An Application of Multiagent Learning in Highly Dynamic Environments

Kyle Hollins Wray
University of Massachusetts, Amherst
Amherst, MA 01003, USA
wray@cs.umass.edu

Benjamin B. Thompson
The Pennsylvania State University
State College, PA 16802 USA
bbt10@psu.edu

Abstract

We explore the emergent behavior of game theoretic algorithms in a highly dynamic applied setting in which the optimal goal for the agents is constantly changing. Our focus is on a variant of the traditional predator-prey problem entitled Defender. Consisting of multiple predators and multiple prey, Defender shares similarities with rugby, soccer, and football, in addition to current problems in the field of Multiagent Systems (MAS). Observations, communications, and knowledge about the world-state are designed to be information-sparse, modeling real-world uncertainty. We propose a solution to Defender by means of the well-known multiagent learning algorithm fictitious play, and compare it with rational learning, regret matching, minimax regret, and a simple greedy strategy. We provide the modifications required to build these agents and state the implications of their application of them to our problem. We show fictitious play's performance to be superior at evenly assigning predators to prey in spite of it being an incomplete and imperfect information game that is continually changing its dimension and payoff. Interestingly, its performance is attributed to a synthesis of fictitious play, partial observability, and an anti-coordination game which reinforces the payoff of actions that were previously taken.

Introduction

Multiagent Systems (MAS) research has recently began to focus on the adaptability of agents from dynamism and the managing of uncertainty as they work in groups to meet their design objectives. Ad hoc teams (Stone et al. 2010; Barrett, Stone, and Kraus 2011), robotic incarnations of MAS (Kitano et al. 1997; Stone 2000; Vidal et al. 2002), and mathematical models including partial observability (e.g., Dec-POMDPs) (Bernstein, Zilberstein, and Immerman 2002; Goldman and Zilberstein 2004; Nair and Tambe 2005) all describe aspects of dynamic scenarios. Most solutions can overcome high degrees of uncertainty by relying on strong but understandably problem-specific solutions (Undeger and Polat 2010; Vidal et al. 2002; Stone 2000) or robust auction approaches (Smith 1980; Parker 1998). These methods do not usually use traditional game theory in their construction since the mathematical model does not incorporate all realistic aspects of uncertainty. Conversely, solutions under game theory are often evaluated in well-defined problems (Jafari et al. 2001) or only incorporate some dynamism (Boutilier 1996). This paper investigates this relatively unexplored intersection of traditional game theory and highly dynamic MAS scenarios. Specifically, we examine the direct application of game theory to an information-sparse predator-prey problem entitled Defender.

Anti-coordination games only award agents for selecting an action different than any other player. We leverage this property to evenly distribute predators (agents) to prey (actions). Casting the problem with this game theoretic structure enables us to directly employ adaptive learning algorithms such as fictitious play and regret matching. The problem domain, which adjusts payoffs over time and has both incomplete and imperfect information, complicates this learning process. Agents do not observe all other predators (agents) or even prey (actions). This dynamically shifts as they move in the environment, causing the internal game's payoffs and dimension to change. Moreover, agents must successfully interact without any prior coordination or direct communication; they do not even know how many predators or prey there are in the environment.

This paper empirically investigates the effects of this dynamism on the aforementioned game theoretic learning algorithms in an applied setting. Our contributions include empirical evidence that suggests the direct application game theoretic learning can be successfully applied even with a changing game. We also provide the Defender problem itself, an extension of predator-prey, as a framework for the development of multiagent learning algorithms in dynamic scenarios. Furthermore, we show that emergent commitment and adaptability behavior arises with a combination of anti-coordination games and these well-known learning algorithms.

The next section introduces the formal definition of Defender and our rationale behind its construction. We then state the design of our agents. With the model in place we present and interpret our results from experimentation.

The Defender Problem

The predator-prey problem, originally proposed by Benda *et al.* (1986), has become a standard proving ground for MAS research over the years since its inception. We propose a multi-predator multi-prey version of the original problem

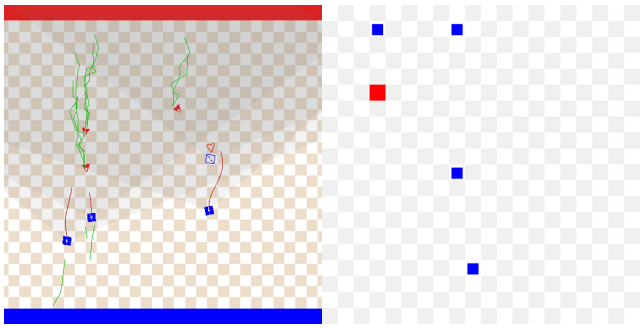

Figure 1: A visual example to compare Defender (left) to the traditional predator-prey problem (right). Blue shapes denote predators and red shapes denote prey.

that situates predators and prey on opposite sides of a region (see Figure 1). The goal of the predators is to successfully capture the prey before one of the prey reaches their haven behind the predators. The construction of Defender was inspired by the aforementioned problems in the state-of-the-art and other variants (Hespanha, Kim, and Sastry 1999; Arslan, Marden, and Shamma 2007; Stone et al. 2010; Barrett, Stone, and Kraus 2011; Chung, Hollinger, and Isler 2011; Keshmiri and Payandeh 2013). Durfee *et al.*'s (1989) work influenced Defender's design. They proposed a discrete multi-predator multi-prey problem with limited field-of-view and update rate, which differs from our continuous version which requires a unique assignment of predators to prey. Defender's design incorporates the uncertainty found in realistic applications in which agents must rapidly converge to a solution while still retaining the ability to adapt to an ever-changing environment.

Defender was also inspired by a predator-prey variant that incorporates the kind of dynamism found in realistic incarnations of MAS (Parker and Emmons 1997; Tomlin, Lygeros, and Sastry 2000). Additionally, we took inspiration work which investigated a predator(s) trying to capture prey in worlds with obstacles (Undeger and Polat 2010; Gerkey, Thrun, and Gordon 2006). Lastly, we built upon similar research on multiagent assignment problems which used game theory (Arslan, Marden, and Shamma 2007).

Defender consists of a set of predators N, and a set of prey M. These agents interact in an environment which operates under an agent position update rule. Each of these components is described in the following subsections.

Agents

All agents $i \in N \cup M$ in Defender have a state described by the tuple $(\vec{p}_i, \vec{v}_i, \vec{a}_i, \theta_i, \Omega_i, L_i)$. There are three vectors that define the position, velocity and acceleration of the agent: $\vec{p}_i$, $\vec{v}_i$, and $\vec{a}_i$, respectively. The orientation of the agent is defined by the scalar variable $\theta_i \in [0, 2\pi)$. Any observations made by the agent over all stages are given in the list Ω_i. We will denote Ω_{ij} to be the list of observations that agent i made over all stages of agent $j \in N \cup M \setminus \{i\}$, which the tracker internally manages within each predator. Ω_{ij} will only contain the positions, velocities, and orientations of agent j at each stage. We denote a particular observation at a stage to be ω_{ij}, e.g., $\Omega_{ij} = \{\omega_{ij}^1, \ldots, \omega_{ij}^t\}$. L_i refers to the particular learning algorithm used by agent i. A predator $i \in N$ follows fictitious play, rational learning, regret matching, minimax regret, or a simple greedy strategy. A prey $i \in M$ has either a simple or maneuvering strategy. Lastly, we will use $\vec{v}_{i,des}$ and $\theta_{i,des}$ as the variables the agent may control to adjust its state in the world.

Environmental Parameters

The environment is defined by the tuple $(s^*, c^*, t^*, \phi^*, (\delta_p, \delta_v, \delta_\phi), (p_{miss}, p_{err}), (a_{vmax}, a_{tmax}, a_{update}))$. The area in which our agents reside is a two-dimensional, continuous world of size s^* by s^*. Each agent team begins opposite the other on their "goal" side; each goal is of size s^* by $\frac{1}{15}s^*$. First, let $\mathcal{U}(a, b)$ denote a uniform distribution on $[a, b]$. Formally, for all predators $i \in N$, $\vec{p}_i = [\mathcal{U}(0, s^*), \mathcal{U}(0, \frac{1}{15}s^*)]^T$ and $\theta_i = \mathcal{U}(-\phi^*, \phi^*)$, with ϕ^* denoting the field-of-view. For all prey $j \in M$, $\vec{p}_j = [\mathcal{U}(0, s^*), \mathcal{U}(\frac{14}{15}s^*, s^*)]^T$ and $\theta_j = \mathcal{U}(-\phi^*, \phi^*)$. Agents enter play after being inactive for $\mathcal{U}(0, t^*)$ seconds.

For predator $i \in N$ to capture a prey $j \in M$, the predator must be within c^* units of the prey, i.e., $\|\vec{p}_i - \vec{p}_j\|_2 < c^*$. Upon capture, the predator and prey pair are halted and removed from the game. A predator may only capture one prey. All agents are limited to a field-of-view ϕ^*, which may be described for any agent $i \in N \cup M$ by the orientation interval $(\theta_i - \frac{\phi^*}{2}, \theta_i + \frac{\phi^*}{2})$.

Observations made by agents are subject to sensor noise, which we model using a normal distribution denoted as $\mathcal{N}$. Given $i, j \in N \cup M$, $i \neq j$, if agent i observes j, with its position being inside i's field-of-view, the new observation i makes of j, $\omega_{ij} = (\vec{p}_{ij}, \vec{v}_{ij}, \theta_{ij})$, is subject to noise such that i observes $\vec{p}_{ij} = \vec{p}_j + [\mathcal{N}(0, \delta_p), \mathcal{N}(0, \delta_p)]^T$, $\vec{v}_{ij} = \vec{v}_j + [\mathcal{N}(0, \delta_v), \mathcal{N}(0, \delta_v)]^T$, and $\theta_{ij} = \theta_j + \mathcal{N}(0, \delta_\phi)$. We use p_{miss} and p_{err} as the probability that any agent completely misses an observation, and the probability that the observation is erroneous, respectively. The variables a_{vmax} and a_{tmax} define the maximal agent movement speed and turn speed, respectively. Agents only update their decisions via their learning rule L_i after every a_{update} seconds, restricting the agent update rate to emulate the realistic computational limitations of robotic MAS.

Agents have limitations on their capability to store information such that they do not preserve any information over repeated trials of Defender. Also, agents are only able to infer the actions that others take; no mechanism exists which lets them to directly communicate their selected actions. Agents are not provided with any initial knowledge about any trial *a priori*. This means that the number of agents in the system is not known. For observed agents, the learning algorithms (including payoffs) of others is also unknown.

The State Transition

The movement of the agents follows the dynamics found in robotic applications. With this in mind, we define a friction constant $\mathcal{F}$ and an acceleration rate for agents $\mathcal{R}$. For our

purposes, we set $\mathcal{F} = 0.96$ and $\mathcal{R} = 0.75$ since we found it provided a good balance of movement uncertainty while still retaining realistic behavior. Any agent $i \in N \cup M$ adjusts its desired velocity $\vec{v}_{i,des}$ and desired orientation $\theta_{i,des}$ only during their update stage. The state update for an agent $i \in N \cup M$ is as follows.

$$\vec{v}_i \leftarrow \vec{v}_i + \vec{a}_i$$
$$\vec{p}_i \leftarrow \vec{p}_i + \vec{v}_i + \vec{a}_i^2/2$$
$$\vec{v}_i \leftarrow \mathcal{F} \cdot \vec{v}_i$$

$$\theta_i \leftarrow \theta_i + \max(-\theta^*/2, \min(\theta^*/2, \theta_{i,des} - \theta_i))$$
$$\vec{a}_i \leftarrow \vec{a}_i + sign(\|\vec{v}_{i,des}\| - \|\vec{v}_i\|) \cdot \mathcal{R} \cdot [\cos\theta_i, \sin\theta_i]^T$$

In our simulation environment, we also use the state transition to check for the various conditions embedded in the problem. These include the checks for if a predator captured a prey and the game ending conditions: a prey reached its haven behind the predators or all prey were successfully captured by the predators.

A Game Theoretic Approach

We will use game theoretic learning algorithms that adapt based on the actions of others and/or the payoffs obtained at each stage. The three learning algorithms that we evaluate are fictitious play, rational learning, and regret matching. These were selected because of their convergence in traditional game theory, inherent adaptability, optimizable structure in anti-coordination games, and the important property that they do *not* require knowledge of other agents' payoffs. We use minimax regret and a simple greedy strategy as a baseline comparison; both of these also do not require the knowledge of others' payoffs. There have been few applications of these algorithms to solve problems that are not perfectly known and remain static over iteration. This is to be expected, since they are constructed to converge only in these controlled domains. Knowing this fact, we explore their usefulness in an applied setting, which does not satisfy the assumed structure of the formal model.

In addition to the learning algorithms, we use an *anti-coordination game*. Informally, players in these games are only rewarded a positive payoff if they successfully select a different action from another player, and a zero payoff otherwise. Our predators and prey will be the players and actions in this game, respectively. Our payoff function will be proportional to the expected distance a predator must travel to capture a prey, given both agents' trajectories. We will show that this game structure, in combination with our learning algorithms, creates an emergent behavior for predators to distribute themselves to prey.

This section details our predator design and overall methodology. We use the standard game theoretic definitions and learning algorithms (Shoham and Leyton-Brown 2009).

Game Definition

As mentioned above, we will use an anti-coordination game with payoffs proportional to expected travel distance. We begin our discussion by setting notation in Defender's "normal form" (Definition 1).

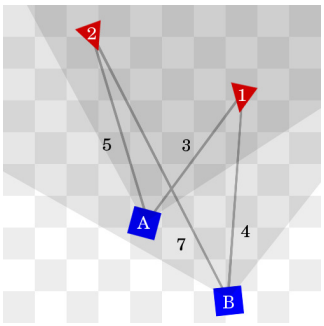

Figure 2: A simplified example of two predators and two prey. The set of predators is $N = \{A, B\}$ and the set of prey is $M = \{1, 2\}$, shown in blue and red, respectively. The observations for predator A are $\Omega_A = \{\omega_{A1}, \omega_{A2}\}$ and for predator B are $\Omega_B = \{\omega_{BA}, \omega_{B1}, \omega_{B2}\}$. The path numbers denote the expected travel distance.

Definition 1. Defender's normal form *(at any stage t) is a tuple (N, A, U):*

- *N is the set of n predators.*
- *$A = (A_1, \ldots, A_n)$ with each $A_i = \{a_1, \ldots, a_{m_i}\}$, $m_i \leq n$, being a set of prey (actions) available to predator i as defined by observations within i's field-of-view.*
- *$U = (u_1, \ldots, u_n)$ is a set of utility functions; defined for a predator i, $u_i(a_i, a_{-i})$ is the utility for an action profile $a = (a_i, a_{-i}) \in A$.*

Equation 1 below defines our payoff function. The payoff value is computed using equation 2.

$$u_i(a) = \begin{cases} u_i(a_i), & \text{if } \forall 1 \leq j \leq n, j \neq i, a_i \neq a_j \\ 0, & \text{otherwise} \end{cases} \quad (1)$$

$$u_i(a_i) = \max(0, \min(p_{max}, p_{max} - d \cdot \gamma)) \quad (2)$$
$$\gamma = \max(0.7, \min(1.0, 0.85 + 0.05 \cdot (\beta - \alpha)))$$

The variables used in equation 2 are detailed below.

- p_{max} is a constant used to ensure the payoff is positive. It is defined as the maximal distance the agent may travel.
- d is an expected travel distance estimate for the predator to capture the prey. It is computed using an iterative solver based on the known capabilities of the predator and the observed trajectory of the prey.
- β is the number of observed predators who have shorter distance estimates. This performs the same estimated travel distance function as above except that it is centered at the observed predators.
- α is the number of additional potential prey that might exist if a miss occurs, which again uses the estimated travel distance function to count the number of visible prey after reaching the estimated path intersection point.

General anti-coordination games ensure that a set of agents selecting the same action awards a zero payoff to all within the set. Ideally, all components of the game are

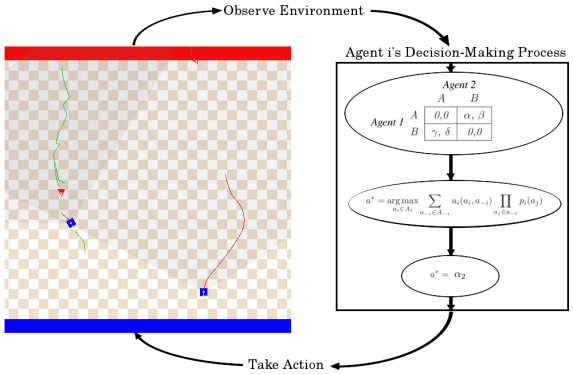

Figure 3: Predator visual control flow diagram.

known; however, predators do not know of other agents outside their field-of-view. Therefore, the "global view" of the anti-coordination game being played by all agents at any given stage is not truly an anti-coordination game. A simple example of this is depicted in Figure 2. Predator A can see prey 1 and 2, but is unable to see predator B. Predator B can see prey 1, prey 2, and predator A. By default, since no other predators are observed from predator A's perspective, it will select the action that yields the highest utility: prey 1. If predator A continues to select prey 1, then our learning algorithms will hopefully cause predator B to select prey 2, since prey 1 would provide a zero payoff to predator B. This means our agent's limited field-of-view and their tracker capabilities play an important role in determining the game's stability (in the sense of its dimension and payoff consistency over stages). There are other issues, e.g., the game's payoffs changing over time, and the actual dimension of the game (number of players and actions) changes as agents move around the area. This will be discussed in detail later.

Learning Algorithms

Agents update their decisions asynchronously, i.e., agents update in a continuous time, not together in discrete stages. At each update, they first make observations of the environment, then update their tracker accordingly. With this new information, they compute their next action following their learning algorithm, which may update its internal variables given this new information as well. For a visual explanation of the agents' run-time operation, see the visual flow diagram depicted in Figure 3. If predators do not observe other predators, it follows the greedy strategy by default below.

Fictitious play was originally proposed by Brown (1951) and Robinson (1951) as a means to compute Nash equilibria. Many notable variants exist to improve convergence properties such as *smooth fictitious play* by Fudenberg *et al.* (1999). The underlying assumption is that others are performing a stationary mixed strategy. The empirical frequencies over observations determine the probability distributions over actions for each other player. As others have done, we define it with an expected utility equation (Arslan, Marden, and Shamma 2007). We use the action-based version because it provides the desired emergent commitment behavior, which will be discussed later (Equation 3).

$$L_i = \arg\max_{a_i \in A_i} \sum_{a_{-i} \in A_{-i}} u_i(a_i, a_{-i}) \prod_{a_j \in a_{-i}} p_i(a_j) \qquad (3)$$

$$p_i(a_j) = \frac{1}{|\Omega_{ij}|} \sum_{\omega \in \Omega_{ij}} \mathcal{I}\{\omega = a_j\}$$

Rational learning follows the expected utility representation of fictitious play; however, it instead defines the probability distribution of other players' actions using Bayesian inference (Equation 4). While any prior and likelihood distributions are valid, we will update using the Dirichlet distribution's expected value, similar to Boutilier's implementation (1996), with an initial prior distribution equivalent to the discrete uniform.

$$L_i = \arg\max_{a_i \in A_i} \sum_{a_{-i} \in A_{-i}} u_i(a_i, a_{-i}) \prod_{a_j \in a_{-i}} p_i(a_j) \qquad (4)$$

$$p_i(a_j) = \frac{p(\Omega_{ij}|a_j)p(a_j)}{\sum_{a_k \in a_{-i}} p(\Omega_{ik}|a_k)p(a_k)}$$

Regret matching and other regret-based methods have become popular in the multiagent learning community due its logical rationale and powerful convergence properties (Equation 5). In summary, the algorithm guarantees the proportion of times each player selects an action to converge to a correlated equilibrium. The regret that player $i \in N$ receives for an action $a_i \in A_i$ is defined as the average payoff i could have received if it had just played a_i at all stages minus the average payoff i has received. It selects the action randomly following the distribution, denoted by the "$\sim$" symbol for brevity. Below, we also denote $a_i^{*(t)}$ to be the action performed by i at stage t.

$$L_i \sim \frac{\max\{r_i(a_i) - r_i, 0\}}{\sum_{a' \in A_i} \max\{r_i(a') - r_i, 0\}}, \quad \forall a_i \in A_i \qquad (5)$$

$$r_i(a') = \frac{1}{|\Omega_{ij}|} \sum_{t=1}^{|\Omega_{ij}|} u_i^{(t)}(a'), \quad r_i = \frac{1}{|\Omega_{ij}|} \sum_{t=1}^{|\Omega_{ij}|} u_i^{(t)}(a_i^{*(t)})$$

Minimax regret uses regret only on a single stage to select the action that minimizes the maximum regret. We selected this as one of our baselines because it, like all the other algorithms here, does not require the knowledge of other agents' payoffs. Furthermore, it uses the traditional minimax logic, it was constructed from the field of game theory, and it shows the behavior of a stage-based algorithm that does not incorporate observation history (Equation 6).

$$L_i = \arg\min_{a_i \in A_i} \max_{a_{-i} \in A_{-i}} \left(\max_{a_i' \in A_i} u_i(a_i', a_{-i}) - u_i(a_i, a_{-i}) \right) \qquad (6)$$

Greedy algorithms are used throughout the literature to provide a simple baseline. Ours simply selects the observable prey with the highest utility (Equation 7).

$$L_i = \arg\max_{a_i \in A_i} u_i(a_i) \qquad (7)$$

The Tracker

Robotic MAS often must employ the use of a tracker which enables robotic agents to keep track of other agents in the area. We use a standard score and threshold gating mechanism. Our score is proportional to the difference from the prediction and the new observation state (x, y, θ, and s). We employ an Extended Kalman Filter (EKF) over the observation state space to reduce noise (Welch and Bishop 2006).

Experimentation

Our experimentation focuses on $n = m$ scenarios, i.e., the same number of predators and prey. We experimented with heterogeneous team sizes and found that the homogeneous cases produced the experimentally interesting results. It is clear why this is the case; if the predators outnumber the prey, it is easier for the predators to completely capture all the prey. Conversely, if the prey outnumber the predators, there is no possible way to capture all the prey in the system since a predator may only capture one prey. Evenly matched teams, like in sports, provide the most distinctive results. Combined with strict space constraints, we have omitted this subset of the experiments in favor of highlighting the most demonstrative experiments to compare the algorithms.

Instead, we examine different team sizes, varying the number of predators and prey from 1 to 8. We experiment with all permutations of the five predator learning algorithms against two kinds of prey: simple and maneuvering. This provides 80 points of comparison. Note that we were unable to run minimax regret agents beyond team sizes of 4 due to computational issues. We ran each of the 80 configurations for 5000 trials using a 3.2 GHz Intel(R) Pentium(R) 4 CPU with 1 GB of RAM under OpenSuse Linux 11.4. The simulation itself was written in C++ using OpenGL.

Simple prey sprint toward their haven given their initial orientation. This provides a straight forward baseline. In practice, it is actually quite difficult to coordinate the predators even in this simple case. *Maneuvering prey* are initialized with a random wavelength λ on the interval $\mathcal{U}(4, 9)$ and a periodicity γ on the interval $\mathcal{U}(tan(\frac{\pi}{12})\lambda, tan(\frac{5\pi}{12})\lambda)$, i.e., 15 to 75 degree arcs that are 4 to 9 units long.

For all of our experiments, we used the environmental parameters $E = (s^*, c^*, t^*, \phi^*, (\delta_{loc}, \delta_{vel}, \delta_\phi), (p_{miss}, p_{err}), (a_{vmax}, a_{tmax}, a_{update})) = (\ 100$ units, 3.3 units, 10 seconds, $\frac{\pi}{3}$, (1 unit, 3 units, 0.01 units), (5%, 5%), (0.3 units, 0.75 units, $\frac{1}{2}$ second)). This configuration provides robotic-like movement, e.g., UAVs or four-wheeled robots, and emulates realistic sensing limitations (see Figure 4).

Discussion

In spite of the high degree of uncertainty in Defender, we found fictitious play and rational learning produced an emergent distributive and commitment behavior. The percentage of prey captured for all three learning algorithms were high, ranging between 98% and 80% over team sizes from 1 to 8. While maneuvering prey consistently reduced the capture percentage of all algorithms by about 2% to 3%, they have proportionally similar prey captured percentages.

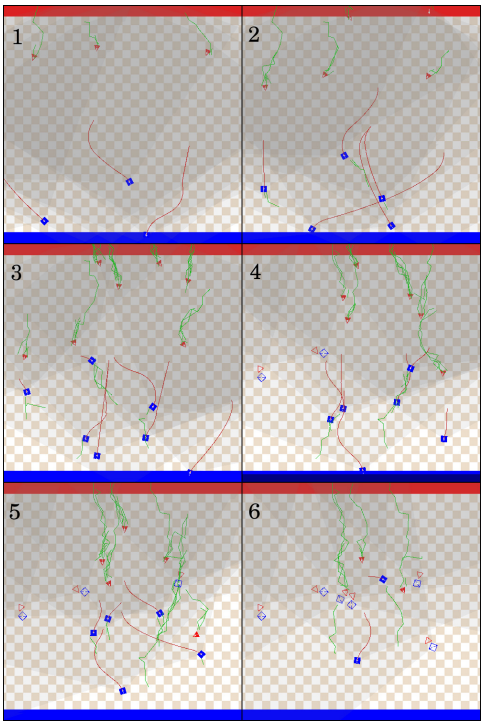

Figure 4: An example of fictitious play predators chasing maneuvering prey ($n = m = 8$). The numbers denote the sequence over time.

The disparity in the complete capture percentages, however, suggests that fictitious play produces much more cooperative behavior than the others. In the case with team sizes of eight, fictitious play reached 39.64% as compared to 27.42%, 12.94%, and 11.00% for rational learning, regret matching, and the greedy strategy, respectively. All averages show a definitively small 95% confidence interval, demonstrating that the averages are statistically distinct.

Regret matching surprisingly did not perform that well in practice. The main problem is that the agents' *strategy profiles* are not proven to converge, like they do in fictitious play or rational learning. Instead, it is the *empirical average* of action selections that converges to a correlated equilibrium. In this application, the predators' tended to switch back and forth between prey. Our regret-based agents lacked commitment which, as noted by Jennings (Jennings 1993), is one of the most important aspects of cooperating MAS. This has been observed by other researchers before (Jafari et al. 2001), and should be taken into consideration when designing agents that must solve their problem *in situ*.

The learning predators performed quite well compared to the baselines, reaching 80% to 90% in the case of team sizes set to eight using simple prey. Minimax regret had computation issues, suffered from the aforementioned regret-based issues, and was not designed to incorporate any history into its decision-making process, thus it performed poorly at 47.28% in the much simpler case with a team size of four. Our greedy algorithm performed adequately because it pos-

	Simple							
	1	2	3	4	5	6	7	8
Fictitious Play	97.0 ± 0.5	95.8 ± 0.4	94.0 ± 0.4	92.3 ± 0.4	92.0 ± 0.3	90.7 ± 0.3	90.3 ± 0.3	90.2 ± 0.3
Rational Learning	98.1 ± 0.4	93.8 ± 0.5	91.3 ± 0.4	89.2 ± 0.4	88.6 ± 0.4	88.1 ± 0.3	87.8 ± 0.3	87.6 ± 0.3
Regret Matching	98.2 ± 0.4	93.3 ± 0.5	88.6 ± 0.5	85.1 ± 0.5	83.0 ± 0.4	81.8 ± 0.4	80.6 ± 0.4	80.1 ± 0.3
Minimax Regret	97.6 ± 0.4	57.9 ± 1.0	50.4 ± 0.8	47.3 ± 0.7	n/a	n/a	n/a	n/a
Greedy	97.2 ± 0.5	87.8 ± 0.6	83.3 ± 0.5	82.6 ± 0.4	81.3 ± 0.4	81.2 ± 0.3	81.1 ± 0.3	81.3 ± 0.3
	Maneuvering							
Fictitious Play	97.0 ± 0.5	93.6 ± 0.5	92.2 ± 0.4	90.2 ± 0.4	88.9 ± 0.4	88.4 ± 0.3	87.7 ± 0.3	87.4 ± 0.3
Rational Learning	96.0 ± 0.5	91.8 ± 0.5	88.6 ± 0.5	87.0 ± 0.4	85.9 ± 0.4	85.3 ± 0.4	85.3 ± 0.3	85.1 ± 0.3
Regret Matching	97.7 ± 0.4	90.8 ± 0.6	85.8 ± 0.5	82.7 ± 0.5	80.1 ± 0.5	78.8 ± 0.4	78.1 ± 0.4	77.3 ± 0.4
Minimax Regret	95.3 ± 0.6	66.1 ± 0.9	56.4 ± 0.8	52.8 ± 0.7	n/a	n/a	n/a	n/a
Greedy	95.1 ± 0.6	87.5 ± 0.6	82.9 ± 0.5	80.9 ± 0.4	81.1 ± 0.4	80.6 ± 0.4	80.4 ± 0.3	80.5 ± 0.3

Table 1: Average percentage of prey captured ($\%$) for each configuration of predators and prey with even teams sizes of 1 to 8 agent. The adjustment from the mean, defining a 95% confidence interval, is also provided. Each cell summarizes 5000 trials.

	Simple							
	1	2	3	4	5	6	7	8
Fictitious Play	97.0 ± 0.5	97.7 ± 0.8	82.4 ± 1.1	71.8 ± 1.2	64.3 ± 1.3	54.1 ± 1.4	45.2 ± 1.4	39.6 ± 1.4
Rational Learning	98.1 ± 0.4	87.6 ± 0.9	74.7 ± 1.2	60.0 ± 1.4	50.5 ± 1.4	41.1 ± 1.4	33.7 ± 1.3	27.4 ± 1.2
Regret Matching	98.2 ± 0.4	86.8 ± 0.9	68.4 ± 1.3	50.6 ± 1.4	36.5 ± 1.3	26.4 ± 1.2	18.2 ± 1.1	12.9 ± 0.9
Minimax Regret	97.6 ± 0.4	35.7 ± 1.3	11.2 ± 0.9	04.7 ± 0.6	n/a	n/a	n/a	n/a
Greedy	97.2 ± 0.5	75.7 ± 1.2	51.5 ± 1.4	39.5 ± 1.4	26.5 ± 1.2	19.4 ± 1.1	14.4 ± 1.0	11.0 ± 0.9
	Maneuvering							
Fictitious Play	97.0 ± 0.5	87.5 ± 0.9	78.1 ± 1.1	65.0 ± 1.3	53.8 ± 1.4	45.6 ± 1.4	36.6 ± 1.3	30.3 ± 1.3
Rational Learning	96.0 ± 0.5	84.0 ± 1.0	68.3 ± 1.3	54.5 ± 1.4	42.4 ± 1.4	33.6 ± 1.3	26.5 ± 1.2	21.3 ± 1.1
Regret Matching	97.7 ± 0.4	82.4 ± 1.1	62.0 ± 1.3	44.8 ± 1.4	30.1 ± 1.3	19.4 ± 1.1	14.3 ± 1.0	08.8 ± 0.8
Minimax Regret	95.3 ± 0.6	42.2 ± 1.4	15.8 ± 1.0	06.8 ± 0.7	n/a	n/a	n/a	n/a
Greedy	95.1 ± 0.6	75.0 ± 1.2	51.2 ± 1.4	34.5 ± 1.3	25.7 ± 1.2	18.8 ± 1.1	13.3 ± 0.9	10.0 ± 0.8

Table 2: Average complete capture percentages ($\%$) for each configuration of predators and prey with even teams sizes of 1 to 8 agents. The adjustment from the mean, defining a 95% confidence interval, is also provided. Each cell summarizes 5000 trials.

sessed high commitment behavior, but almost no cooperative behavior. In comparison, fictitious play predators would also have a positive-feedback loop; however, this is dampened by the probability that other predators might also chase the prey. These two juxtaposed components are the key to their success. Each component vies to tip the scale in favor of either chasing a nearby prey or avoiding it with the understanding that another predator will capture it instead.

Lastly, the game structure in Defender changes over time. The payoffs change based on the locations of the predators and prey; the number of players and actions internal to each predator is also dynamic. Even though the initial and final games are different, the change is gradual, and the algorithms can keep up as the system evolves. We experimented with the update rates of the algorithms, which logarithmically improve the capture rates. Unfortunately, space constraints prevent us from including the graph. Addressing this "moving the goalposts" problem is an important topic for the future of cooperative multiagent learning (Panait and Luke 2005). Due to the limited mobility of the predators, the field-of-view encompasses *most* (but not all) of the relevant information for deciding which prey to chase. While there are situations for which an optimal decision requires non-observed information, it is an unavoidable issue without direct communication. Additionally, without direct communication or complete information, the agents do not know the actions of others; the actions are inferred. Predators are also unable to correlate past observations of an agent with future ones, given that the agent has left and re-entered the field-of-view. This dynamism greatly affects the outcome of the game, but it is acceptable if we assume an agent's sensing capabilities encompass enough pertinent information to approximately solve its local optimization problem.

Conclusion

We have examined the direct application of game theoretic learning algorithms to a highly information-sparse predator-prey problem entitled Defender. This problem was designed to incorporate many of the issues faced in realistic multiagent systems. With the design objective to distribute predators to prey, we leveraged the particular structure of a special anti-coordination game, with a utility proportional to expected travel distance. We investigated fictitious play, rational learning, regret matching, minimax regret, and greedy predators against simple and maneuvering prey. The implementation of Defender considered all these combinations of the predators and prey ranging team sizes from one to eight. Our results concluded that fictitious play performed the best overall, creating an emergent commitment and coordination behavior. We have shown that game theory can be used to create a rapid cooperative and emergent commitment behavior while still remaining adaptive to a dynamic environment.

Acknowledgments

This material is based upon work supported by The Office of Naval Research (ONR) through The Naval Sea Systems Command under Contract No. N00024-02-D-6604. The authors would like to thank the anonymous reviewers for their feedback, as well as both ONR and The Pennsylvania State University's Applied Research Laboratory for their support.

References

Arslan, G.; Marden, J.; and Shamma, J. 2007. Autonomous vehicle-target assignment: A game theoretical formulation. *ASME Journal of Dynamic Systems, Measurement, and Control* 129:584–596.

Barrett, S.; Stone, P.; and Kraus, S. 2011. Empirical evaluation of ad hoc teamwork in the pursuit domain. In *Proceedings of Tenth International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2011)*.

Benda, M.; Jagannathan, V.; and Dodhiawalla, R. 1986. On optimal cooperation of knowledge sources - an empirical investigation. Technical Report BCS-G2010-28, Boeing Advanced Technology Center, Boeing Computing Services, Seattle, WA.

Bernstein, D.; Zilberstein, S.; and Immerman, N. 2002. The complexity of decentralized control of markov decision processes. In *Proceedings of the Sixteenth Conference on Uncertainty in Artificial Intelligence*, 32–37.

Boutilier, C. 1996. Learning conventions in multiagent stochastic domains using likelihood estimates. *Uncertainty in Artificial Intelligence* 106–114.

Brown, G. 1951. Iterative solutions of games by fictitious play. *Activity Analysis of Production and Allocation* 374–376.

Chung, T.; Hollinger, G.; and Isler, V. 2011. Search and pursuit-evasion in mobile robotics. *Autonomous Robots* 31(4):299–316.

Durfee, E., and Montgomery, T. 1989. Mice: A flexible testbed for intelligent coordination experiments. In *Proceedings of the 1989 Distributed AI Workshop*, 25–40.

Fudenberg, D., and Levine, D. 1999. Conditional universal consistency. *Games and Economic Behavior* 29(1-2):104–130.

Gerkey, B. P.; Thrun, S.; and Gordon, G. 2006. Visibility-based pursuit-evasion with limited field of view. *International Journal of Robotics Research* 25(4):299–315.

Goldman, C., and Zilberstein, S. 2004. Decentralized control of cooperative systems: Categorization and complexity analysis. *Journal of Artificial Intelligence Research* 22:143–174.

Hespanha, J.; Kim, H. J.; and Sastry, S. 1999. Multiple-agent probabilistic pursuit-evasion games. In *Decision and Control, 1999. Proceedings of the 38th IEEE Conference on*, volume 3, 2432–2437.

Jafari, A.; Greenwald, A.; Gondek, D.; and Ercal, G. 2001. On no-regret learning, fictitious play, and nash equilibrium. In *Proceedings of the Eighteenth International Conference on Machine Learning*, ICML '01, 226–233.

Jennings, N. 1993. Commitments and conventions: The foundation of coordination in multi-agent systems. *Knowledge Engineering Review* 8:223–277.

Keshmiri, S., and Payandeh, S. 2013. On confinement of the initial location of an intruder in a multi-robot pursuit game. *Journal of Intelligent and Robotic Systems* 71(3-4):361–389.

Kitano, H.; Asada, M.; Kuniyoshi, Y.; Noda, I.; and Osawa, E. 1997. Robocup: The robot world cup initiative. In *Proceedings of the First International Conference on Autonomous Agents*, Agents '97, 340–347. New York, NY, USA: ACM.

Nair, R., and Tambe, M. 2005. Hybrid bdi-pomdp framework for multiagent teaming. *Journal of Artificial Intelligence Research* 23:367–420.

Panait, L., and Luke, S. 2005. Cooperative multi-agent learning: The state of the art. *Autonomous Agents and Multi-Agent Systems* 11:387–434.

Parker, L., and Emmons, B. 1997. Cooperative multi-robot observation of multiple moving targets. In *In Proceedings of the IEEE International Conference on Robotics and Automation*, 2082–2089.

Parker, L. 1998. Alliance: an architecture for fault tolerant multirobot cooperation. *IEEE Transactions on Robotics and Automation* 14(2):220–240.

Robinson, J. 1951. An iterative method of solving a game. *The Annals of Mathematics* 54(2):296–301.

Shoham, Y., and Leyton-Brown, K. 2009. *Multiagent Systems: Algorithmic, Game-Theoretic, and Logical Foundations*. Cambridge, U.K.: Cambridge University Press.

Smith, R. 1980. The contract net protocol: High-level communication and control in a distributed problem solver. *IEEE Transactions on Computers* C-29(12):1104–1113.

Stone, P.; Kaminka, G.; Kraus, S.; and Rosenschein, J. 2010. Ad hoc autonomous agent teams: Collaboration without pre-coordination. In *Proceedings of the Twenty-Fourth Conference on Artificial Intelligence*.

Stone, P. 2000. *Layered Learning in Multiagent Systems: A Winning Approach to Robotic Soccer*. MIT Press.

Tomlin, C.; Lygeros, J.; and Sastry, S. 2000. A game theoretic approach to controller design for hybrid systems. *Proceedings of the IEEE* 88(7):949–970.

Undeger, C., and Polat, F. 2010. Multi-agent real-time pursuit. *Autonomous Agents and Multi-Agent Systems* 21:69–107.

Vidal, R.; Shakernia, O.; Kim, H.; Shim, D.; and Sastry, S. 2002. Probabilistic pursuit-evasion games: theory, implementation, and experimental evaluation. *IEEE Transactions on Robotics and Automation* 18(5):662–669.

Welch, G., and Bishop, G. 2006. An introduction to the kalman filter.