since v=0 in this example, the seekers capture hiders present in their own cells. Similarly, if v = *x*, the seekers can capture hiders from *x* grids adjacent to their current cell.
- When an obstacle is encountered by the seeker team, additional seekers are deployed to inspect all the corresponding strategic points of that obstacle.
- The additional seekers capture every alternate strategic point in the obstacle and one more seeker visits the unguarded nodes in subsequent steps. If any hider agent is found during the process, the hider is eliminated from the environment. The obstacle is marked safe after the seeker team visits all the strategic points.
- The number of additional seekers required to inspect a particular obstacle depends on the number of strategic points associated with an obstacle. For an obstacle with n_{sp} strategic points, we need $n_{sp}/2 + 1$ seekers.

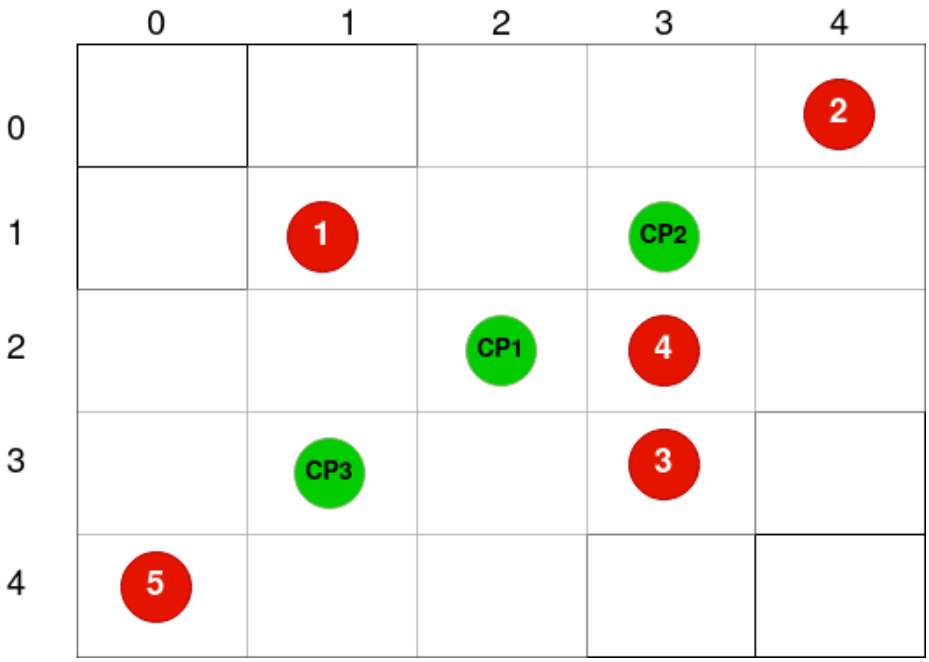

Figure 3: Position of coverage points calculated from set-cover strategy
Position of coverage points calculated from set-cover strategy

- The number of game steps required to inspect an obstacle is again dependent on n_{sp}. Seekers need one game step to occupy $*n_{sp}/2$ cells, while the additional seeker takes $n_{sp}/2$ steps to traverse the remaining strategic points. So, we can conclude that seekers need $n_{sp}/2 + 1$ game steps to mark an obstacle safe.
- In the next step, seekers occupy the next series of un-visited grid cells while ensuring that there are no gap nodes.
- The seekers capture hiders from their recently occupied cell.
- Seekers keep occupying new cells, until all the cells in the grid are visited by the team of seekers.
- When the last cell is visited, the seekers conclude that all obstacles are searched for hiders, and hence all hiders are captured.

4.3.1 Summarising Sweep Strategy

- *No of Coverage Points required for Sweep Strategy* : The maximum number of Coverage points required during the game is the number of seekers required to keep track of the longest sequence of cells in the grid (diagonal, in case of a square/rectangular environment).
- *Role of Offset* : Since offset defines the visibility range of seekers, increase in offset value would require less number of seekers to inspect cells in the grid.
- *Game completion time*: The game completion time is dependent on the number of steps required to cover the entire grid. In our example, a square environment with each side *a*, has its diagonal as the longest sequence of cells in the grid. Since *v = 0*, we need as many seekers as there are cells in the diagonal. The number of steps required by the seekers to cover the entire environment is : *2a - 1*. This is however without any obstacles present in the environment. Each obstacle would take additional $n_{sp_i}/2 + 1$ game steps to be marked safe by the seekers, *0 < i < NO*, where NO is number of obstacles.

4.3.2 Example

Consider a square environment divided into a 5 x 5 grid. The offset, **v** is set to 0 and let the number of moving obstacles be 5. Each obstacle has one strategic point associated with it and there are 3 hiders introduced in the environment. This particular example would take $2a - 1 + n_{sp}$ (here, $n_{sp} = 5$) steps to traverse from one end of the environment to another diagonally. Since each obstacle has one strategic point associated with it, it would take the seekers one game step for each obstacle to mark safe.

- The 5 obstacles take any arbitrary position in the environment *(shown in red)*. Each hider is associated with any one of the strategic points.
- The seeker team *(shown in green)* starts searching for hiders. The first seeker is placed in one of the corners of the environment *(0,0)*. Visibility of seekers is limited based on **v.** If any obstacle comes under its visibility region, additional seekers are deployed to inspect the corresponding strategic points.
- The seeker is accompanied by another seeker for the next traversal step. Since the visibility of seekers is limited to the cell they are currently on in this example, the seekers take the immediate next cell towards the diagonal. This ensures absence of any gap nodes for the hiders to escape to already visited cells.
- The seekers continue the above step for $2a - 1 + n_{sp}$ times. At the end of last game step, the seeker team would have visited all the cells and captured the respective hiders and can conclude that all the hiders are captured.
- The nine game steps can be summarised as shown in Figure 4. Since each obstacle has one strategic point associated, it would take additional five intermediary steps to mark the five obstacles safe. *The obstacle isn't shown in the figure once marked safe.*

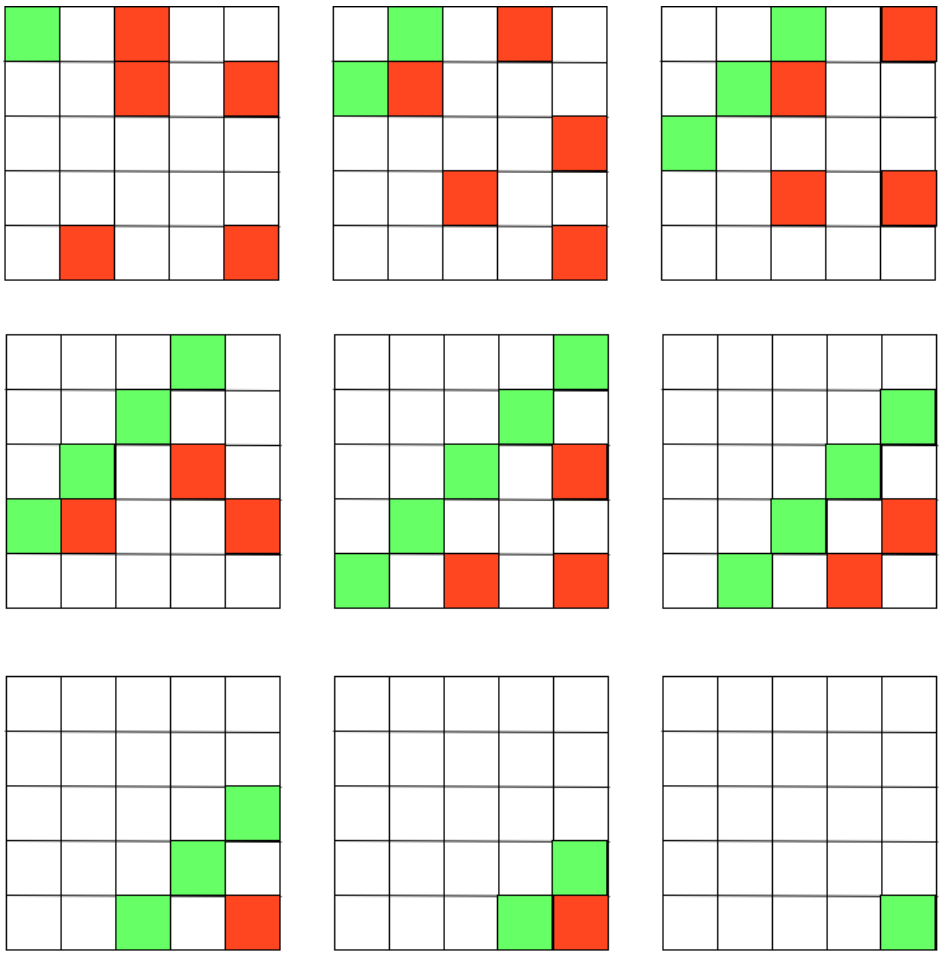

Figure 4: Seekers eliminating hiders following sweep strategy. The green cells indicate active position of seekers. The red cells shows obstacles which are not yet marked safe.

Seekers eliminating hiders following sweep strategy

5 Simulation

We simulate the *set-cover strategy* and the *sweep strategy* with two different 2-D environment maps (MAP-1 and MAP-2). The maps provide a bounded playing area for the hide-and-seek game. The simulation includes all the spatial features and abstraction from the defined environment [see section 2]. The simulation occurs in terms of discrete game

Environment Properties					
	Dimension	$\|Cells\|$	$\|Obstacles\|$	$\|SP\|$	$\|Hiders\|$
MAP-1	50x50	2500	500	2000	300
MAP-2	100x100	10000	5000	20000	2500

Table 3: Properties of environments

steps. In each step of the game, every agent and obstacle is expected to take an action. Since the movement of agents is based on a strategy and obstacles may choose to remain associated in the same cell, staying in the same cell is a valid action for both, agents and obstacles. The properties of each environment is listed in table 3.

Both the maps differ on the basis of grid size and number of obstacles present. Each obstacle has four strategic points associated with it. We run the game for each map-strategy pair with a significantly high value of **L**. The high vale of *L* resulted in seeker's victory in every round. Since the main idea of the simulation is to evaluate seeker's strategies based on the game completion steps, we choose a high value for *L*. We run 50 game-rounds each for the set-cover strategy with partial seekers for the following cases : 100% seekers, 80% seekers, 50% seekers and 20%seekers. For both strategies, 50 game rounds were simulated for the three type of obstacle movement: *hider-friendly*, *seeker-friendly* and *random*.

5.1 Results

5.1.1 Set-Cover Strategy

The set-cover strategy demands **864** seeker agents for MAP-1, while **3792** seekers for MAP-2 on average, to execute set-cover strategy. We analyse the result of the strategy by providing complete and partial seekers in the environment. We run 50 rounds of the game for both the maps with random obstacle movement and following one of the conditions:

All seekers available: If the game is simulated with the demanded number of seekers, the game ends the very next step after it starts. It takes one game step for the hiders to choose a strategic point to hide, while it takes the seekers one game step to capture the hider in the respective strategic point in their field of vision. The game ends with *seekers victory* in two game steps, in every round of game for both the maps.

Partial seekers available: The game is played with less than the required number of seekers demanded by the set-cover strategy. The median game completion steps for both the maps can be seen in Figure 5 and Figure 6.

From the median game completion time, we can conclude that the set-cover strategy works well for large environments even with partial seekers available. Even with only 20% of the seekers available, the seekers take around **432** steps (median across 50 game-rounds) to capture 300 hiders in an environment with 500 moving obstacles and 2000 strategic points in MAP-1. In the case of MAP-2, we observe that the seekers eliminate 2500 hiders from an environment with 5000 obstacles and 20000 strategic points in **2297** game-steps (median across 50 game-rounds).

5.1.2 Sweep Strategy

To execute sweep strategy in map 1 and 2, the seekers demand $\sqrt{2}$a i.e, 71 and 142 initial seekers respectively to diagonally traverse from one corner of the environment to another. The sweep strategy also requires additional seekers to mark obstacles *safe* during traversal. Since each obstacle has four strategic points associated with it, it would take three additional seekers to mark each obstacle safe [see section 4.3.1]. The number of additional seekers required depends on the number of obstacles simultaneously falling under the filed of vision of the traversing seekers. The average number of seekers required in map 1 and 2, across 50 game rounds and random obstacle movement is **273** and **577** respectively.

The game completion steps for sweep strategy across 50 game rounds with random obstacle movement in map 1 and 2 is given in Table 4 and Table 5. Sweep strategy is observed to perform well even in large environments considering lower number of seeker requirements as compared to set-cover strategy.

5.2 Comparison of Strategies

We compare both strategies in map 1 and 2 with different obstacle movement techniques. The game completion time for each map is listed in Table 4 and Table 5.

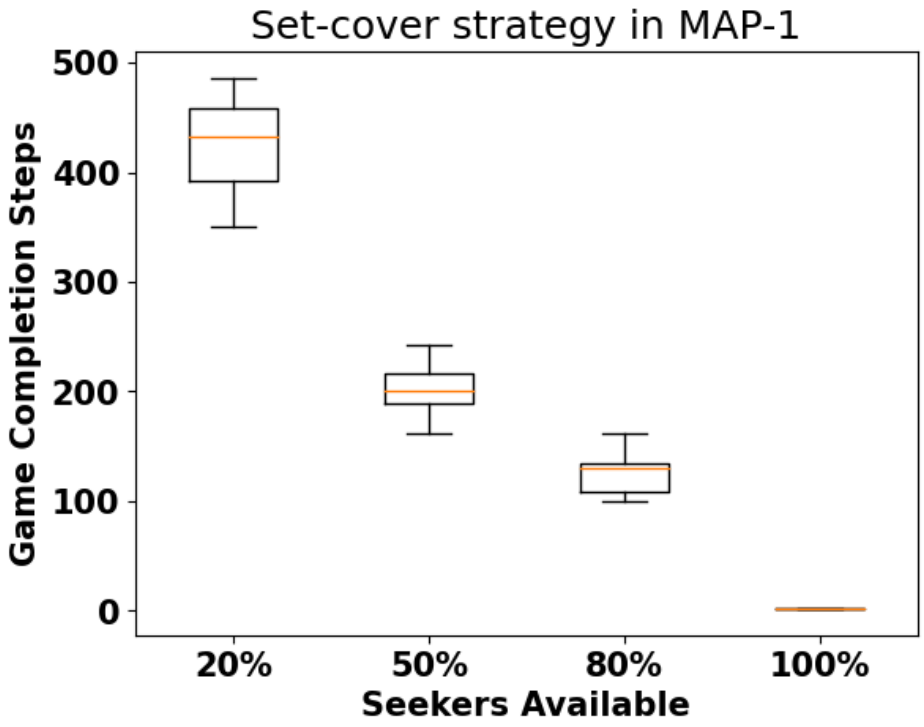

Figure 5: Game Completion time for Set Cover strategy in MAP-1
Game Completion time for Set Cover strategy in MAP-1

Random Obstacle movement: While the set-cover strategy takes less game steps to eliminate hiders from the environment even with partial seekers available, it is to be noted that the number of seekers demanded by set-cover strategy is larger than sweep-strategy. So, sweep strategy is more efficient in case of random obstacle movement in any given environment.

Hider-friendly Obstacle movement: The seekers take more game steps to eliminate all hiders in case of hider-friendly obstacle movement, as expected. However, sweep strategy takes significantly less steps than set-cover strategy in both the environment to capture and eliminate all hiders. This is because the obstacles cannot escape to hider-friendly cells, since the traversal of seekers in sweep strategy ensures that no gap nodes are present.

Seeker-friendly Obstacle movement: The seekers capture hiders in significantly less game steps in any given round of the game. While set-cover strategy works well in a smaller environment like MAP-1, the sweep strategy outperforms in case of larger environment, as in MAP-2. Considering the lower number of seeker agents demand with almost similar game completion steps as set-cover strategy, sweep strategy would be an ideal strategy to follow in case of large environments.

The set-cover strategy takes almost four times of game completion steps in case of hider-friendly obstacle movement as compared to seeker-friendly obstacle movement in in both the maps. The sweep strategy however takes twice the number of game completion steps in case of hider-friendly obstacle movement with respect to seeker-friendly obstacle movement. The sweep strategy can be followed in large environments. It maintains the overall balance of less number of seekers and lower game completion steps to be considered an ideal strategy in most environments. The set-cover strategy however promises much lower game completion times in case of seeker-friendly obstacle movement. This is because more number of seekers can track multiple strategic points when the obstacles move towards the seekers field of vision. While the major drawback of set-cover strategy can be seen as high demand for number of seekers, it is an efficient strategy to be followed in cased where we need to eliminate hiders faster without any constraint on number of seekers. Another reason for set cover strategy to not perform very well in case of seeker-friendly obstacle movement is due to the large number of obstacles. Since half of the grid cells are occupied with obstacles, the strategic points have less scope to move away from the field of vision of seekers. Set-cover strategy performs worse than sweep strategy with hider-friendly obstacle movement in case of a sparse environment where the number of obstacles are significantly less

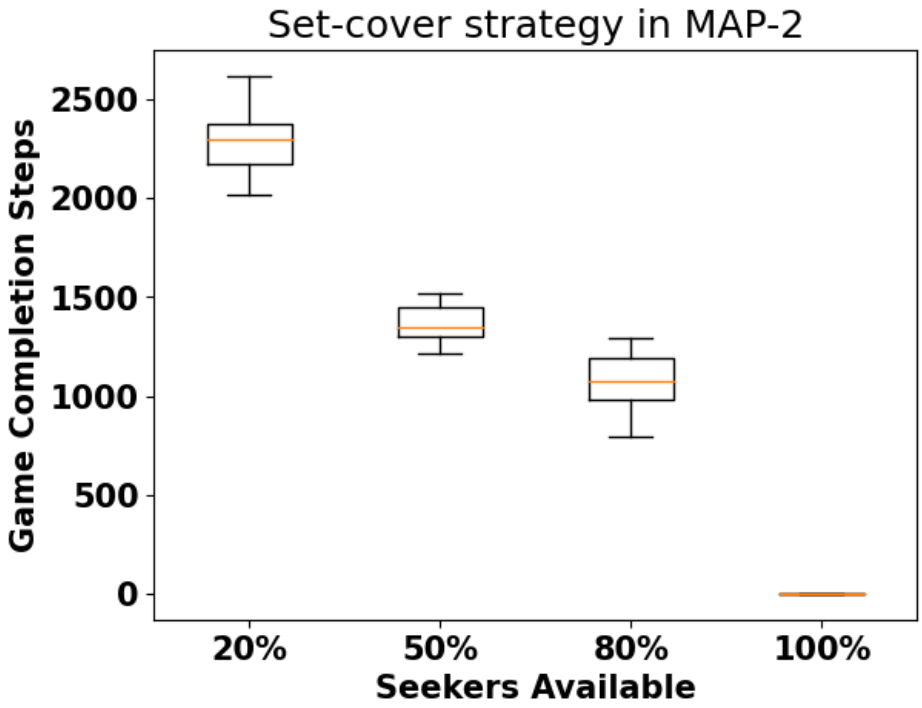

Figure 6: Game Completion steps for Set Cover strategy in MAP-2
Game Completion steps for Set Cover strategy in MAP-2

Obstacle Movement	**Set-Cover (864)**			**Sweep (273)**
	20%	**50%**	**80%**	
Random	432	200	130	324
Hider Friendly	757.97	740.34	566.86	459.55
Seeker Friendly	291.29	184.63	106.59	279.69

Table 4: Median Game Completion Steps in MAP-1. The average number of seekers required by each strategy is given in brackets alongside.

than the number of cells in the grid. The obstacles would have a large scope to move away from the visibility region of the seekers, which would result in recalculating the position of coverage points more often.

6 Related Work

The hide-and-seek game has been an interesting and well known field of study since long. Beside being a game, hide-and-seek problem has found application in other domains like search theory in operation research, pursuit-evasion games with multiple mobile agents (Chung et al. [Chung et al.(2011)], Parsons [Parsons(1978)], Megiddo and Hakimi

Obstacle Movement	**Set-Cover (3792)**			**Sweep (577)**
	20%	**50%**	**80%**	
Random	2297	1347.5	1071	746.5
Hider Friendly	3234.75	1921	1515.3	1023.71
Seeker Friendly	1369.26	884.741	674.11	618.08

Table 5: Median Game Completion Steps in MAP-2. The average number of seekers required by each strategy is given in brackets alongside.

[Megiddo and Hakimi(1978)]), multi-agent security (Pita et al. [Pita et al.(2008)], Fang et al. [Fang et al.(2015)]), Paruchuri et al. [Paruchuri et al.(2008)], Paruchuri et al. [Paruchuri et al.(2007)]), multi-agent deep reinforcement learning (Baker et al. [Baker et al.(2019)]), etc. The princess and monster game by Isaacs [Isaacs(1965)] introduced the hide and seek game with moving hiders. The problem was further studied by Alpern [Alpern et al.(2008)] with changing parameters and was finally solved for a generalised case by Alpern and Gal [Alpern and Gal(2006)]. Most recent work on hide and seek focus on emergent agent behaviour with the help of reinforcement learning, and AI-techniques. Strickland [Strickland(2019)] proposed reinforcement learning based strategies in a 3D environment where agents learned to capture hiders in challenging environment with moving obstacles and mobile agents. This is in contrast to our strategies which works on a 2D environment, based on heuristic approach and works on predictive traversal path planning. Tandon and Karlapalem [Tandon and Karlapalem(2018)] studied the hide and seek game in a 2D environment with static obstacles. They introduced abstractions like coverage and strategic points to encapsulate environment properties for their proposed strategy. Our work uses those abstractions and proposes some more to extend the work to a dynamic environment where obstacles are moving.

7 Conclusion

The *set-cover* and *sweep* strategies presented in the paper perform well and can be followed in a real-life simulation of agents in a given environment. The simulation and comparison of strategies in section 5 of the paper gives insight on the performance of individual strategies in various environments and scenarios. The strategies capture hiders in a reasonable number of steps with different types of obstacle movement. The strategies follow classical set-cover and traversal-based algorithms, which is less compute intensive and ideal for real-time strategic games, as compared to most of the present work, which provide reinforcement learning-based strategies, which demands large training data for the algorithm to *learn* strategies.

While the set-cover and sweep strategy works well in the simulated environment, there are areas where it can be improved and explored. We want to evaluate the performance of the strategies with larger obstacles with more strategic points. The hiders are currently assumed to occupy a strategic point before they can be caught by the seekers. The strategies, however, does not take into account the possibility of hiders staying hidden in the cell without being associated with a strategic point. This would create a stagnant condition where the seekers would keep searching for hiders in the strategic points while the hiders are hidden in some other non-strategic point in the environment.

References

[Akhter(2015)] Fatema Akhter. 2015. A Heuristic Approach for Minimum Set Cover Problem. *International Journal of Advanced Research in Artificial Intelligence* 4 (06 2015). https://doi.org/10.14569/IJARAI.2015.040607

[Alpern et al.(2008)] Steve Alpern, Vic Baston, and Shmuel Gal. 2008. Network search games with immobile hider, without a designated searcher starting point. *International Journal of Game Theory* 37, 2 (01 Jun 2008), 281–302. https://doi.org/10.1007/s00182-008-0116-7

[Alpern and Gal(2006)] Steve Alpern and Shmuel Gal. 2006. *The theory of search games and rendezvous*. Springer. https://www.springer.com/fr/book/9780792374688

[Baker et al.(2019)] Bowen Baker, Ingmar Kanitscheider, Todor Markov, Yi Wu, Glenn Powell, Bob McGrew, and Igor Mordatch. 2019. Emergent Tool Use From Multi-Agent Autocurricula. https://doi.org/10.48550/ARXIV.1909.07528

[Chung et al.(2011)] Timothy H. Chung, Geoffrey A. Hollinger, and Volkan Isler. 2011. Search and pursuit-evasion in mobile robotics. *Autonomous Robots* 31, 4 (20 Jul 2011), 299. https://doi.org/10.1007/s10514-011-9241-4

[Fang et al.(2015)] Fei Fang, Peter Stone, and Milind Tambe. 2015. When Security Games Go Green: Designing Defender Strategies to Prevent Poaching and Illegal Fishing. In *Proceedings of the 24th International Conference on Artificial Intelligence* (Buenos Aires, Argentina) *(IJCAI'15)*. AAAI Press, 2589–2595.

[Isaacs(1965)] Rufus Isaacs. 1965. *Differential games; a mathematical theory with applications to warfare and pursuit, control and optimization [by] Rufus Isaacs*. Wiley New York. xxii, 384 p. pages.

[Megiddo and Hakimi(1978)] Nimrod Megiddo and S.L. Hakimi. 1978. *Pursuing Mobile Hiders in a Graph*. Discussion Papers 360. Northwestern University, Center for Mathematical Studies in Economics and Management Science. https://ideas.repec.org/p/nwu/cmsems/360.html

[Parsons(1978)] T. D. Parsons. 1978. Pursuit-evasion in a graph. In *Theory and Applications of Graphs*, Yousef Alavi and Don R. Lick (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 426–441.

[Paruchuri et al.(2008)] Praveen Paruchuri, Jonathan Pearce, Janusz Marecki, Milind Tambe, Fernando Ordóñez, and Sarit Kraus. 2008. Playing games for security: An efficient exact algorithm for solving Bayesian Stackelberg games. *AMAAS-2008 Conf*, 895–902. https://doi.org/10.1145/1402298.1402348

[Paruchuri et al.(2007)] 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* (Honolulu, Hawaii) *(AAMAS '07)*. Association for Computing Machinery, New York, NY, USA, Article 181, 8 pages. https://doi.org/10.1145/1329125.1329344

[Pita et al.(2008)] 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. *AAMAS*, 125–132. https://doi.org/10.1145/1402795.1402819

[Strickland(2019)] Eliza Strickland. 2019. AI agents play hide-and-seek: An OpenAI project demonstrated "emergent behavior" by AI players - [News]. *IEEE Spectrum* 56 (11 2019), 6–7. https://doi.org/10.1109/MSPEC.2019.8889898

[Tandon and Karlapalem(2018)] Akshat Tandon and Kamalakar Karlapalem. 2018. Medusa: Towards Simulating a Multi-Agent Hide-and-Seek Game. In *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI-18*. International Joint Conferences on Artificial Intelligence Organization, 5871–5873. https://doi.org/10.24963/ijcai.2018/866

[Tandon and Karlapalem(2020)] Akshat Tandon and Kamalakar Karlapalem. 2020. Capturing Oracle Guided Hiders. In *Proceedings of the 19th International Conference on Autonomous Agents and MultiAgent Systems* (Auckland, New Zealand) *(AAMAS '20)*. International Foundation for Autonomous Agents and Multiagent Systems, Richland, SC, 1350–1358.