Adaptive Behavior manuscript No.
(will be inserted by the editor)

Decentralized Strategy Selection with Learning Automata for Multiple Pursuer-Evader Games

Sidney N. Givigi Jr.[1], **Howard M. Schwartz**[2]

[1] Department of Electrical and Computer Engineering
Royal Military College of Canada
P.O. Box 17000 Station Forces
Kingston, ON, Canada, K7K 7B4
[2] Systems and Computer Engineering, Carleton University
1125 Colonel By Drive
Ottawa, ON, Canada, K1S 5B6

the date of receipt and acceptance should be inserted later

Abstract. The multiple pursuers and evaders game may be represented as a Markov game. Using this modeling, one may interpret each player as a decentralized unit that has to work independently in order to complete a task. This is a distributed multiagent decision problem and several different possible solutions have already been proposed. However, most solutions require some sort of central coordination. In this paper, we intend to model each player as a learning automata (LA) and let them evolve and adapt in order to solve the difficult problem they have at hand. We are also going to show that using the proposed learning process, the players' policies will converge to an equilibrium point. Simulations of such scenarios with multiple pursuers and evaders are presented in order to show the feasibility of the approach.

Key words. Learning, Pursuer-evader games, Intelligent systems, Reinforcement learning, Learning automata

1 Introduction

Game theory, introduced first by von Neumann and Morgenstern (1947), may be defined as the study of decision making (Myerson, 1991) in order to solve conflicts. It is based on the idea that each player is given a utility function of its own strategy and the strategies played by the other players. In the general approach, the game and the strategies are discrete, therefore, matrices with strategies and payoffs may be assembled.

Several other views of game theory have been introduced since its inception, one of them, called *differential games*, by Isaacs (1965). Differential games investigate how decision making takes place over time (Yeung and Petrosyan, 2006) considering continuous domains. To represent this game, one needs to specify the dynamic model of the process under investigation by means of differential or difference equations.

Pursuit-evasion games are very much suitable to be modeled by a differential game. However, when dealing with multiple pursuers and evaders, the modeling itself and the solution for the games become very cumbersome (Starr and Ho, 1969).

In order to avoid this problem, herein we make use of a sort of games known as *Markov games* (van der Wal, 1980). In this class of games, the models of the environment and the interaction among players are considered to be Markov chains (Filar and Vrieze, 1997). Players are able to influence the state transitions in a Markov chain by taking actions (Hespanha and Prandini, 2001).

In all these classes of games, one is usually interested to study if there is some equilibrium points related to the model. The most important type of equilibrium is known as the *Nash* equilibrium and the correspondent *value* of the game (Nash, 1951). However, if the calculation of the Nash equilibrium is difficult, one may be satisfied with some suboptimal solution for the game (Li and Cruz, 2006).

Correspondence to: S. Givigi
e-mail: Sidney.Givigi@rmc.ca

One of the most interesting questions in the field is on how players would adapt and learn to play strategies. This is the widely known problem of *learning* in games, which has been largely studied in the last couple of decades, including includes several books (Fudenberg and Levine, 1998; Weibull, 1995) and recent research papers (Hofbauer and Sigmund, 2003; Conlisk, 1993). Comparatively, fewer papers have dealt with learning in differential (Harmon et al., 1995; Givigi et al., 2009) and Markov games (Littman, 2001).

In this article, we propose that the multiple pursuer-evader game be modeled by a Markov game. Moreover, each decision-making unit (or player) is adaptive. They are known as learning automata. It will be shown that using such an algorithm, the game converges to an equilibrium point.

The paper is divided as follows. We start by defining the multiple-pursuer multiple-evader game in terms of a Markov game in section 2. This section introduces the notation used throughout the work (subsection 2.1), the representation of the states of the system (subsection 2.2), the transition probabilities (subsection 2.3), the policies available to each player (subsection 2.4), the rewards each player gets for each time step (subsect 2.5) and, finally, the general model for Markov pursuit-evasion games with the formalization of the problem to be addressed in the next sections (subsection 2.6). The main contribution of section 2 is in presenting a simple and powerful way to represent pursuit-evasion games. In section 3 we detail the learning algorithm that will be used in the solution of the game. Section 4 states the objective of the learning process, namely, achieving an optimal solution for the problem described in section 2. Section 5 discusses the convergence of the approach used in three different levels: the convergence of the learning procedure to an Ordinary Differential Equation (section 5.2), the convergence of the algorithm at each state to a Nash equilibrium (section 5.3) and finally the convergence of the cost function of the game to the optimal solution (section 5.4. In section 6, simulations of the system are presented in order to show the feasibility of the solution. And, finally, section 7 presents our conclusions from theory and simulations and points to possible future work.

2 Markov multiple-pursuers multiple-evaders games

In this section we are going to follow to some extent the model presented in (Hespanha et al., 2000). However, some noticeable differences must be emphasized. First of all, in their work, Hespanha et al. (2000) consider a game between only two players, wherein one player, the pursuers, is in fact a *team of players*. Therefore, either some sort of sophisticated exchange of information among the pursuers or a central intelligence that controls all of them must be assumed. In our approach, we are going to assume that each player, pursuers and evaders, behave as separate entities. Moreover, we deal in here with a game with "rewards" (see subsection 2.5), whereas Hespanha et al. (2000) deal with *probabilities*. Their cost function is in fact an expression of the probabilities of the pursuers to catch the evader at the next time step. However, this optimization is only achieved if we assume *nested information*, i.e., if one player, in this case the evader, has access to all the information available to the others plus information about its own state that is available just to itself.

Before we go any further, let us first define some needed notation. The notation follows closely the ones provided by (Hespanha et al., 2000) and (Hespanha and Prandini, 2001).

2.1 Notation

Throughout this work we use low cap bold face letters (or greek symbols) for random variables, therefore, $\mathbf{x}(t)$ is going to denote a random variable. Also, given a random variable $\xi \in \mathbb{R}^n$ and some $c \in \mathbb{R}^n$, we denote the probability of the event happening as $\mathbb{P}(\xi = c)$. The same notation is used to represent conditional probabilities. The expected value of ξ conditioned to an event A is denoted by $\mathbb{E}[\xi|A]$.

The symbols $\wedge$ and $\vee$ signify the logical operators *and* and *or*, respectively. The symbols $\cap$ and $\cup$ signify the set intersection and union, respectively. If A is a set, $|A|$ represents its cardinality.

When $x \in \mathbb{R}^n$, we denote $\|x\| \in \mathbb{R}$ as the euclidean norm of x. Also, for any $x \in \mathbb{R}$, we define (van der Wal, 1980) $x^+ = \max\{x, 0\}$ and $x^- = \min\{x, 0\}$. Therefore, the absolute value is $|x| = x^+ - x^-$.

In order to simplify the notation, from now on we will use $\mathbf{x}(t) = \mathbf{x}^t$. Therefore, for time $t+1$, we will use $\mathbf{x}^{t+1}$.

Finally, we are going to arbitrarily use the masculine pronoun "he" to refer to a pursuer, while using the feminine pronoun "she" for an evader.

2.2 State Representation

We start by considering a grid with a finite set of cells $\mathcal{X} = \{1, 2, \cdots, n_c\}$. Let us also consider that there are two groups of players. One group is known as the *pursuers* and the other as the *evaders*. The set of pursuers is $U = \{U_1, \cdots, U_{n_p}\}$ and the set of evaders is $D = \{D_1, \cdots, D_{n_e}\}$. Therefore, there are n_p pursuers and n_e evaders. One may also define the set of all players as $N = U \cup D$ or $N = \{U_1, \cdots, U_{n_p}, D_1, \cdots, D_{n_e}\}$. It is clear that

$|N| = n_p + n_e = \eta$. Since the number of players is countable, we are simply going to denote a player by a positive integer number $k \in \{1, \cdots, \eta\}$.

Consider also that the game is quantized in time. The positions of the players are defined for all time steps $t \in \mathcal{T}$ for $\mathcal{T} = \{1, 2, \cdots\}$. If we denote by $\mathbf{x}^t = (\mathbf{x}_1^t, \cdots, \mathbf{x}_\eta^t) \in \mathcal{X}^\eta$ the positions at time t then the state of the game is only $\mathbf{x}(t)$. Notice that we assume that the number of possible states $x \in \mathcal{X}^\eta$ of the system is finite, let us say $n_s \in \mathbb{Z}^+$. Hence, the cardinality of the set $\mathcal{X}^\eta$ of possible states of the game is $|\mathcal{X}^\eta| = n_s$.

2.3 Transition Probabilities

Transition probabilities of the game govern the evolution of the game from an arbitrary state $x^t \in \mathcal{X}^\eta$ at time t to another state $x^{t+1} \in \mathcal{X}^\eta$ at time $t+1$. In order for the results to be described later to carry, the initial state x^0 is assumed to be independent of all the other random variables at time $t = 0$.

Each player k has a set of *allowable actions*. We denote this set by $\mathcal{A}_k$ and the actual set $\mathcal{A}_k$ depends on the current state of the game. An action is the desired next allowable cell that the player wants to reach. Therefore, we can define a vector of desired actions at time t for all the players involved in the game as

$$\alpha^t = [\alpha_1^t, \cdots, \alpha_\eta^t] \tag{1}$$

where each α_k^t is the action chosen by each player k at time t from his or her set of allowable actions $\mathcal{A}_k$. Observe then that $\alpha^t \in \mathcal{A}^\eta := \mathcal{A}_1 \times \cdots \times \mathcal{A}_\eta$.

The Markov game formalism (van der Wal, 1980) requires that the probability of transition from one arbitrary state to another is only a function of the action $\alpha \in \mathcal{A}^\eta$ chosen by each of the players at time t when the game is at state $x^t \in \mathcal{X}^\eta$. Hereafter we assume a stationary transition probability, i.e., $\mathbb{P}(\mathbf{x}^{t+1} = x^{t+1} | \mathbf{x}^t = x^t, \mathbf{a}(t) = \alpha) = \mathbb{P}(x^t, \alpha, x^{t+1}),\ x^t, x^{t+1} \in \mathcal{X}^\eta,\ \alpha \in \mathcal{A}^\eta,\ t \in \mathcal{T}$. The function $\mathbb{P} : \mathcal{X}^\eta \times \mathcal{A}^\eta \times \mathcal{X}^\eta \to [0, 1]$ is the *transition probability function*.

Moreover, given the current state of the game $\mathbf{x}^t$, we assume that the positions of all players k at the next time instant are determined independently by each action $\mathbf{a}_k(t)$. Hence, the probability the state is going to change from state x^t to state x^{t+1} due to action α may be written as

$$\mathbb{P}(x^t, \alpha, x^{t+1}) = \prod_{k=1}^{\eta} \mathbb{P}(x_k^t \overset{\alpha_k}{\to} x_k^{t+1}) \tag{2}$$

where $x_k^t \in \mathcal{X}$ is the position of the k^{th} agent *before* it moves and $x_k^{t+1} \in \mathcal{X}$ is the position of the k^{th} agent *after* it moves.

Furthermore, the probability that a player effectively reaches the chosen cell is ρ_k. This translates into:

$$\mathbb{P}((x_k^t) \overset{\alpha_k}{\to} x_k^{t+1}) = \begin{cases} \rho_k, & x_k^{t+1} = \alpha_k \in \mathcal{A}_k(x_k^t) \\ 1 - \rho_k, & x_k^{t+1} = x_k^t \wedge \alpha_k \in \mathcal{A}_k(x_k^t) \\ 1, & x_k^{t+1} = x_k^t \wedge \alpha_k \notin \mathcal{A}_k(x_k^t) \\ 0, & otherwise \end{cases} \tag{3}$$

Observe that equation (3) applies to both pursuers and evaders. The probabilities ρ_k simulate the speed of the players. If $\rho_i = 1.0$ for player i and $\rho_j = 0.8$ for player j, the speed $\dot{x}_i > \dot{x}_j$. Also, if two pursuers or two evaders try to move to the same cell, it is assumed that none of them can move.

2.4 Players Policies and Strategies

A "policy" for a player is a rule the player uses to select which action to take. The choice is based on past observations of how the game is played (Fudenberg and Levine, 1998). In this work, all policies are *stochastic*.

Let us say that at time t the state of the game is $x^t \in \mathcal{X}^\eta$ and the position of player $k \in \{1, \cdots, \eta\}$ is $x_k^t \in \mathcal{X}$. A *policy* for player k is a function $\varphi_k : \mathcal{X}^\eta \times \mathcal{A}_k(x_k) \to [0, 1]$, such that (Hespanha et al., 2000)

$$\sum_{\alpha_k \in \mathcal{A}_k(x_k)} \varphi_k(x^t, \alpha_k) = 1,\ \forall x^t \in \mathcal{X}^\eta \tag{4}$$

In the following sections we will use the term $\varphi_k^t = \varphi_k(x^t, \alpha_k)$. In order to represent the time evolution of the policy, $(\varphi_k^t)^n$ will be used, meaning that it is the n^{th} update of the policy vector.

A *stochastic strategy* π_k for the k^{th} player is a sequence $\pi_k^0, \pi_k^1, \cdots$. Since the game is *Markovian*, the strategy for the k^{th} player is going to be realized by a policy. Therefore, if we understand the superscript to be the time index for the *state the game*, we can represent his or her strategy by the sequence $\pi_k = \{\varphi_k^0, \varphi_k^1, \cdots\}$. This is so because $\pi_k^t(\alpha)$ depends only on the present state. This means that $\pi_k^t(\alpha|(x^0, \cdots, x^t)) = \varphi_k(x^t, \alpha) = \varphi_k^t$. Notice that without loss of generality we assume that at time t the game is at state x^t. Therefore, t is also the state identifier and this is why the notation φ_k^t can be used.

Now, let $\pi = \{\pi^0, \pi^1, \cdots\}$, where for each $t = 0, 1, \cdots$ each term π^t of the sequence π, represented by $\pi^t = \{\pi_1^t, \cdots, \pi_\eta^t\}$, is the set of strategies for all players at time t. We can then define probability measures dependent on π and state $x^t \in \mathcal{X}^\eta$ and denote it by $\mathbb{P}_{\pi,x^t}$. Therefore, $\mathbb{P}_{\pi,x^t}(E)$ denotes the probability of event E happening when strategy π is used (Derman, 1970) and the state is x^t. Also, if the probability measure is dependent only on policy π_k, we are going to denote it by $\mathbb{P}_{\pi_k,x^t}$. In the same fashion, we may also define the expectation $\mathbb{E}_{\pi,x^t}[.]$. Notice also that the policies φ_k and φ_j, $j \neq k$ are conditionally independent of all random variables representing the states and actions at times smaller or equal to t.

2.5 Rewards

Consider that at each time step t, as given above, the state of the game $\mathbf{x}^t$ changes from the state $\mathbf{x}^t = x^t \in \mathcal{X}^\eta$ to another state $\mathbf{x}^{t+1} = x^{t+1} \in \mathcal{X}^\eta$ due to the action α^t (Derman, 1970). Whenever the transition occurs, let us consider that each player k gets a "reward" associated with the transition from state x^t under the actuation of action α^t. We are going to denote this reward by $R_k(x^t, \alpha^t)$ (Howard, 1960), where $R_k : \mathcal{X}^\eta \times \mathcal{A}^\eta \to \mathbb{R}$.

One may observe that the reward $R_k(x^t, \alpha)$ is also a random variable

$$R_k^t(\alpha) = R_k(x^t, \alpha) \tag{5}$$

Hence, one may also define the expected value of the random variable. If we consider the policies π, action α and state x^t, we may define

$$\mathbb{E}_{\pi,x^t}[R_k^t] = \sum_{\alpha^t \in \mathcal{A}^\eta} \mathbb{P}_{\pi,x^i}(x^t, \alpha^t) R_k(x^t, \alpha^t) \tag{6}$$

Observe that equation (6) is only the definition of the expected value for a discrete random variable, and, the probability on the right hand side is

$$\mathbb{P}_{\pi,x^t}(x^t, \alpha^t) = (\prod_{k=1}^{\eta} \varphi_k(x^t, \alpha^t)) \tag{7}$$

and, therefore,

$$\mathbb{E}_{\pi,x^t}[R_k^t] = \sum_{\alpha \in \mathcal{A}^\eta} (\prod_{j=1}^{\eta} \varphi_j(x^t, \alpha^t)) R_k(x^t, \alpha^t) \tag{8}$$

We are going to investigate equations (6), (7) and (8) further in the next sections.

2.6 Pursuit-Evasion Games Model

Markov games (Shapley, 1953) are extensions of the Markov decision process (van der Wal, 1980). In a Markov game, actions are the joint result of multiple agents choosing an action individually (Vrancx et al., 2008).

Since the game may be only in states that belong to the set $\mathcal{X}^\eta$, we may denote by $\mathcal{A}_k^t(x_k)$ the action set available to player $k \in \{1, \cdots, \eta\}$ when the game is found in state $x^t \in \mathcal{X}^\eta$. Furthermore, recall that the sets $\mathcal{A}_k^t(x_k)$ follow some rule that may be different for each k^{th} player.

The game is considered over when all the evaders are captured. This happens when a pursuer $i \in \{1, \cdots, n_p\}$ occupies the same cell as all the evaders $j \in \{1, \cdots, n_e\}$. Clearly, this implies that $n_p \geq n_e$. Consider $x_{p,i}$ as being the cell location of the i^{th} pursuer and $x_{e,j}$ as being the location of the j^{th} evader. Likewise, x_p is a vector with the position of all pursuers and x_e is the vector with the position of all evaders, then $x = (x_p, x_e) \in \mathcal{X}^\eta$. Therefore, we formally define the set $\mathcal{X}_{over}^\eta = \{(x_p, x_e) \in X : x_{e,j} = x_{p,i}, \forall j \in \{1, \cdots, n_e\}$, *and for some* $i \in \{1, \cdots, n_p\}\}$ as the *game-over set*. We assume that all players can detect when the game enters $\mathcal{X}_{over}$.

Hence, we may say that a pursuer's objective is to choose an action that would move him towards a faster end of the game (i.e., he will achieve a point in the game-over set $\mathcal{X}_{over}$) and an evader's objective is to choose an action that would make her more difficult for the pursuers to catch. These actions are dependent on the rewards that each player gets. We can then formally define the pursuit-evasion game as follows.

Definition 1 *A Markov multiple pursuer-evader game is represented by the tuple* $\Gamma = \{N, \mathcal{X}^\eta, \mathcal{A}^\eta, \pi, R(.)\}$.

Moreover, we also assume the following.

Assumption 1 *Each player* $k \in \{1, \cdots, \eta\}$ *does not have full information about any other player* $j \in \{1, \cdots, \eta\}$ *such that* $k \neq j$. *In particular, player* k *does not know the reward function* $R_j(\cdot)$, *action space* $\mathcal{A}_j$ *and strategy* π_j.

Now that the definition of the game is formally completed, we want to focus on the existence of an algorithm that would lead the game to a solution. This is going to be done in the following sections.

3 The Learning approach

The problem of learning in a Markov process has had a lot of interest in the social sciences (Suppes and Atkinson, 1960; Norman, 1972) and in engineering (Wheeler and Narendra, 1986).

The technique that we are going to use is known as "*Learning Automata*" (Thathachar and Sastry, 2004). Using these automata, we want to solve the online stochastic optimization problem (Posnyak and Najim, 1997) of finding the best policies for each player.

The automata we are going to focus on in this work is known as *Finite Action Learning Automata* (FALA). This is so because the actions available to each player at each time step is *finite*. More specifically, we are going to focus on the subclass of algorithms known as *linear reward-penalty* algorithms (Thathachar and Sastry, 2004). Moreover, it must be emphasized that no pre-adaptation is required and all the quantities necessary are computed online during the operation of the algorithm.

3.1 The Learning Algorithm

The algorithm we are going to use is a variation of the *linear reward-penalty automaton* (Thathachar and Sastry, 2004), or L_{R-P}.

Let us consider an n-action automaton, i.e., an automaton that has n different actions from which it can choose. Also, as described in section 2.5, when an action is chosen, the automaton receives some *reward* $R(\alpha)$ from the environment (Thathachar and Sastry, 2004).

Suppose that at time step t, the action chosen from the n possible actions $\{\alpha_{k1}, \cdots, \alpha_{kn}\}$ by player k is $\alpha_k^t = \alpha_{ki}$. Let us also consider the probabilities of choosing any one of the actions as a vector $\varphi_k^t = [\varphi_{k1}^t \cdots \varphi_{kn}^t]$ where $\sum_{j=1}^{n} \varphi_{kj}^t = 1$. Recall that this is what we previously called the *policy vector*. Then the vector φ_k^t is updated as follows (Thathachar and Sastry, 2004, p. 17)

$$(\varphi_{ki}^t)^{n+1} = (\varphi_{ki}^t)^n + \lambda_1 R_k^t(\alpha^t)(1 - (\varphi_{ki}^t)^n) - \lambda_2(1 - R_k^t(\alpha^t))(\varphi_{ki}^t)^n \tag{9}$$

$$(\varphi_{kj}^t)^{n+1} = (\varphi_{kj}^t)^n - \lambda_1 R_k^t(\alpha^t)(\varphi_{kj}^t)^n + \lambda_2(1 - R_k^t(\alpha^t))(\frac{1}{n-1} - (\varphi_{kj}^t)^n),\ j \neq i \tag{10}$$

The rewards $R_k^t(\alpha^t)$ are *stochastic* and, in general, different. Therefore, if we have two different agents k and j, the rewards $R_k^t(\alpha^t)$ and $R_j^t(\alpha^t)$ are, in general, $R_k^t(\alpha^t) \neq R_j^t(\alpha^t)$. In real applications, the rewards $R_k^t(\alpha^t)$ represent rewards or penalties from the environment. In the scenario presented in section 6, they are the distance between the players. In a case of a robot following a line, for example, it would be the sensor reading of the distance of the robot from the line.

The rationale behind equations (9) and (10) is that the action that is chosen (i) will receive a positive reward, while all others (all j's) will receive a penalty that is equally distributed among all of them in order to keep the total probability equal to 1. In this way, if the system converges, one of the φ_{kl} will converge to a value close to 1.

3.2 Learning Algorithm Applied to the Pursuer-Evader Game

Now, let us return to the game defined in section 2.6. Recall that at each time $t \in \mathcal{T}$, each $\alpha_k^t \in \mathcal{A}_k$ is the desired position for player k (who may be a pursuer or an evader) at the next time instant. These are called the *actions* at each player's disposal. Therefore, we assume here the one-step motion for both players are constrained, since $\mathcal{A}_k \subset \mathcal{X}$. In the same way, recall that $\mathcal{A}_k(x_k)$ is the set of *reachable* cells given its current position x_k and we say that these cells are *adjacent* to x_k for player k. Let us discuss an example.

Let us consider that there are two pursuers ($n_p = 2$) and one evader ($n_e = 1$) in the game. Let us further assume that the cells reachable to each pursuer are those neighboring cells located at north, south, east and west. Therefore, unless the pursuer k is located at the sides of the grid, he has five possible actions at his disposal, the four neighboring

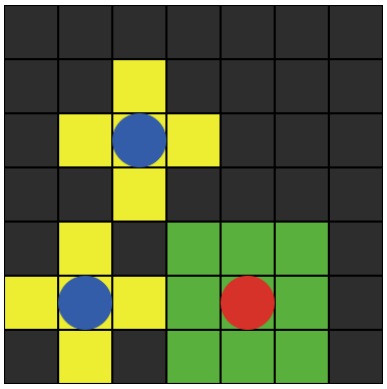

Fig. 1: Grid of the Markov Game

cells plus to keep his own position, and $|\mathcal{A}_k(x_k(t))| = 5$. For the evaders, let us assume that they can move to north, south, east, west plus the diagonals. Therefore, unless the evader j is located at the sides of the grid, she has nine actions at her disposal, the eight neighboring cells plus to keep her own position, and $|\mathcal{A}_j(x_j(t))| = 9$. This is shown graphically in figure 1.

In the game described in figure 1, $n_s = 49 \times 49 \times 49 = 117649$ states. It is easy to see that as we increase the number of cells or the number of players, the number of states grows very fast. This has an impact on the speed of the learning of the system, for each decision is based on the states of the game. Recall that the policies of each player (section 2.4) is a mapping from the state space and action space into a simplex. Therefore, in order to decide the action that is going to be taken, the players must check a table that stores the probability of each action being executed given the state in which the game is. In the beginning of the game, it is assumed that all probabilities are the same and the player behaves erratically and randomly.

When the player chooses an action, the probabilities for a given state are updated as described in equations (9) and (10). After the game is played enough times it is expected that the overall performance of the players will improve. The whole game procedure is given in algorithm 1.

Algorithm 1 Adaptation algorithm

Initialize the states of the system and a table of actions to each player $k \in \{1, \cdots, \eta\}$ according to the rule $\mathcal{A}_k(x_k)$.
Initialize the learning parameters λ_1 and λ_2 to the desired values.
Place all players $k \in \{1, \cdots, \eta\}$ randomly in the play grid.
while All the evaders are not captured **do**
  Get the state of the system $\mathbf{x}^t = [x_1^t, \cdots, x_\eta^t]$
  for All evaders j **do**
    Calculate her next action $\alpha_j^t \in \mathcal{A}_j(x_j^t)$.
    if Player moves to α_j^t with probability given by (3) **then**
      $x_j^{t+1} = \alpha_j^t$
    end if
  end for
  for All pursuers i **do**
    Calculate his next action $\alpha_i^t \in \mathcal{A}_i(x_i^t)$.
    if Player moves to α_i^t with probability given by (3) **then**
      $x_i^{t+1} = \alpha_i^t$
    end if
  end for
  for All players $k \in \{1, \cdots, \eta\}$ **do**
    Calculate reward $R_k^t(\alpha^t)$
    Learn according to equations (9) and (10).
  end for
end while

In the next section, we are going to present some theorems that show that the algorithm given provides a framework for the convergence to the Nash equilibrium of the game.

4 Objective of Learning

4.1 Cost Functions

Let us consider that the pursuit game is played and that the current time step is $t \in \mathcal{T}$. The question that arises is what is the accumulated reward. The expected reward for player k is dependent on the initial state $x^0 \in \mathcal{X}^\eta$ and the strategies π that are played by each player (van der Wal, 1980). Therefore, it may be defined as

$$v_k^t(x^0, \pi) = \mathbb{E}_{\pi, x^0}[\sum_{j=0}^{t-1} R_k^j(\alpha) + v(x^t)] \tag{11}$$

The term $v(x^t)$ on the right hand side of equation (11) is the *terminal payoff* of the game and it is defined as

$$v(x^t) = \begin{cases} v_{over}, \, if \; x^t \in \mathcal{X}_{over} \\ 0, \qquad otherwise \end{cases} \tag{12}$$

Notice that in Game Theory, the usual term for *reward* is *payoff* and this is the reason for the change of nomenclature. In this paper, both terms may be considered synonyms. However, we will use reward when we discuss the learning algorithm and payoff when discussing the convergence of the game.

Moreover, in order for the right hand side of equation (11) to be properly defined, we must have (van der Wal, 1980)

$$\mathbb{E}_{\pi, x^0}[\sum_{j=0}^{t-1} \|R_k^j(\alpha)\|] < \infty \; , \; \forall x^0 \in \mathcal{X}^\eta \tag{13}$$

$$\mathbb{E}_{\pi, x^0}[\|v(x^t)\|] < \infty \; , \; \forall x^0 \in \mathcal{X}^\eta \tag{14}$$

Since there may be different reward functions for different players, equation (11) implies that, in general, it is impossible to find an optimal policy for all agents playing as a group (Vrancx et al., 2008). In other words, individual optimization does not imply group optimization (Wheeler and Narendra, 1986). The problem of finding some kind of individual optimization that would lead to group optimization is a design problem and we will discuss it further when we present our simulations.

Let us define a function $f : \mathcal{X}^\eta \times \mathcal{A}^\eta \to \mathcal{X}^\eta$ as a *Markov mapping* from a state to another depending on an action α. Therefore, equation (11) could be written iteratively as (Bertsekas, 1995)

$$v_k^0 = v_0 \tag{15}$$

$$v_k^{t+1}(x^t, \pi) = \mathbb{E}_{\pi, x^i}[R_k^t(\alpha) + v_k^t(f(x^t, \alpha), \pi)] \tag{16}$$

where $x^{t+1} = f(x^t, \alpha)$ is a mapping from the current state x^t to x^{t+1}.

Since equations (11) and (16) represent the same quantity, in the next subsections we are going to state the objective of the game interchangeably using any one of these equations.

4.2 Optimal Solution

Before we go any further, we must define what an optimal (or nearly optimal) solution for the problem means. This is referred as the *Nash equilibrium* of a multi player game (Sastry et al., 1994).

Definition 2 *The array of strategies* $\pi^* = (\pi_1^*, \cdots, \pi_\eta^*)$ *is called a* **Nash equilibrium** *if for each* k, $1 \leq k \leq \eta$, $\forall \pi \in [0,1]^{\mathcal{A}}$, *we have*

$$v_k^t(x^0, (\pi_1^*, \cdots, \pi_\eta^*)) \geq v_k^t(x^0, (\pi_1^*, \cdots, \pi_{k-1}^*, \pi_k, \pi_{k+1}^*, \cdots, \pi_\eta^*)) \tag{17}$$

Definition 2 means that the optimal solution for the game is the maximization of the cost functions of each individual player. Furthermore, this definition refers to the *strategies* used by each player. However, recall that since the game is Markovian, the strategies are realized, at each time instant $t \in \mathcal{T}$, by a *policy* as described in section 2.4. We are going to discuss the relationship between strategies and policies further when we discuss the convergence of the learning algorithm.

4.3 Statement of Objective

The objective of the game is to find the optimal (or suboptimal) solution for each player in a decentralized fashion. The format of the reward may vary from player to player, but they are written in such a way that all the players try to maximize their gains.

Furthermore, it must also be emphasized that no prior computation must be performed in order to pre-adapt some parameters (Wheeler and Narendra, 1986) and that only minimal (or no) communication is allowed among the players. Therefore, the problem may be formalized as follows.

Problem Statement 1 *Find the policy π_k for each player $k \in \{1, \cdots, \eta\}$ such that the cost functions described in equation (11) are maximized (or nearly maximized) for all players k.*

The expected value in equation (11) is justified by the fact that the learning algorithm does not operate in the space of actions (Thathachar and Sastry, 2004, p. 58), but it updates the probability distributions φ_k for the actions of all automata $k \in \{1, \cdots, \eta\}$.

5 Convergence

As stated before, we are interested to know if the game defined in section 2.6 with the learning algorithm as defined in section 3.2 will converge to the Nash equilibrium as discussed in section 4.2.

In order to show that the optimal play may be found, it is sufficient to show that the learning process takes the system to a mapping that maximizes the cost of equation (11). The problem could be solved by using *dynamic programming*; however, the problem is that in order to do so it is required that the players have access to all possible payoffs ahead of time and the number of combinations grows very fast (Wheeler and Narendra, 1986).

Therefore, this section is divided as follows. We will start, in section 5.1, by presenting a sketch of the proof that will be presented in the next subsections. In this section, we will also link the notation of learning in automata systems to the notation we are using so far. Section 5.2 shows that an Ordinary Differential Equation can be used in place of the difference equation used to describe the algorithm. Then, in section 5.3 we will state the theorems that show that the learning automata system converges to the equilibrium point as required. Finally, section 5.4 shows that the game converges to the desired Nash equilibrium.

5.1 Sketch of the Proof

In order to prove that the learning algorithm of equations (9) and (10) converges to the strategy that achieves the Nash equilibrium of definition 2, we need to prove three separate theorems. Theorem 1 deals with the possibility to represent the algorithm by an Ordinary Differential Equation (ODE). Theorem 2 deals with the convergence of the algorithm to some stable optimal policies (a Nash equilibrium). Finally, theorem 3 states the convergence of the strategies to the Nash equilibrium (definition 2).

We are going to state all theorems in this section and will prove them separately in the subsequent sections.

Notice that there are two different time steps involved in the algorithm. The first is the change of states of the game and is referred to as t. This index is used as a superscript in the state $x^t \in \mathcal{X}^\eta$, for example. The other time instant will be referred to as n and it refers to the number of times a given state is visited and is used in referencing the evolution of the policies. Therefore, $\varphi_k^n(x^t, \alpha^t)$ refers to the n^{th} visit that player k makes to state x^t. For short, we are going to use $\varphi_k^n(\alpha^n) = \varphi_k^n(x^t, \alpha^t) = (\varphi_k^t)^n$ in the remaining of the article, unless it is strictly necessary to explicitly include the state x^t.

Now, let us go back to equations (9) and (10). Consider that λ_2 is set to zero, i.e., $\lambda_2 = 0$. If we also set $\lambda_1 = \lambda$, this would result in the evolution of the policies φ_k to (Thathachar and Sastry, 2004)

$$\varphi_k^{n+1} = \varphi_k^n + \lambda G(\varphi_k^n, \xi_k^n) \tag{18}$$

where $\xi_k^n = [\alpha_k^n, R_k^n]$ and the function $G(\cdot)$ is the updating given by equations (9) and (10).

For the case described above, consider that $e_{\alpha_k^n}$ is a unit vector with the same dimension of φ_k^n with the unit at the position that represents the action α_k^n. Also, let us represent $\xi_k^n = [\alpha_k^n, R_k^n] = [\xi_{k1}^n, \xi_{k2}^n]$, then function $G(\cdot)$ is

$$\begin{aligned} G(\varphi_k^n, \xi_k^n) &= R_k^n(e_{\alpha_k^n} - \varphi_k^n) \\ G(\varphi_k^n, \xi_k^n) &= \xi_{k2}^n(e_{\xi_{k1}^n} - \varphi_k^n) \end{aligned} \tag{19}$$

One could observe that even the equations (9) and (10) could also be described with equation (18). In order to do so, the only thing necessary is to set λ as a vector encompassing λ_1 and λ_2 and change the definition of $G(\cdot)$. This

would not change our results considerably. However, some details would become more cumbersome and we chose to simplify the approach taken.

Moreover, let us assume, without loss of generality, that the reward $\xi^n_{k2} = R^n_k \in [0,1]$, i.e., it is normalized. Also, let us represent $G(\cdot) = [G_1(\cdot), \cdots, G_{r_k}(\cdot)]$ where r_k is the number of actions at player k's disposal, therefore

$$G_i(\varphi^n_k, \xi^n_k) \in [0,1], \ \forall i = 1, \cdots, r_k \tag{20}$$

With these considerations, we turn our attention to the sketch of the proof that will be implemented in the next subsections.

The first theorem (theorem 1) states that the learning algorithm in its discrete form can be represented by an ODE. After this is established, we can state theorem 2 that guarantees that, in any given state, the learning algorithm results in the convergence to the Nash equilibrium (Thathachar and Sastry, 2004) in the policy space. And finally, with the statement of theorems 1 and 2, we can prove the convergence of the game defined in section 2.6 to the Nash equilibrium defined in 4.2. In other words, the learning algorithm converges to the optimal policies independent of the initial state $x^0 \in \mathcal{X}^\eta$.

5.2 Analysis of the ODE

In this subsection, since it is clear that we are discussing a feature that is common to all players $k \in \{1, \cdots, \eta\}$, we are going to drop the subscript k from the equations that describe the evolution of the learning algorithm. We also introduce the notation $\varphi^{n,\lambda}$, $\xi^{n,\lambda}$, $G^\lambda(\cdot)$ etc. to emphasize that the evolution of the processes is dependent on a parameter λ that is supposed to be small.

Let us again consider the process $\{G(\varphi^n, \xi^n)\}$ in equation (19). Recall that $\xi^n = [\alpha^n, R^n]$. Now, let us assume that the process $\{\xi^n\}$ depends on the states of the game in a Markovian way. We are going to deal with the actual form of this dependency later. But, for now, it suffices to notice that the reward R^n_k is dependent on the strategies π used and, therefore, must depend on the past actions taken so far.

Let us next define the expected value of $G(\cdot)$ as (Kushner and Yin, 1997, p. 37)

$$\mathbb{E}[G(\varphi^n, \xi^n) | \varphi^i, \xi^i, i \leq n] = \mathcal{G}(\varphi^n, \xi^n) \tag{21}$$

From equation (20) it may also be concluded that

$$\{\mathcal{G}(\varphi^{n,\lambda}, \xi^{n,\lambda}); \lambda, n\} \ \textit{is uniformly integrable} \tag{22}$$

And also define

$$\begin{aligned} \theta^n &= G(\varphi^n, \xi^n) - \mathbb{E}[G(\varphi^n, \xi^n)] \\ \theta^n &= G(\varphi^n, \xi^n) - \mathcal{G}(\varphi^n, \xi^n) \end{aligned} \tag{23}$$

Then, we can rewrite equation (18) as

$$\varphi^{n+1} = \varphi^n + \lambda \mathcal{G}(\varphi^n, \xi^n) + \lambda \theta^n \tag{24}$$

where θ^n are martingale differences. However, the conditional mean term $\mathcal{G}(\varphi^n, \xi^n)$ is not simple (Kushner and Yin, 1997) due to the fact that the evolution of $\{\xi^n\}$ depends on the evolution of the sequence $\{\varphi^n\}$.

Since we are interested in studying the asymptotic behavior of $\varphi^{n,\lambda}$, it is useful to introduce a process wherein the parameter φ is kept constant. Let $\{\xi^n(\varphi)\}$ denote a Markov chain that results from $\{\xi^n\}$ when the parameter $\varphi^{n,\lambda}$ is held constant at value φ (Kushner and Yin, 1997, p. 37). If λ is small, then $\varphi^{n,\lambda}$ varies slowly and it may be argued that the law of large number suggests that

$$\bar{\mathcal{G}}(\varphi) = \mathbb{E}_\varphi[\mathcal{G}(\varphi, \xi^n(\varphi) | \varphi] \tag{25}$$

where $\bar{\mathcal{G}}(\varphi)$ is a continuous function of φ. Henceforth, we want to show that the algorithm in equation (24) may be represented by

$$\dot{\varphi} = \bar{\mathcal{G}}(\varphi) \tag{26}$$

Notice that since the process $\{\varphi^{n,\lambda}\}$ is bounded, the limit set represented by equation (26) is also bounded. Furthermore, one should notice that we are looking for the zeros of $\bar{\mathcal{G}}$.

We can now enunciate the theorem.

Theorem 1 *For any initial condition φ^0, if we assume the algorithm described in equation (18), the sequences $\{\varphi^n\}$ are trajectories of equation (26).*

Proof. This theorem is satisfied if the conditions of theorem 8.4.4 of (Kushner and Yin, 1997, p. 247) are satisfied. The condition on the uniform integrability of $G(\varphi^{n,\lambda}, \xi^{n,\lambda})$ is satisfied according to the discussions that led to equation (22). Observe also that according to equation (21), function $\mathcal{G}(\cdot)$ does not depend explicitly on n and λ and the noise term assumed in the theorem of (Kushner and Yin, 1997) can be taken to be zero.

Now, observe that $\mathcal{G}(\cdot)$ is also bounded and continuous. Therefore, small variations on the parameters φ will not lead to large variations in the update of the algorithm as described by equation (24). Therefore, conditions $A4.16'$ and $A4.17'$ of (Kushner and Yin, 1997, p. 247) are readily satisfied.

Let us now, focus on condition $A.4.6$ of (Kushner and Yin, 1997, p. 239) and let us write the equation according to the conditions of the present problem

$$\lim_{N\to\infty} \frac{1}{N} \sum_{i=jN}^{jN+N-1} \mathbb{E}[\mathcal{G}(\varphi, \xi^i(\varphi)) - \bar{\mathcal{G}}(\varphi)]$$
$$= \lim_{N\to\infty} \frac{1}{N} \sum_{i=jN}^{jN+N-1} \mathbb{E}[\mathcal{G}(\varphi, \xi^i(\varphi)) - \mathbb{E}_\varphi[\mathcal{G}(\varphi, \xi^i(\varphi))]]$$

Observe that the term $\mathcal{G}(\varphi, \xi^i(\varphi)) - \mathbb{E}_\varphi[\mathcal{G}(\varphi, \xi^i(\varphi))]$ is only white noise with an unknown distribution. Therefore, it may be concluded that the limit goes to zero and condition $A.4.6$ holds.

Finally, recall that the tightness of $\{\xi^{n,\lambda}\}$ is guaranteed by the definition of the rewards R^n and the boundedness of $\{G(\cdot)\}$. Therefore, the function

$$\mathbb{P}(\xi, A|\varphi) = P(\xi^{n+1,\lambda} \in A|\varphi^{n,\lambda} = \varphi, \xi^{n,\lambda} = \xi)$$

is measurable for all the range space of ξ and the remaining conditions of the theorem are satisfied.

Therefore, the evolution of policies may be represented by equation (26). □

Now that we have shown that the game may be approximated by the ODE, we are ready to prove the convergence of the policies as updated by the learning automata algorithm to the Nash equilibrium.

5.3 Analysis of the Learning Algorithm

We now turn our attention to the problem of the convergence of the algorithm in each state to the Nash equilibrium, i.e., given that the game is at state $x^t \in \mathcal{X}^\eta$, the policies of the k^{th} player converge to a set of policies that maximize the gains that the player will have.

Recall that we defined the reward at time $t \in \mathcal{T}$ for a game starting at state x^0 and player k to be equation (11). Also, let us define the *payoff function* $\beta_k(x^t, \alpha)$ as

$$\beta_k(x^t, \alpha) = R_k(x^t, \alpha) + v_k(f(x^t, \alpha), \pi) \tag{27}$$

We can then define the expected payoff due to an action.

Definition 3 *Functions* $\mathcal{F}_k^{x^i} : \prod_{j=1}^{\eta} \mathcal{A}_j \to [0,1]$ *for* $k \in \{1, \cdots, \eta\}$ *and* $x^t \in \mathcal{X}^\eta$ *are defined by*

$$\mathcal{F}_k^{x^t}(\alpha) = \mathbb{E}_{\pi_k, x^t}[\beta_k(x^t, \alpha)|\alpha \in \mathcal{A}^\eta] \tag{28}$$

Furthermore, the players have no knowledge of the payoff functions given by the environment, they only have access to the payoff signals (the $\beta_k(x^t, \alpha)$*).*

This definition actually encompasses some assumptions as well. The first is that the players have incomplete information on the environment and on the behaviour of the other players. This is done in order to approximate the algorithm with real life situations. Also, it may be noticed that

$$v_k^{t+1}(x^t, \pi) = \sum_{\alpha\in\mathcal{A}} (\prod_{i=1}^{\eta} \varphi_i(\alpha)) \mathcal{F}_k^{x^t}(\alpha) \tag{29}$$

Let us now define the vector of policies $\boldsymbol{\Phi}_{x^t} \in \boldsymbol{\Psi}_{x^t}$ as being the policies when the game is in state $x^t \in \mathcal{X}^\eta$, i.e, $\boldsymbol{\Phi}_{x^t} = [\varphi_1^{x^t}, \cdots, \varphi_\eta^{x^t}]$. We now define yet another set of functions $\mathcal{V}_k^{x^t} : \Psi_{x^t} \to [0,1]$ as

$$\mathcal{V}_k^{x^t}(\boldsymbol{\Phi}_{x^t}) = \mathbb{E}_{\pi_k, x^t}[\beta_k(x^t, \alpha)|\boldsymbol{\Phi}_{x^t}] \tag{30}$$

i.e., the payoff for the k^{th} player under the set of policies Φ_{x^t}. Therefore, functions $\mathcal{F}_k^{x^t}$ are the expected payoffs due to the action chosen while $\mathcal{V}_k^{x^t}$ are the expected payoffs due to the policy profile (or probability distribution).

Now we are ready to determine what a maximal point for state $x^t \in \mathcal{X}^\eta$ is (Thathachar and Sastry, 2004, p. 59)

Definition 4 *Consider the set of policies $\Phi^*_{x^t} = [\varphi_1^{x^t *}, \cdots, \varphi_\eta^{x^t *}]$. We say that Φ^* is a maximal point (and in this context a Nash equilibrium) of the game at stage $x^t \in \mathcal{X}^\eta$ if for each $k \in \{1, \cdots, \eta\}$*

$$\mathcal{V}_k^{x^t}(\Phi^*_{x^t}) \geq \mathcal{V}_k^{x^t}(\Phi_{x^t})$$

for all Φ such that $\Phi = [\varphi_1^{x^t *}, \cdots, \varphi_{k-1}^{x^t *}, \varphi_k^{x^t}, \varphi_{k+1}^{x^t *}, \cdots, \varphi_\eta^{x^t *}]$, with $\varphi_k^{x^t} \in |\mathcal{A}_k|$ is a probability vector different from $\varphi_k^{x^t *}$, i.e., $\varphi_k^{x^t} \neq \varphi_k^{x^t *}$.

In the same fashion, we may also define the maximal point for the set of actions (a *modal point*) as follows (Thathachar and Sastry, 2004, p. 61)

Definition 5 *The the set of actions $\alpha^* = [\alpha_1^*, \cdots, \alpha_\eta^*]$ is a modal point (and in this context also a Nash equilibrium) of the game at stage $x^t \in \mathcal{X}^\eta$ if for each $k \in \{1, \cdots, \eta\}$*

$$\mathcal{F}_k^{x^t}(\alpha^*) \geq \mathcal{F}_k^{x^t}(\alpha)$$

for all $\alpha = [\alpha_1^*, \cdots, \alpha_{k-1}^*, \alpha_k, \alpha_{k+1}^*, \cdots, \alpha_\eta^*]$ such that $\alpha_k \neq \alpha_k^*$.

We also need an extra assumption before we proceed to prove the theorem.

Assumption 2 *For every x^t, k and α, $\mathcal{F}_k^{x^t}(\alpha)$ has a finite number of maxima, all in a compact set and has no maxima at infinity.*

This assumption, which is in line with equation (13), is somewhat intuitive since the game should actually have a finite number of maxima due to the fact that it has a finite number of states (Givigi, 2009).

The evolution of each individual policy j for each player k of array Φ_{x^t} is given by the ODE in equation (26) and may be rewritten as

$$\frac{d\varphi_{kj}^{x^t}}{dt} = \bar{\mathcal{G}}_{kj}^{x^t}(\Phi_{x^t}) \tag{31}$$

for all players $k \in \{1, \cdots, \eta\}$ and individual policy $1 \leq j \leq |\mathcal{A}_k|$ at each state x^t.

Let us now define α_k^j as the unit vector with unity in position $1 \leq j \leq |\mathcal{A}_k|$, i.e., the k^{th} player chooses action j. Now, we can define the function

$$f_{kj}^{x^t}(\Phi_{x^t}) = \mathbb{E}_{\pi, x^t}[\beta_k^t | \varphi_l^{x^t}, k \neq l, \alpha_k^j] \tag{32}$$

Function $f_{kj}^{x^t}(\Phi_{x^t})$ is the expected payoff function given that player k chooses action j and all other players $l \neq k$ play their expected mixed strategy defined by vectors φ_l. Observe that if we define the set $\Upsilon \subset \mathcal{A}^\eta$ as the set of all actions with the k^{th} player always choosing action j but all other players choosing every possible combination, we get

$$f_{kj}^{x^t}(\Phi_{x^t}) = \sum_{\alpha \in \Upsilon} \left(\prod_{i \neq k} \varphi_i(\alpha)\right) \mathcal{F}_k^{x^t}(\alpha) \tag{33}$$

One could also write the function $\mathcal{V}_k^{x^t}(\Phi_{x^t})$ in terms of functions $f_{kj}(\Phi_{x^t})$

$$\mathcal{V}_k^{x^t}(\Phi_{x^t}) = \sum_{l=1}^{|\mathcal{A}_k|} f_{kl}^{x^t}(\Phi_{x^t}) \varphi_{kl}^{x^t} \tag{34}$$

Finally, we can write each term $\bar{\mathcal{G}}_{kj}^{x^t}(\Phi_{x^t})$ of vector $\bar{\mathcal{G}}_k^{x^t}(\Phi_{x^t})$ in terms of $f_{kj}^{x^t}(\Phi_{x^t})$ as (Givigi, 2009)

$$\bar{\mathcal{G}}_{kj}^{x^t}(\Phi_{x^t}) = \varphi_{kj}^{x^t} f_{kj}^{x^t}(\Phi_{x^t}) - (\varphi_{kj}^{x^t})^2 f_{kj}^{x^t}(\Phi_{x^t}) - \sum_{q \neq j} \varphi_{kq}^{x^t} \varphi_{kj}^{x^t} f_{kq}^{x^t}(\Phi_{x^t}) \tag{35}$$

Now we can enunciate theorem 2.

Theorem 2 *With the updating rule of equation (24), the algorithm presents the following characteristics.*

i. All strict Nash equilibria in pure policies are asymptotically stable.

ii. Any equilibrium point which is not a Nash equilibrium in pure or mixed strategies is unstable.

Proof. We now consider the first part of the theorem.

i. Let $\Phi^*_{x^t} = [\mathbf{e}^*_{\alpha_1}, \cdots, \mathbf{e}^*_{\alpha_\eta}]$ be a pure maximal point. Now, define a region $\Phi^\rho_{x^t} = \{\Phi_{x^t} \in \mathcal{A}^\eta | \Phi_{x^t} = [\varphi_1^{x^t}, \cdots, \varphi_\eta^{x^t}]\}$ such that each $\varphi_k^{x^t}$, $k \in \{1, \cdots, \eta\}$, is a probability vector of dimension $|\mathcal{A}_k|$ close to the vector $\mathbf{e}^*_{\alpha_k}$ by a distance of $\epsilon > 0$.
Now let $\Phi_{x^t} \in \Psi^\rho_{x^t}$ such that $\Phi_{x^t} = [\mathbf{e}^*_{\alpha_1}, \cdots, \mathbf{e}^*_{\alpha_{k-1}}, \varphi_k, \mathbf{e}^*_{\alpha_{k+1}}, \cdots, \mathbf{e}^*_{\alpha_\eta}]$. Then, substituting equation (35) into equation (31), we would have

$$\frac{d\varphi^{x^t}_{kl}}{dt} = \varphi^{x^t}_{kj} f^{x^t}_{kj}(\Phi_{x^t}) - (\varphi^{x^t}_{kj})^2 f^{x^t}_{kj}(\Phi_{x^t}) - \sum_{q \neq j} \varphi^{x^t}_{kq} \varphi^{x^t}_{kj} f^{x^t}_{kq}(\Phi_{x^t}) \tag{36}$$

Terms $f^{x^t}_{ki}$ and φ_{ki} are in the interval $[0, 1]$. Hence, the two last terms in the RHS of equation (36) are negative. Let us then analyze the term $\varphi^{x^t}_{kj} f^{x^t}_{kj}(\Phi_{x^t})$. Since it is supposed that Φ_{x^t} is close to $\Phi^*_{x^t}$, a Taylor expansion around $\Phi^*_{x^t}$ is valid. Then, a Taylor expansion of (33) is

$$\varphi^{x^t}_{kj} f^{x^t}_{kj}(\Phi_{x^t}) = \varphi^{x^t}_{kj}[\mathcal{F}^{x^t}_k(\alpha) - \mathcal{F}^{x^t}_k(\alpha^*)] + \ high - order \ \ terms$$

where $\alpha = [\alpha^*_1, \cdots, \alpha^*_{k-1}, \alpha_k, \alpha^*_{k+1}, \cdots, \alpha^*_\eta]$, i.e., $\alpha \neq \alpha^*$ just by its k^{th} element.
By definition 5 it is clear that $\varphi^{x^t}_{kj} f^{x^t}_{kj}(\Phi_{x^t}) < 0, \ \ \forall \alpha \neq \alpha^*$ and $\dot{\varphi}^{x^t}_{kl} < 0$.
We now consider a Lyapunov function defined over $\Phi^\rho_{x^t}$ as

$$V_k(\Phi_{x^t}) = \sum_k \sum_{l \neq \alpha_k} \varphi^{x^t}_{kl} \tag{37}$$

Note that we are dealing with pure policies as such $\sum_{l \neq \alpha_k} \varphi^{x^t}_{kl} = 0$ and then $V_k(\Phi^*_{x^t}) = 0$. Also, since $\Phi^*_{x^t}$ is supposed to be a strict maximal point, for every other $\Phi_{x^t} \neq \Phi^*_{x^t}$, $V_k(\Phi_{x^t}) > 0$. Let us now take the derivative of $V_k(\cdot)$ with respect to time

$$\begin{aligned} \dot{V}_k(\Phi_{x^t}) &= \sum_k \sum_{l \neq \alpha_k} \dot{\varphi^{x^t}_{kl}} \\ &< 0, \ \ \forall l \neq \alpha^*_k \end{aligned}$$

Therefore, $\Phi^*_{x^t}$ is asymptotically stable.

ii. Let Φ^* be an equilibrium point of the game Γ. Since it is supposed not to be a Nash equilibrium, it is not a maximal point. Therefore, by definition 4, we have

$$\mathcal{V}^{x^t}_k(\Phi^*_{x^t}) < \mathcal{V}^{x^t}_k(\Phi_{x^t})$$

for some Φ_{x^t} in the neighbourhood of $\Phi^*_{x^t}$. This implies that (Basar and Olsder, 1999) there is a $f^{x^t}_{kj}(\Phi^*_{x^t})$ such that

$$f^{x^t}_{kj}(\Phi^*_{x^t}) > \mathcal{V}^{x^t}_k(\Phi^*_{x^t}) \tag{38}$$

Now, consider that $\bar{\mathcal{G}}^{x^t}_{kj}(\Phi_{x^t})$ may be rewritten as (Givigi, 2009)

$$\bar{\mathcal{G}}^{x^t}_{kj}(\Phi_{x^t}) = \varphi^{x^t}_{kj}[f^{x^t}_{kj}(\Phi_{x^t}) - \sum_q \varphi^{x^t}_{kq} f^{x^t}_{kq}(\Phi_{x^t})]$$

Using equation (34), we get

$$\bar{\mathcal{G}}^{x^t}_{kl}(\Phi_{x^t}) = \varphi^{x^t}_{kl}[f^{x^t}_{kl}(\Phi_{x^t}) - \mathcal{V}^{x^t}_k(\Phi_{x^t})]$$

And, by equation (38) we have that $\bar{\mathcal{G}}^{x^t}_{kl}(\Phi_{x^t}) > 0$, which means that

$$\frac{d\varphi^{x^t}_{kl}}{dt}(\Phi_{x^t}) > 0$$

Therefore, in small neighbourhoods of $\Phi^*_{x^t}$ the policies will diverge and the policy set $\Phi^*_{x^t}$ is unstable.

This completes the proof of the theorem. □

5.4 Analysis of the Game

Finally, in order to prove that the game with the learning algorithm described so far will converge to the Nash equilibrium in the strategy space, we need to prove that the maximization in the policy space as shown in theorem 2 will lead to the optimal solution for the Markov game. Before we proceed to theorem 3, let us enunciate the following lemma.

Lemma 1 *Consider that algorithm 1 is given enough time to converge and let* $v_k^{t\,*}(x^0, \pi^*)$ *be the optimal value for the cost function*

$$v_k^t(x^0, \pi) = \mathbb{E}_{\pi, x^0}[\sum_{j=0}^{t-1} R_k^j(\alpha) + v(x^t)] \tag{39}$$

at time $t \in \mathcal{T}$ *for player* k*. Then, the maximization defined by equation*

$$v_k^{t+1}(x^t, \pi) = \max_{\Phi_{x^t} \in \Psi_{x^t}} \mathbb{E}_{\pi, x^i}[R_k^t(\alpha) + v_k^t(f(x^t, \alpha), \pi)] \tag{40}$$

leads to the optimal value $v_k^{t\,*}(x^0, \pi^*)$ *and, moreover, the strategy* $\pi_k^* = \{\varphi_k^{x^0\,*}, \cdots, \varphi_k^{x^{t-1}\,*}\}$ *is optimal.*

Proof. The maximization derived in theorem 2 may be explicitly written as equation (40) (Givigi, 2009). Also notice that since the game we are considering is Markovian, the optimal value $v_k^{t\,*}(x^0, \pi^*)$ for equation (39) is

$$\begin{aligned}
v_k^{t\,*}(x^0, \pi^*) = & \max_{\varphi_k^{x^0}, \cdots, \varphi_k^{x^{t-1}}} \{\mathbb{E}_{x^0}[R_k(x^0\alpha^0) + \mathbb{E}_{x^1}[R_k(x^1, \alpha^1) + \cdots \\
& + \mathbb{E}_{x^{t-1}}[R_k(x^{t-1}, \alpha^{t-1}) + v(x^t)] \cdots]]\} \\
v_k^{t\,*}(x^0, \pi^*) = & \max_{\varphi_k^{x^0}} \{\mathbb{E}_{x^0}[R_k(x^0, \alpha^0) + \max_{\varphi_k^{x^1}} \{\mathbb{E}_{x^1}[R_k(x^1, \alpha^1) + \cdots \\
& + \max_{\varphi_k^{x^{t-1}}} \{\mathbb{E}_{x^{t-1}}[R_k(x^{t-1}, \alpha^{t-1}) + v(x^t)]\} \cdots]\}]\}
\end{aligned} \tag{41}$$

By using equation (40) in equation (41) sequentially, we get

$$\begin{aligned}
v_k^{t\,*}(x^0, \pi^*) = & \max_{\varphi_k^{x^0}} \{\mathbb{E}_{x^0}[R_k(x^0, \alpha^0) + \max_{\varphi_k^{x^1}} \{\mathbb{E}_{x^1}[R_k(x^1, \alpha^1) + \cdots \\
& + v_k^t(x^t, \varphi_k \pi)]\} \cdots]\} \\
v_k^{t\,*}(x^0, \pi^*) = & \max_{\varphi_k^{x^0}, \cdots, \varphi_k^{x^{t-1}}} \{\mathbb{E}_{x^0}[R_k(x^0 \alpha^0) + v_k^t(x^1, \pi)]\} \\
v_k^{t\,*}(x^0, \pi^*) = & v_k^t(x^0, \pi)
\end{aligned}$$

Since the right hand side and the left hand side of the equation above are equal, it shows that the use of equation (40) at each step of the optimization process indeed leads to the optimal solution for the problem. Also, the equation makes it explicit that the strategy $\pi_k^* = \{\varphi_k^{x^0\,*}, \cdots, \varphi_k^{x^{t-1}\,*}\}$ maximizes equation (40) at every time step related to each state $x^i \in \mathcal{X}^\eta$, $i = 0, \cdots, t$. □

With the proof of lemma 1, we can finally introduce theorem 3. Before we do so, notice that we assume that at time t the game reaches one of the final states, i.e., that $x^t \in \mathcal{X}_{over}$.

Theorem 3 *With the updating rule of equation (24) applied to all possible states* $x^t \in \mathcal{X}^\eta$*, the game defined in definition 1 converges to the optimal strategy profile as defined in definition 2.*

Proof. Lemma 1 shows that the maximization of equation (40) found in section 5.3 converges to the optimal solution. Now, we only need to prove that this optimal solution is the Nash equilibrium in the space of strategies.

Assume that all players $j \neq k$ play their optimal strategies, but player k plays something different, say $\tilde{\pi}_k$. Also, without loss of generality, assume that only the policy at time step $t-1$ is different from the optimal, i.e.,

$$\tilde{\pi}_k = \{\pi_k^{1\,*}, \cdots, \pi_k^{t-2\,*}, \tilde{\pi}_k^{t-1}\}$$

and we call this set of strategies $\tilde{\pi} = \{\pi_1^*, \cdots, \pi_{k-1}^*, \tilde{\pi}_k, \pi_{k+1}^*, \cdots, \pi_\eta^*\}$

Hence, we could write the cost function $v_k^t(\cdot)$ as

$$\begin{aligned} v_k^t(x^0,\tilde{\pi}) &= \{\mathbb{E}_{x^0}[R_k(x^0\alpha^0) + \mathbb{E}_{x^1}[R_k(x^1,\alpha^1) + \cdots \\ &\quad + \mathbb{E}_{x^{t-1}}[R_k(x^{t-1},\alpha^{t-1}) + v(x^t)|\tilde{\pi}_k^{t-1}]\cdots|\pi_k^{1^*}]|\pi_k^{0^*}]\} \\ v_k^t(x^0,\tilde{\pi}) &= \max_{\varphi_k^{x^0*}}\{\mathbb{E}_{x^0}[R_k(x^0,\alpha^0) + \max_{\varphi_k^{x^{1*}}}\{\mathbb{E}_{x^1}[R_k(x^1,\alpha^1) + \cdots \\ &\quad + \mathbb{E}_{x^{t-1}}[R_k(x^{t-1},\alpha^{t-1}) + v(x^t)|\tilde{\pi}_k^{t-1}]]\}] \end{aligned}$$

Since only the policy for state $x^{t-1}$is supposed to be different of the optimal, we may conclude that the reward received will not be larger than the one received if the optimal policy according to lemma 1 was used. Therefore,

$$v_k^t(x^0,\tilde{\pi}) \leq v_k^t(x^0,\pi^*) \qquad \square$$

5.5 Conclusions of the proofs

The proofs of the three theorems allow us several conclusions.

The first one is that even when the payoffs are not supposed to be independent, the algorithm presented will converge to a Nash equilibrium. This is discussed in theorem 1. In (Thathachar and Sastry, 2004), all the cases discussed assume that the payoffs are independent of each other. In our problem, however, the payoffs depend on the past decisions and values of the policy parameters φ such that independence cannot be assumed.

The second conclusion is that the stable maximal points are those with pure strategies. This is in fact a characteristic of the Finite Action Learning Automaton (FALA) (Thathachar and Sastry, 2004) that is made explicit by theorem 2. One should notice that depending on the problem under study, mixed policy Nash equilibria could also arise. However, for the pursuit-evasion game studied in this paper, intuitively these mixed policies do not arise. The general proof of this characteristic is still an open question.

The third conclusion is that given enough time the algorithm will lead to Nash equilibria in the strategy space. This is a result similar to the one reached by Dynamic Programming. However, it must be emphasized again that the process to get to the solution is very different in the approach used in this paper, since the maximization is achieved iteratively through the learning procedure of algorithm 1 and not through the investigation of all possible transitions.

6 Simulation

We now discuss a simulation that shows the feasibility of the suggested approach. In it, only the pursuers learn using the algorithm 1 described so far and we consider a game played in a 7×7 grid.

For this simulation, we place in the grid 2 pursuers and 1 evader. The set of pursuers is defined by $U = \{1, 2\}$ and the set of evaders by $D = \{3\}$. The pursuers have as the *reachable* cells the neighboring cells located at north, south, east and west as depicted in figure 1. The evader can move to north, south, east, west plus the diagonals as also depicted in figure 1.

The evader plays some fixed strategy defined as getting as far from the pursuers as possible. In fact, the evader tries to occupy cells that cannot be reached by the pursuers at the next time step. In doing so, the evader actually postpones the capture time by at least one time step.

Furthermore, the probability that the pursuers will move, ρ_1 and ρ_2, is only 0.8, meaning that 20% of the time the players do not change position when commanded to do so. However, the evader moves all the time, i.e., $\rho_3 = 1.0$. These numbers simulate the speed of the players. Therefore, the pursuers are substantially slower than the evader. Also, notice that the evader is more maneuverable than the pursuers, for she can move to more cells than the pursuers. Hence, in order to capture the evader the pursuers must work together, otherwise the evader can always avoid capture.

The reward function $R_k(x^t, \alpha^t)$ for the pursuers is only a normalized version of the *Manhattan distance* from the pursuer to the evader. The Manhattan distance is the distance between the players measured along axes at right angles. Suppose that the pursuer is located at position $x^p = (x_p, y_p)$ and the evader is located at position $x_e = (x_e, y_e)$. The Manhattan distance between both is simply $|x_p - x_e| + |y_p - y_e|$.

The expected value $v_k^t(\cdot)$ for each state $x^t \in \mathcal{X}^\eta$ is calculated numerically. Since the policies are evolving and we want the expected value given to the use of the current policy, we need to use some time of *forgetting factor* in order to guarantee the convergence of the game. We use a forgetting factor of 0.95 meaning that the expected value is calculated based on the past 20 visits to that state.

In order to establish a baseline for comparison, we run the game 100 times and collect the time required for the pursuers to capture the evader. The mean time for capture and the standard deviation are depicted in table 1.

Table 1: Mean and standard deviation of capture before training.

Mean	Standard Deviation
12,990.11	14,064.27

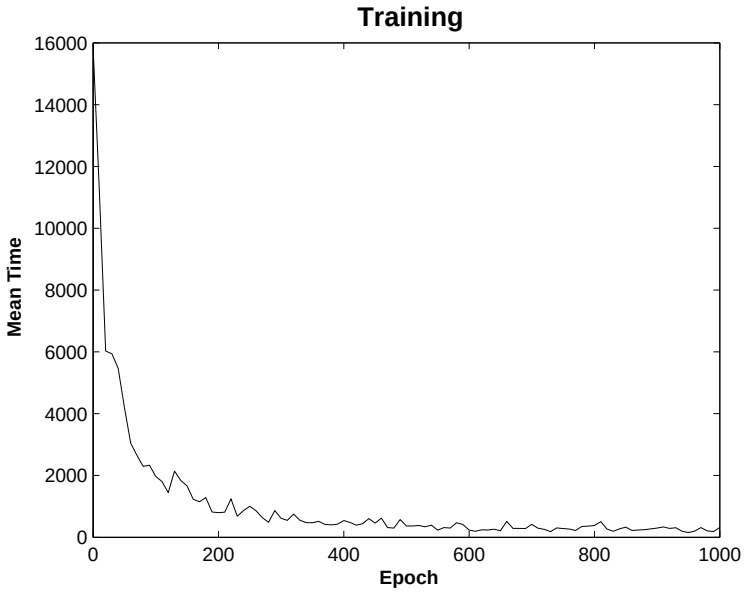

Fig. 2: Performance of the training

We then train the pursuers for 1000 games. We collect the times for capture and in intervals of 10 games calculate the average time for capture. The data is plot in figure 2

In fact, in order to guarantee that the game converges to the optimal solution, the number of iterations (the number of different games played) should be even greater due to the fact that we would need to visit all the states of the game a considerable number of times so the expected values could be evaluated and converge to its real value. Therefore, the game converges to the optimal solution given the current expected values, but it would take longer for the optimal solution to be found. Regardless, one may observe that the average time for capture decreases sharply with training.

In order to assess how well the training is, we run the game for more 100 time and recalculate the mean time and standard deviation for capture. The results are show in table 2. The decrease shows that the pursuers actually "learn" to capture the evader. Although the choice of rewards take into account only the benefit for the player himself actually leads him to cooperate with the other. In summary, the pursuers learn to work together towards an objective.

Table 2: Mean and standard deviation of capture after training.

Mean	Standard Deviation
249.0000	214.0179

One should also observe how large the standard deviations in tables 1 and 2 are in comparison with the mean values. The reason for this is the stochastic nature of the game, especially the choices of $\rho_1 = \rho_2 = 0.8$. If we choose this number to be 1.0 the standard deviation decreases substantially. However, we think that leaving it 0.8 brings some good features to the system, since it forces the pursuers to work together besides simulating delays and noise in their commands.

7 Conclusion

In this paper we have shown how a pursuit-evasion game may be modeled with Markov chains. We have also shown that using a learning automata will lead to the optimal solution of the finite time/finite state game with incomplete information.

Simulations were provided in order to show the feasibility of the approach. It is shown that the learning phase leads to a considerable improvement in the response of the game when compared to the initial configuration of policies. Moreover, it is shown that players can learn to cooperate even when they optimize only their own cost function. However, it must be emphasized that the optimization achieved is not the optimal group optimization. This would only be accomplished it there was a global cost function to be optimized and if full information were assumed, which is not the case.

One drawback of the technique is that convergence to the optimal can only be proved if the player have full representation of all possible states of the system. As either the number of players or the size of the grid increases, the calculation would become too cumbersome. In the future, we intend to use a fuzzy representation of the environment in order to reduce the complexity of the game. However, this would make it impossible to prove a theoretical convergence to the optimal policy.

References

Basar, T. and Olsder, G. J., 1999, Dynamic Noncooperative Game Theory, SIAM, Philadelphia, 2 edition.

Bertsekas, D. P., 1995, Dynamic Programming and Optimal Control, Athena Scientific, Belmont, MA.

Conlisk, J., 1993, Adaptation in games: two solutions to the Crawford puzzle, Journal of Economic Behaviour and Organization, 22, 25–50.

Derman, C., 1970, Finite State Markovian Decision Processes, Academic Press, New York, New York.

Filar, J. and Vrieze, K., 1997, Competitive Markov Decision Processes, Springer-Verlag, New York, New York.

Fudenberg, D. and Levine, D. K., 1998, The Theory of Learning in Games, The MIT Press, Cambridge, Massachusetts.

Givigi, S. N., 2009, Analysis and Design of Swarm-Based Robots Using Game Theory, Ph.D. thesis, Carleton University.

Givigi, S. N., Schwartz, H. M., and Lu, X., 2009, A Reinforcement Learning Adaptive Fuzzy Controller for Differential Games, Journal of Intelligent and Robotic Systems.

Harmon, M. E., III, L. C. B., and Klopf, A. H., 1995, Reinforcement Learning Applied to a Differential Game, Adaptive Behavior, 4, 1, 3–28.

Hespanha, J. P. and Prandini, M., 2001, Nash Equilibria in Partial-Information Games on Markov Chains, in 40th IEEE Conference on Decision and Control, volume 3, 2102–2107.

Hespanha, J. P., Prandini, M., and Sastry, S., 2000, Probabilistic Pursuit-Evasion Games: A One-Step Nash Approach, in 39th IEEE Conference on Decision and Control, 2272–2277.

Hofbauer, J. and Sigmund, K., 2003, Evolutionary game dynamics, Bulletin of the American Mathematical Society, 40, 4, 479–519.

Howard, R. A., 1960, Dynamic Programming and Markov Processes, M. I. T. Press, Cambridge, MA.

Isaacs, R., 1965, Differential Games: A Mathematical Theory with Applications to Warfare and Pursuit, Control and Optimization, John Wiley and Sons, Inc., New York, New York.

Kushner, H. J. and Yin, G. G., 1997, Stochastic Approximation Algorithms and Applications, Springer-Verlag, New York, New York.

Li, D. and Cruz, J. B., 2006, Improvement with Look-ahead on Cooperative Pursuit Games, in 45th IEEE Conference on Decision and Control, 4313–4318.

Littman, M. L., 2001, Value-function reinforcement learning in Markov games, Journal of Cognitive Systems Research, 2, 55–66.

Myerson, R. B., 1991, Game Theory: Analysis of Conflict, Harvard University Press, Cambridge, Massachusetts.

Nash, J. F., 1951, Noncooperative Games, Annals of Mathematics, 54, 2, 289–295.

Norman, M. F., 1972, Markov Processes and Learning Models, Academic Press, New York, New York.

Posnyak, A. S. and Najim, K., 1997, Learning Automata and Stochastic Optimization, Springer, Berlin; New York.

Sastry, P. S., Phansalkar, V. V., and Thathachar, M. A. L., 1994, Decentralized Learning of Nash Equilibria in Multi-Person Stochastic Games With Incomplete Information, IEEE Transactions on Systems, Man, and Cybernetics, 24, 5, 769–777.

Shapley, L. S., 1953, Stochastic Games, Proc. Nat. Acad. Sci. U.S.A., 39, 1095–1100.

Starr, A. W. and Ho, Y. C., 1969, Nonzero-Sum Differential Games, Journal of Optimization Theory and Applications, 3, 3, 184–206.

Suppes, P. and Atkinson, R. C., 1960, Markov Learning Models For Multiperson Interactions, Stanford University Press, Stanford, California.

Thathachar, M. A. L. and Sastry, P. S., 2004, Networks of Learning Automata: Techniques for Online Stochastic Optimization, Kluwer Academic Publishers, Boston, Massachusetts.

van der Wal, J., 1980, Stochastic Dynamic Programming, Ph.D. thesis, Technische Hogeschool Eindhoven.

von Neumann, J. and Morgenstern, O., 1947, The Theory of Games and Economic Behavior, Princeton University Press, Princeton, 2nd edition.

Vrancx, P., Verbeeck, K., and Nowe, A., 2008, Decentralized Learning in Markov Games, IEEE Transactions on Systems, Man, and Cybernetics - Part B, 38, 4, 976–981.

Weibull, J. W., 1995, Evolutionary Game Theory, MIT Press, Cambridge, Massachusetts.

Wheeler, R. M. and Narendra, K. S., 1986, Decentralized Learning in Finite Markov Chains, IEEE Transactions on Automatic Control, 31, 6, 519–526.

Yeung, D. W. K. and Petrosyan, L. A., 2006, Cooperative Stochastic Differential Games, Springer Science+Business Media, Inc., New York, NY.