

# Optimal Policy in Discrete Pursuit-Evasion Games

Marcos A. M. Vieira and Ramesh Govindan and Gaurav S. Sukhatme

*Abstract—*

**In a discrete Pursuit-Evasion game, we describe an algorithm to calculate the optimal policy that pursuers should use in order to capture evaders with the minimum number of steps. We assume that all players have full knowledge and that evaders also play an optimal strategy. We illustrate how convergence time varies by topology. We have implemented this algorithm, and present results on a physical multi-robot testbed.**

## I. INTRODUCTION

We are motivated by practical problems in security and monitoring for large, structured, spaces (e.g., to ensure the integrity of a large building or complex). The problem we focus on is pursuit-evasion wherein robots must pursue and catch evaders.

In Pursuit-Evasion Games (PEGs), multiple robots (the pursuers) collectively determine the location of one or more evaders, and try to corral them. The game terminates when every evader has been corralled by one or more robots. Several versions of the problem exist. In certain frameworks, it is acceptable to merely “sight” an evader for it to be “located”, in others, a precise coordinate must be reported. Other formulations insist on a certain speed of convergence with fewer constraints on accuracy. Finally, formulations vary depending on whether the multi-robot control algorithm is required to have provably correct behavior, whether the number of evaders is known a priori, and whether they are malicious or benign. Each variation of the problem brings with it a different set of challenges, and several of these variations have been solved to varying degrees.

We focus on PEGs in bounded, spatially complex environments similar to today’s office environments. In such environments, we can assume imperfect geometric regularity (e.g., the presence of corridors and 90-degree turns, but possibly a regular placement of doorways or elevator exits). However, it is increasingly true that such environments are well provisioned with wireless communication capability, and that many such environments will likely have dense embedded sensing (for surveillance or environmental control). This environmental-embedded sensing can provide, for the players, full knowledge of the game.

We consider the class of PEGs played on a discrete graph. Specifically, we use a topological map of the environment, whose nodes correspond to coarse-grained regions and whose links connect neighboring regions [1], [2]. Discrete graph based games are acceptable for many uses of pursuit-evasion (e.g., surveillance, finding survivors).

In this paper, we consider a version of the game in which  $M$  pursuers collectively attempt to capture  $N$  evaders (Section II). We are interested in the convergence time of the game (i.e., the minimum number of steps for the pursuers to capture the evaders). To the best of our knowledge (Section VI), the first provably minimal convergence time algorithm for multiple pursuers and evaders is presented (Section III). In our formulation, all players have complete knowledge of the state of others, and the evaders also pursue an optimal evasion strategy. Clearly, convergence time depends on the topology as well as on the number of pursuers and evaders: we explore these dependencies for a few canonical topologies (Section IV). Finally, we present results from running an implementation of our algorithm on a physical robot testbed (Section V). The experiments confirm the feasibility of our algorithm, even on a worst-case initial configuration.

## II. GAME DEFINITION AND ASSUMPTIONS

Let  $G = (V, L)$  be a finite connected undirected graph with  $V$  vertices and  $L$  lines or edges. There are two teams of players called pursuers  $P$  and evaders  $E$ . Initially,  $P$  and  $E$  occupy some vertices of  $G$ . In describing the algorithm, we assume that time is discrete and increments at steps of 1. At each time step, all pursuers and evaders are given the poses of all participants. Both teams play a game on  $G$  according to the following rule. At each step, each pursuer chooses a neighboring vertex of  $G$  to move to, then the evaders do the same. They then move to the corresponding vertex in  $G$ , as defined in [3], and repeat the previous step. An evader is captured if at a time  $t$ , the pursuer  $P$  is on the same vertex as the evader being captured. The team of pursuers  $P$  wins if it captures all evaders. If an evader can avoid capture indefinitely, then the evader team wins the game.

We can now define our game more formally as follows:

**Input** Coarse estimated poses of  $N$  robots and  $M$  evaders in a bounded environment  $E$ .

**Output** Motion commands for  $N$  robots.

**Goal** Minimize the capture time of the evaders.

**Restriction** No motion model of the evaders is available to pursuers.

Before describing the optimal strategy for this game, we discuss some assumptions and describe our notation. We assume the pursuers and the evaders have full knowledge. This is possible in an indoor scenario with an environment-embedded wireless communication network that also provides a sensing capability. The communication network provides a backbone for sharing game state, and the sensing

capability detects evader positions, thereby giving full visibility to the pursuers. We assume that the evaders also play an optimal strategy: they too have, at each time instant, the locations of all the pursuers, and decide their moves with a view to avoiding capture for the longest possible time. Finally, we also assume that pursuers and evaders move at the same speed (more precisely, we assume that they move exactly one hop in the topology at each time step).

Let  $p=|P|$ ,  $e=|E|$ ,  $v=|V|$ . Let  $P_i$  be the position of the  $i^{th}$  pursuer and  $E_i$  be the position of the  $i^{th}$  evader.

The tuple  $a = \langle P_0, \dots, P_p, E_0, \dots, E_e \rangle$  represents the current position of all participants. We define a boolean variable  $T(turn)$  to denote if it is the pursuers' turn to move or not (recall that, in our algorithm, pursuers and evaders alternate at each time step<sup>1</sup> We say that the tuple  $\langle a, T \rangle$  encodes the state  $s$  of a game.

In general, a game can have  $2 * v^{p+e}$  states, because each pursuer and evader can be at one of  $v$  positions, and for each configuration of pursuers and evaders, there are 2 turns (evader and pursuer moves). Each game can be represented by a sequence of transitions through this state space. Each pursuer and evader executes a deterministic algorithm (called its *policy*) for determining, given the current state, what the next move should be. We call  $\rho$  the pursuer policy and  $\epsilon$  the evader policy. Since we consider deterministic policies, if in a particular game, a state is repeated, the game will not terminate and the evaders win.

The game terminates when a *capture state* is reached. In a capture state, at least one pursuer occupies each vertex in which an evader resides. There exists a different definition of termination. If, during the evolution of the game, a pursuer reaches an evader's position, the evader exits the game. It is easy to see that our definition results in a game that is strictly harder than this variant. When evaders exit the game, the remaining pursuers can always, and more quickly, capture the remaining set of evaders. Indeed, there exist cases where our game might not terminate (a 2-pursuer 2-evader game, in some topologies) but this variant will.

### III. THE OPTIMAL POLICIES FOR PURSUIT-EVASION

Pursuit-Evasion is a zero-sum game since pursuer's gain or loss is exactly balanced by the losses or gains of the evader. The evader's goal is to escape as long as possible whereas the pursuers have to capture the evaders as fast as possible.

Zero-sum games have been extensively studied in the game theory literature, and our solution models a PEG as a zero-sum game that uses the minimax algorithm [4]. This algorithm minimizes the maximum possible loss for each player in the game.

To describe this algorithm, consider first that the evolution of any PEG can be represented by a game graph, a directed graph with possible cycles. The start state (as defined by the starting configuration of the pursuers and evaders) has a directed edge from itself to all possible next states that

the pursuers can make from the start state. (In our game, we assume that pursuers and evaders move alternatively). In turn, from each of these states, there is a directed edge to all possible next states resulting from evaders moves from that state. The graph can thus be recursively defined. In general, a game is a traversal on this graph. If this traversal ends in a capture state, the pursuers win the game. However, it is also possible for the traversal to repeat states: such a traversal will result in a non-terminating game and the evaders win.

Suppose now, that in the game graph, we assign to each state  $S$ , a cost function  $C$ .

- When an evader moves,  $C(s)$  denotes the maximum distance from state  $s$  to a capture state, and
- When a pursuer moves,  $C(s)$  represents the minimum distance from state  $s$  to a capture state.

In our game, we consider the following policies:

- The pursuers' policy  $\rho$  is: choose that neighboring state in the game graph which has the smallest  $C(s)$ . Intuitively, this moves the game at each step as close as possible to a capture state.
- The evaders' policy  $\epsilon$  is: choose that neighboring state in the game graph which has the largest  $C(s)$ . Intuitively, this moves the game at each step as far away as possible from a capture state. Thus, the evader is truly adversarial.

In what follows, we first show how to compute the game graph efficiently. Then, we prove that these policies are optimal from the pursuer's perspective: if the game terminates, they reach a capture state in the shortest possible number of moves.

First, we construct the game graph by generating all states and all possible transitions between states. Initially, all states have cost infinity, except the initial set of capture states which have cost 0 (Algorithm 1).

Let us define  $F_0$  as a set of capture states, in other words, the states where at least one pursuer occupies each vertex in which an evader resides. These states have cost value 0 since no move is necessary for the termination of the game. Now define  $F_1$  as the set of states which can reach one of the capture states in one pursuer move.  $F_1$  consists of both states where it is the pursuer's turn to move, and states where it is the evader's turn to move. A state where it is the evader's turn to move belongs to  $F_1$  where irrespective of the next move made by the evader, the pursuer can still reach the capture state in a single step. Similarly, we can define inductively the  $F_{i+1}$  set of states as the states from which the pursuers only needs  $i+1$  steps to terminate the game, irrespective of evader's movements. For any state  $s = (a, 1)$ , when it is the pursuer's turn to move, the cost  $C(s) = \text{minimum}(C(s'))+1$  where  $s'$  is any state that the pursuer can transition to from  $s$ . For any state  $s = (a, 0)$ , when it is the evader's turn to move, the cost  $C(s) = \text{maximum} C(s')$ , where  $s'$  is any state that the evader can transition to from  $s$ .

For a pursuer, the goal is to reach capture state, i.e. minimize  $C$ . For an evader, if  $C(s)$  is finite, where  $s$  is the current state, then irrespective of what the evader does,

<sup>1</sup>This assumption enables us to analyze our algorithm. Of course, in real world experiments, it is difficult to ensure this synchrony.

it will be captured within  $C(s)$  pursuer's move. The evader should then choose a transition  $a$  to  $a'$  which maximizes the time of capture. If any state has a cost function value infinity, it means the number of pursuers is not sufficient to capture the evaders. It is necessary to add more pursuers to the game.

---

**Algorithm 1** Algorithm for computing the game graph.

---

```

{Initialization}
Generate all states
Generate all possible transitions

for all state  $s$  do
  if  $s$  is a capture state then
    add  $s$  to  $F_0$ 
     $C(s) \leftarrow 0$  {cost function}
  else
     $C(s) \leftarrow \infty$ 
  end if
end for
 $i \leftarrow 0$ 
repeat
   $i \leftarrow i + 1$ 
   $change \leftarrow false$ 
   $U \leftarrow$  set of all unmarked states that have a transition
  to a marked state.
  for all  $s$  in  $U$  do
    if  $s$  is a pursuer move then
      if  $s$  has at least one transition to a marked state
      then
        add  $s$  to  $F_i$  {mark  $s$ }
         $C(s) \leftarrow \min$  (over all marked neighbors)+1
        {count this move}
        add transition to  $\rho$ 
         $change \leftarrow true$ 
      end if
    else
      {evader move}
      if all transition from  $s$  reach a marked state then
        add  $s$  to  $F_i$  {mark  $s$ }
         $C(s) \leftarrow \max$  (over all marked neighbors)
        add transition to  $\epsilon$ 
         $change \leftarrow true$ 
      end if
    end if
  end for
until not  $change$ 

```

---

Algorithm 1 presents the optimal policy for evaders and pursuers that we describe above. The algorithm loops while we add more states to  $F_i$ . Eventually, when there are no more states to be added, the algorithm terminates. Since each state can only be added once, the algorithm terminates.

*Theorem 3.1:* Algorithm 1 provides the optimal policy  $\rho$  for pursuers to play the game in a graph  $G$ .

*Proof:* We prove this by induction. The induction hypothesis is if state  $s$  belongs to  $F_i$ , then the game will

terminate in at most  $C(s) = i$  steps independent of evader movements. If  $i = 0$ , by definition of  $F_i$ , it is a capture state, and the game is over. Suppose the claims holds for  $i$  and let us prove it for  $i + 1$ . Let state  $s = (a, 1) \in F_{i+1}$ . By definition of  $F_{i+1}$ , there exists transition  $(s, (a', 0))$  with  $(a', 0) \in F_i$ . Let the pursuer move. The cost will be minimum of  $C(a') + 1$ . Suppose the evader can escape on arrangement  $a'$ . But then, the evader would have chosen a transition with cost infinity, and  $(a', 0)$  would not have been in  $F_i$ . By definition of  $F_i$ , there exists transition  $((a', 0), (a'', 1))$  and  $(a'', 1)$  is a captured state with  $C(a'') = i$ . Hence, if a state  $s$  belongs to  $F_{i+1}$ , the cost  $C(s)$  will be 1 to get to  $(a'', 1) + C(a'')$ , which terminates in  $i$  steps by the induction hypothesis. ■

For a graph with  $v$  vertices, the time complexity and space complexity is  $O(v^{p+e})$ .

#### IV. RESULTS

Table I shows some instances of games and their properties. The first and second column are the topology name and its graphical representation respectively. The third column is the necessary number of pursuers to guarantee the termination of the game. The fourth and fifth columns represent the maximum number of steps to terminate the game, given the number of pursuers. It is interesting to see that even though the torus topology needs at least 3 pursuers, if we play the game with 3 pursuers, the game will be shorter than in a 4x4 grid. Another interesting fact is increasing the number of edges from grid 2d 4x4 to cylinder 4x4 does not increase the necessary number of pursuers but it decreases the number of steps to terminate the game. Increasing the edges again (from cylinder to torus), increases the necessary number of pursuers.

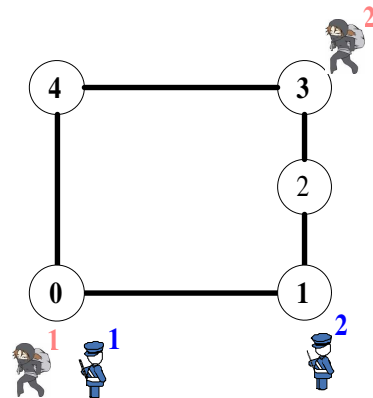


Fig. 1. Ring Topology

Figure 1 shows a ring topology with 5 nodes. A game with 2 pursuers and 2 evaders will not terminate in this topology, since we defined the game to end only when all the evaders are captured. To illustrate this, suppose pursuer 1 stayed on the same location as evader 1. Pursuer 2 needs to capture evader 2, but evader 2 just needs to stay away from pursuer 2.

Topology Name	Graphical	C(G)	2 pursuers	3 pursuers
Grid 2D 4x4		2	6	6
Cylinder Grid 4x4		2	5	5
Torus Grid 4x4		3	$\infty$	4

TABLE I  
INSTANCES OF GAMES AND THEIR PROPERTIES

## V. EVALUATION

In this section, we describe the results on a physical robot testbed. To validate that our scheme is implementable, we played a 4 pursuer, 2 evader game. We analyzed all possible such games and chose a worst-case initial configuration (there can be more than one initial configuration with the same convergence time).

Our robot platform consists of an iRobot Create and a small embedded computer mounted on top of it (Figure 2). The Create, a differential drive robot, has a round chassis of 33 cm diameter. The embedded computer, the Ebox 3854, is an 800MHz embedded PC with 256MB shared DDR memory, and supports compact flash sockets. The embedded

computer runs Linux Fedora Core 6 as the operating system. For sensing and control, we developed a Create driver for Player [5]. The nominal speed is 0.2 m/s.

The robots use the network shown in Figure 3. This network is deployed above the false ceiling on one floor of a large office building and consists of two tiers: an upper tier containing 9 Stargates (embedded computers running Linux), and a lower-tier containing 56 tmoteSky nodes (tiny commercial sensor nodes). The sensor nodes run an 8MHz Texas Instruments MSP430 microcontroller, have 10KB RAM and a 2.4 GHz IEEE 802.15.4 Chipcon Wireless Transceiver with a nominal bit rate of 250 Kbps.

The sensor nodes provide the capability of a virtual



Fig. 2. The robot platform - an iRobot Create with an Xbox

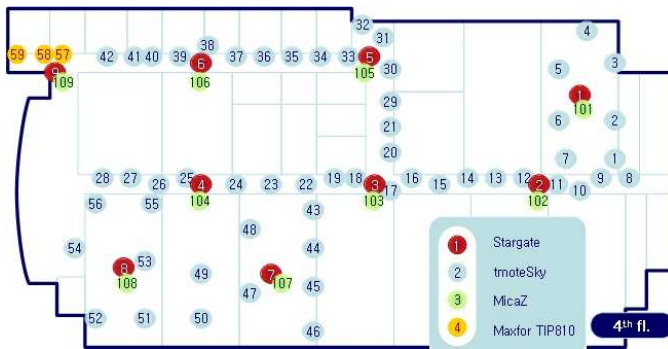


Fig. 3. Layout of the network testbed

position sensor, and can sense the position of pursuers and evaders using radio signal strength indicator (RSSI).

We executed the algorithm discussed in this paper on a computer and generate a policy file for pursuers and evaders. This file (available to all robots) contains the next state to go given the current state. We code a state as a number in base  $v$ . For example, suppose in a 3 pursuer, 1 evader game with 3 topological nodes, the evader is at location 1 and pursuers are located at 2,2 and 0 respectively. The sequence 1220 in base 3 is 51. Thus, the current state is represented by number 51. For a 4 pursuer, 2 evader game on 9 nodes, the file size is about 240 Kb. Each robot has an id, which gives a priority. Robots with high priority have their position code first. In this way, robots can coordinate among themselves without the need to communicate directly with each other. Robots receive position estimation from the network, check their policy file and get the next state number to go. With the id and the next state number information, each robot determines what is the next position to go by decoding it. To decode, each robot considers the next state number as a number in base  $v$ , and extracts the  $i^{th}$  digit, where  $i$  represents its priority.

We play the games on the floor plan shown in Figure 3, using the network whose nodes are shown in that figure. We use a topological map of the environment consisting of 9 nodes, whose nodes correspond to coarse-grained regions and whose links connect neighboring regions, as shown in Figure 4.

Figure 4 shows the trajectory of each pursuer during one instance of a game. The nodes and solid undirected edges connecting them represent the topological map. The

solid directed lines (lines with arrows) show the pursuer path as determined by the localization system. The dashed lines illustrate the evader path. The edge labels represent the time sequence of the robot. The pursuer and evader's initial positions are indicated by the corresponding icons.

Our robots can only follow walls in one direction (and must cross the corridor to reverse direction) and therefore must correct their orientation a posteriori. For example, it is not possible for the robots to go from position 2 to position 1 directly because the robots follow the walls to their right.

To make it clear, we give a narrative of the game. All pursuers start at position 4 and all evaders start at position 0. Pursuer 3 is the first to move (it tries to corral by going around). The other pursuer do not move because they know pursuer 3 has to go around so they can corral the evaders. At time 1, pursuer 3 is at location 3. At time 2, pursuer 1 tries to capture evaders by moving towards them. Pursuer 3 moves to location 2. At time 3, pursuer 2 tries to capture evaders by moving towards them and move to location 3. Pursuer 1 and 3 keep moving and are at location 2 and 5 respectively. At time 4, pursuer 4 just once moves closer to evaders, it is not really necessary in the game. Pursuers are at 5,2,6,3. At location 5, pursuers 1 and 2 switch their orientation (turn around to another wall). At time 5, pursuers are at 2,5,6,3. At time step 6, pursuers are at 1,5,6,3 and evaders at location 0. If no evader moves, pursuer 1 would captured them. So, at time step 7, evader 2 tries to escape by going to location 1. Evader 2 knew evader 1 was not captured, thus it tried to escaped even though it went to same location as pursuer 1. Evader 1 did not move to location 8 because it would be captured by pursuer 3 anyways. Finally, at time 8, pursuer 1 captured evader 1 at location 0 and pursuer 2 captured evader 2 at location 1. Pursuer 2 is at location 8 to guarantee evaders are corral.

In a simulation analysis, the convergence time for the game to end is 6 steps. Our results took 8 steps because our robots can only follow walls in one direction and we did not take this constraint into our analysis. Indeed, if you take into account that to move from node 2 to node 1, you need to go to node 5 and node 2 again, our results match our analysis. Hence, the experiments confirm the feasibility of our algorithm.

## VI. RELATED WORK

To situate our work in the existing literature, we classify the type of PEGs using seven criteria: the *ratio* of the number of pursuers to the number of evaders; whether pursuers and evaders have full and/or global *visibility*, or whether they can only see within a threshold distance or until occluded by an obstacle (usually modelled by the edge of a polygon in 2D); what additional *information* robots have with respect to the opponents' strategy or planning algorithm; whether the *environment* is modelled as a graph (discrete) or a polygon (continuous half-space with lines in 2D as boundaries); how the evader is *captured*, whether by being surrounded, seen or sensed by the pursuer, or approached within a certain distance, or physically contacted; the relative *speed* between

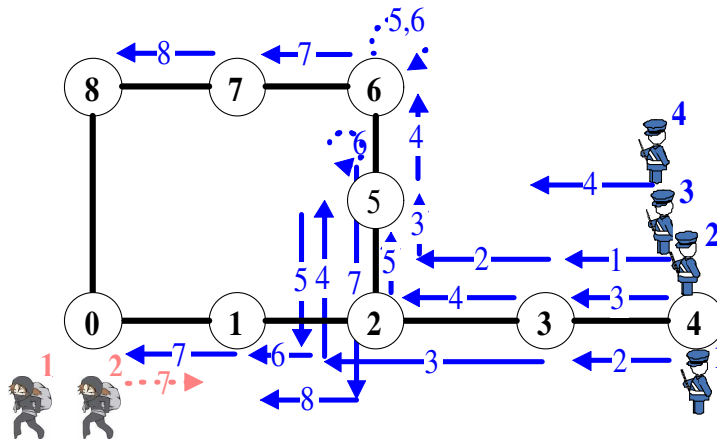


Fig. 4. A 4 pursuer, 2 evader game

the pursuer and evader; and, how much *uncertainty* in sensing, actuation and communication is injected into the game.

Our work assumes the pursuers and evaders have full visibility; they can know about others' strategy; the environment is modeled as a graph; the evaders are captured only if all of them share a vertex with some pursuers; the evader speed is the same as the pursuer; and we did not consider uncertainty.

Other work has explored theoretical bounds on eventual capture [6], [7], or pursuit-evasion under constrained geometries [8], [9], [10], [11], or has examined sophisticated control strategies [12], [13].

In [14], an algorithm to determine if  $K$  pursuers are sufficient to capture an evader is presented. The minimum necessary pursuers to capture an evader is called the the cop number  $c(G)$ . They also shown that every graph is topologically equivalent to a graph with pursuer number at most two. Pursuit-Evasion Games are also called Cops and Robbers by mathematicians.

In the survey presented by Alspach [15], a number of references on the necessary number of pursuers for a given graph class can be found. Aigner and Fromme [6] proved that in a planar graph  $G$ , 3 pursuers are sufficient for the pursuers to win the game. Quilliot [16] extended this result, giving an upper bound to the number of pursuers depending on the genus of the graph  $G$ . In [17], the necessary number of pursuers is studied under three graph product operations.

To the best of our knowledge, we are the first to present an algorithm to minimize the time to capture of all evaders.

## VII. CONCLUSIONS

We presented an optimal algorithm (causing pursuers to take the minimum number of steps to win a Pursuit-Evasion game in a discrete graph). We illustrate how convergence time varies with different topologies. We have validated the feasibility of our algorithm by experimentally playing mobile robot-based pursuit evasion games on a physical testbed.

## REFERENCES

- [1] B. Kuipers and Y.-T. Byun, "A robot exploration and mapping strategy based on a semantic hierarchy of spatial representations, Tech. Rep. AI90-120, 1, 1990. [Online]. Available: [citeseer.ist.psu.edu/kuipers91robot.html](http://citeseer.ist.psu.edu/kuipers91robot.html)
- [2] M. J. Mataric, "Integration of representation into goal-driven behavior-based robots," *IEEE Transactions on Robotics and Automation*, vol. 8, no. 3, pp. 304–312, 1992.
- [3] R. J. Nowakowski and P. Winkler, "Vertex-to-vertex pursuit in a graph," *Discrete Mathematics*, vol. 43, no. 2-3, pp. 235–239, 1983.
- [4] S. J. Russell and P. Norvig, *Artificial Intelligence: A Modern Approach*. Pearson Education, 2003. [Online]. Available: <http://portal.acm.org/citation.cfm?id=773294>
- [5] B. Gerkey, R. Vaughan, and A. Howard, "The player/stage project: Tools for multi-robot and distributed sensor systems," in *11th Int. Conf. on Advanced Robotics (ICAR 2003)*, June 2003, <http://playerstage.sourceforge.net>.
- [6] F. M. Aigner, M. "A game of cops and robber," Tech. Rep., 1984.
- [7] J. Sgall, "Solution of David Gale's lion and man problem," *Theor. Comput. Sci.*, vol. 259, no. 1-2, pp. 663–670, 2001.
- [8] L. J. Guibas, J.-C. Latombe, S. M. LaValle, D. Lin, and R. Motwani, "Visibility-based pursuit-evasion in a polygonal environment," in *WADS '97: Proceedings of the 5th International Workshop on Algorithms and Data Structures*. London, UK: Springer-Verlag, 1997, pp. 17–30.
- [9] R. Murrieta-Cid, T. Muppilala, A. Sarmiento, S. Bhattacharya, and S. Hutchinson, "Surveillance strategies for a pursuer with finite sensor range," *Int. J. Rob. Res.*, vol. 26, no. 3, pp. 233–253, 2007.
- [10] S. Bhattacharya, S. Candido, and S. Hutchinson, "Motion strategies for surveillance," in *Robotics: Science and Systems*, 2007.
- [11] W. Cheung, "Constrained pursuit-evasion problems in the plane," *Master Thesis, U.British Columbia*, 2005.
- [12] R. Vidal, O. Shakernia, H. J. Kim, D. H. Shim, and S. Sastry, "Probabilistic pursuit-evasion games: theory, implementation, and experimental evaluation," *Robotics and Automation, IEEE Transactions on*, vol. 18, no. 5, pp. 662–669, 2002.
- [13] S. Oh, L. Schenato, P. Chen, and S. Sastry, "Tracking and coordination of multiple agents using sensor networks: system design, algorithms and experiments," *Proceedings of the IEEE*, vol. 95, no. 1, pp. 234–254, January 2007. [Online]. Available: <http://www.truststc.org/pubs/244.html>
- [14] A. Berarducci and B. Intrigila, "On the cop number of a graph," *Adv. Appl. Math.*, vol. 14, no. 4, pp. 389–403, 1993.
- [15] B. Alspach, "Searching and sweeping graphs: a brief survey," *Le Matematiche (Catania)*, vol. 59, pp. 5–37, 2004.
- [16] A. Quilliot, "A short note about pursuit games played on a graph with a given genus," *J. Comb. Theory, Ser. B*, vol. 38, no. 1, pp. 89–92, 1985.
- [17] S. Neufeld and R. Nowakowski, "A game of cops and robbers played on products of graphs," *Discrete Math.*, vol. 186, no. 1-3, pp. 253–268, 1998.