Resolution-based policy search for imperfect information differential games

Minh Nguyen-Duc, Brahim Chaib-draa
DAMAS Laboratory
Laval University
Sainte-Foy (Quebec), Canada
{mnguyen, chaib}@damas.ift.ulaval.ca

Abstract

Differential games (DGs), considered as a typical model of game with continuous states and non-linear dynamics, play an important role in control and optimization. Finding optimal/approximate solutions for these game in the imperfect information setting is currently a challenge for mathematicians and computer scientists. This article presents a multi-agent learning approach to this problem. We hence propose a method called resolution-based policy search*, which uses a limited non-uniform discretization of a perfect information game version to parameterize policies to learn. We then study the application of this method to an imperfect information zero-sum pursuit-evasion game (PEG). Experimental results demonstrate strong performance of our method and show that it gives better solutions than those given by traditional analytical methods.*

1 Introduction

In this paper, we will consider the application of multi-agent learning techniques to differential games, which aims to study problems related to conflicts. In a basic DG, there are two agents: a *pursuer*, that tries to catch the other agent, and an *evader*, the mission of which is to prevent this capture. DGs are mathematically modeled by defining state variables, such as the agents' position, velocity and acceleration, by means of differential equations. The study of these games particularly has implications for real-life air combats, or more generally for applications in control and optimization.

Researchers have been recently interested in DGs with imperfect information, *i.e.* in the situation where some agents cannot obtain complete knowledge of the state variables. Since there is currently no general optimal solution method for such games, approximate methods have been considered. However, in most of the cases, the complexity of approximate calculations is still prohibitive.

Therefore, we introduce an approach to find approximate solutions by using multi-agent learning techniques. As discussed in [11], the state/action discretization resolution strongly affects numerical solutions of DGs. This is not only the question how high the resolution is, but also if the resolution is not uniform among different parts of the state/action space. Sheppard [10] proved that an uniform discretization does not guarantee sufficiently good solutions for a DG. We thus propose a method of *policy search* [1, 3, 8] based on *variable resolution* [9, 12]. Such a method also allows us to use of computer simulations with Monte Carlo experiments.

2 Differential games

2.1 Definitions

The theory of DGs is an application of the general game theory, whose subject is the control in conflict situations. It is usually assumed that the motion of the overall system is defined by differential equations called kinematic equations:

$$\frac{\partial x}{\partial t} = f(x(t), u_1(t), ..., u_N(t)) \tag{1}$$

where x is the global kinematic state including the states of all the agents, t is the time, and u_i is the policy of the agent i.

Each agent tries to drive state variables of the game, *i.e.* the agents' position, velocity and acceleration, into a particular target set in the space, by controlling its key variables.

An important DG family is that of *minmax* games [5], each of which has a payoff that some agents minimize and the others maximize:

$$J = \int G(x(t), u_1(t), ..., u_N(t))dt + H \tag{2}$$

$$J^* = \max_{u_1} ... \max_{u_j} \min_{u_{j+1}} ... \min_{u_N} J \tag{3}$$

where J is the overall payoff, J^* is its optimal value, $\int G(.)dt$ is the accumulated payoff, and H is the terminal payoff. Particularly, a two-player *minmax* game is equivalent to a zero-sum game.

The solution of a DG is normally formulated as the agents' optimal policies $u_i^*, i = 1, ..., N$.

2.2 A pursuit-evasion game

The pursuit-evasion game (PEG), a specific zero-sum game, that we present here has played an important role in the research on anti-missile problems [6]. The considered situation is when a defensive missile launched from a ship has to intercept an incoming missile.

The velocity elements subjected to the straight line along which the threaten missile comes to the ship are assumed to be constant. All the policies that the two "agents" can play are thus concerned with the velocity elements perpendicular to this line. In addition, the game is constrained by time, trajectory and control limits. Therefore, it represents a kind of one-dimensional bounded control game. A particular state variable called *zero-effort miss distance* Z is calculated at each moment. It is in fact the minimal relative separation that two agents can reach if they do not change their acceleration. The payoff (see Equation 2) is determined as the final value of Z at the end of the game (a pure terminal payoff).

Gutman [4] introduced the following optimal solution of the game:

$$\frac{a_p^*(\theta)}{(a_p)_{max}} = \frac{a_e^*(\theta)}{(a_e)_{max}} = \operatorname{sgn}(Z(\theta)), \forall \theta < \theta_{endgame} \quad (4)$$

where θ is the normalized remaining time, $a_p(\theta)$ and $a_e(\theta)$ are respectively the accelerations of the pursuer (defensive missile) and the evader (incoming one), $a_p^*(\theta)$ and $a_e^*(\theta)$ are their optimal values.

This result suggests that before the interception moment (if it happens) the evader performs its maximal manoeuvre in only one direction during at least $\theta_{endgame}$, and the pursuer follows it by performing its maximal manoeuvre.

3 Imperfect information setting

A DG is said to be in the imperfect information setting if some agents, called "blind" agents, cannot obtain complete knowledge about the positions and actions of the other agents. Notice that in the perfect information setting, the agents always deterministically play games.

3.1 Stochastic playing

Because of the lack of information about the other agents, a "blind" agent has to play a stochastic policy while hoping some probability of success. Consequently, a "no blind" agent, in its turn, also plays a stochastic policy, still takes advantage of full information. The agents cannot directly minimize/maximize the targeted payoff J of the original perfect information game (see Equation 2), and thus seeks to minimize/maximize the probability J_{pr} that this initial payoff is superior (or inferior) to some threshold ζ.

$$J_{pr} = Pr\{J > \zeta\} = 1 - Pr\{J \leq \zeta\} \quad (5)$$

Such stochastic DGs still cannot be totally resolved. Researchers mainly use optimal solutions of perfect information games as guides for stochastic analyzes of imperfect information games [6].

In the PEG presented in 2.2, the evader can be "blind", *i.e.* it cannot obtain any information of the positions of the pursuer. It does not know the "launching" moment of the pursuer, so it does not know the interception moment either. By following the suggestion from the solution of the perfect information game, the evader can perform its maximal manoeuvre, and switch its manoeuvre direction at random moments. It is also suggested that it maintains a manoeuvre direction during at least $\theta_{endgame}$ before each switching. The random duration between any two consecutive direction switches can follow a distribution $F(\theta)$, *e.g.* the following random telegraph type:

$$F(\theta) = 1 - \exp(-\lambda\theta) \quad (6)$$

This is an example of stochastic policy of the evader. To ensure some interception probability, the pursuer follows the evader by performing randomly biased manoeuvres (see [6]). We recognize that the two assumptions that the evader always performs its maximal manoeuvre and that its direction switching follows the random telegraph rule are not general, and of course, do not guarantee an optimal solution. Actually, the complexity of the mathematical calculation of optimal stochastic policies is so prohibitive that researchers have no choice but to impose such specific assumptions.

3.2 Approximate solutions

When a problem cannot be optimally resolved yet, one tries to find approximate solutions. A natural way to do that is to rely on the discretization, which leads to numerical algorithms.

We now return to the imperfect information PEG described in 3.1. Shinar and Silberman [11] proposed a discrete model of the game. The one-dimensional kinematic space is uniformly discretized into $2m + 1$ positions from $-m$ to m. The velocities of the pursuer and the evader respectively vary from $-d$ to d and from $-kd$ to kd (the evader is k times quicker than the pursuer). The duration of the game is fixed to T time steps. The two agents do not use their accelerations but, more simply, use their velocities to play the game. So this game is much less general than the one described in 3.1. The pursuer taking advantage of full observation can moreover play with its "launching" moment.

Shinar and Silberman [11] also tried to solve this discrete PEG in very simple cases, *e.g.* $m = 4$, $T = 4$, $d = 1$, $k = 2$, by establishing a global matrix game from all the non-dominated deterministic policies (pure policies), calculated for both agents. Formally, any pair of pure policies, $U_{e,i}$ and $U_{p,j}$, can be presented as follows:

$$U_{e,i} = \{x_{e,0}, v_{e,1}, ..., v_{e,T}\} \in \mathbf{U}_e \quad (7)$$

$$U_{p,j} = \{t_{p,0}, x_{p,0}, v_{p,t_{p,0}+1}, ..., v_{p,T}\} \in \mathbf{U}_p, \quad (8)$$

$$\text{with } v_{p,t} = d \, \text{sgn}(x_{e,t} - x_{p,t}), \forall t \in \{1, ..., T\}$$

where $\mathbf{U}_e$ and $\mathbf{U}_p$ are respectively the sets of all the possible policies for the two agents, $x_{e,t}, x_{p,t}$ and $v_{e,t}, v_{p,t}$ are their positions and velocities at time t, and $t_{p,0}$ is the "launching" moment of the pursuer.

The two stochastic policies are in fact the Nash equilibrium of the matrix game. Such policies are not instantaneous or, in other words, they must be totally pre-calculated before one makes the agents play the game. The size of the matrix as well as the policy calculation complexity are polynomial in m and exponential in T. One should pay attention that for the DGs the time is usually considered as a state variable.

4 Learning approach

4.1 How to learn stochastic policies?

With the aim to reduce the complexity of the stochastic policies' calculation while ensuring their high quality, we rely on the application of multi-agent reinforcement learning techniques. However, learned policies will only bring some significance if we reach some degree of control over the learning process. For example, we are able to increase/reduce the learning time while ensuring the convergence. Consequently, we try to control all the parameters concerned with the learning process, *e.g.* the number of "indirect" discretization samples presented below (see 5.1).

We are trying to learn stochastic policies for DGs in the imperfect information setting. If we use a computer simulation as learning environment, we can provide perfect information to all the agents. This allows, firstly, to learn policies for an *intermediate DG version* (modeled in 4.2), that can be defined as a DG in which the agents are provided perfect information, but still play stochastic policies and seek to minimize/maximize the same value as in the imperfect information game (see Equation 5). The overall learning process can thus be decomposed into two steps: (1) agents'policies for an intermediate DG version are learned by using the *resolution-based policy search*; (2) starting from these policies, those for the associated imperfect information DG are estimated by using Monte Carlo experiments (illustrated in 6.1).

4.2 Game modeling

To make agents learn their own stochastic policies, we model an imperfect information DG as a discrete-time decision-making process. Consequently, the time to make decisions is uniformly discretized into small time steps $\Delta\tau$. After each $\Delta\tau$, each agent falls into a state $s_{ag_i} \in \mathbf{S}_{ag_i}$ and chooses an action $a_{ag_i} \in \mathbf{A}_{ag_i}$ according to a stochastic policy, $\pi_{ag_i} : \mathbf{S}_{ag_i} \times \mathbf{A}_{ag_i} \rightarrow [0,1]$. The objective of the agent's playing is to maximize the discounted sum of rewards, $R_{ag_i} : \mathbf{S}_{ag_i} \times \mathbf{A}_{joint} \rightarrow \Re$, over a finite horizon of T time steps. This game model is not Markovian yet [7] because the reward on the joint state, $\widetilde{R}_{ag_i} : \mathbf{S}_{joint} \times \mathbf{A}_{joint} \rightarrow \Re$, and the joint state transition, $Tr : \mathbf{S}_{joint} \times \mathbf{A}_{joint} \times \mathbf{S}_{joint} \rightarrow [0,1]$, cannot be mathematically represented.

Notice that the time, as a state variable, is always continuous. That means that if we change $\Delta\tau$, the time elements of any state remain the same, and the agents can always use the same policies $\pi_{ag_i}, i = 1, ..., N_{agent}$, for both playing and learning.

In a simulation environment, we can provide all the information to all the agents in order, firstly, to learn policies for an intermediate DG version (see 4.1). Based on the imperfect information modeling previously presented, such a game version can be modeled in the form of a Markov game [7]. Indeed, because of full information provided, at any decision-making moment, the state of each agent $s_{ag_i} \in \mathbf{S}_{ag_i}$ is equivalent to the joint state $s_{joint} \in \mathbf{S}_{joint}$. Hence, there exists, for each agent, only one reward function, $R_{ag_i} \equiv \widetilde{R}_{ag_i}$. If we do not take into account erroneous actions and noisy information, this kind of Markov game is quite deterministic, *i.e.* it has a deterministic transition function, $Tr : \mathbf{S}_{joint} \times \mathbf{A}_{joint} \times \mathbf{S}_{joint} \rightarrow \{0,1\}$.

5 Resolution-based policy search

5.1 Variable resolution

As discussed in 4.1, the discretization resolution strongly affects numerical solutions of DGs. We thus intend to apply existing variable resolution methods [9, 12] to the discretization of perfect information game versions, then use this discretization to build parameterized stochastic policies in continuous state-action spaces (see 5.2). Additionally, we seek to control the number of discretization samples.

To be discretized by using variable resolution methods, a perfect information DG must be deterministic or its discretization must be always a known Markov decision process (MDP). This condition will be met if any agent deterministically chooses actions, if any joint action deterministically transforms a joint state into another one and if any joint state determines an immediate reward (*e.g.* the change of *zero-effort miss distance* presented in 2.2). And this is the case for most of the DGs we have studied.

The existing discretization algorithms are mainly based on the definition of state splitting criteria. We are interested in the use of valued criteria (not yes/no criteria) [9, 12]. We begin with a uniform discretization and then improve it. At each "splitting iteration", we select N_{sa} state-action areas with the highest criterion values and split them. At each "merging iteration", we select N_{sa} adjacent area pairs with the lowest criterion values and merge each of them. This process stops after N_{it} iterations or when the criterion values of all the state-action areas are inferior to some threshold.

In this way, we can obtain m discretization samples $s_{joint,i} \in \mathbf{S}_{joint} \times \mathbf{A}_{joint}, i = 1, ..., m$, from which we can derive for each agent m discretization samples $s_i \in \mathbf{S} \times \mathbf{A}, i = 1, ..., m$, and the corresponding set of n state-action areas $\{(\Delta s)_j | j = 1, ..., n\}$.

5.2 Parameterized policy

Our objective is to build for each agent a policy function by focusing it on the most significant state-action areas. The policy parameterization algorithm can be represented as follows:

Inputs A sampling $s_i \in \mathbf{S} \times \mathbf{A}, i = 1, ..., m$ (or n state-action areas); the required number of parameters q.

Output A policy function $\pi(s)$ having q parameters w_j, with $j = 1, ..., q$.

1. Assuming that the initial probability to be in any state-action area $(\Delta s)_i, i = 1, ..., n$, is the same and equal to $1/n$, calculate for each area
$$\widehat{\pi}((\Delta s)_i) = \frac{1}{n(\Delta s)_i} \tag{9}$$
2. Determine the q most significant state-action areas $(\Delta s)_j, j = 1, ..., q$, whose $\widehat{\pi}((\Delta s)_j)$ is to be modified (see Appendix A).
3. Define q parameters $w_j, j = 1, ..., q$, satisfying the following q linear equations:
$$(\Delta s)_j \widehat{\pi}((\Delta s)_j) = \tag{10}$$
$$(\Delta s)_j w_j - \frac{1}{q-1}\Big(\sum_{k=1}^{j-1}(\Delta s)_k w_k + \sum_{k=j+1}^{q}(\Delta s)_k w_k\Big)$$
We can represent $\widehat{\pi}((\Delta s)_j) = f_{(\Delta s)_j}(w_1, ..., w_q)$, with $j = 1, ..., q$.
4. Build the policy function $\pi(s)$ by smoothly interpolating all the $\widehat{\pi}((\Delta s)_i), i = 1, ..., n$, including the q linear functions of the parameters $f_{(\Delta s)_j}(w_1, ..., w_q)$, $j = 1, ..., q$; then normalize $\pi(s)$.

Intuitively, we assume that all the state-action areas initially have the same probability. Afterwards, we define q parameters that modify the probability of the q state-action areas considered as the most significant. Concerning the notation, we use from now $\pi(s, a)$ instead of $\pi(s)$.

5.3 Policy update

The purpose of the learning process for a single agent is to maximize the performance ρ of the policy $\pi(s, a)$ (according to [1, 8]):

$$\rho(\pi) = \int V_\pi(se_T)dse_T, \tag{11}$$
$$\text{with } V_\pi(se_T) = \sum_{t=1}^{T}\Big[\prod_{j=1}^{t}\pi(s_j, a_j)\Big]\gamma^t R(s_t, a_t)$$

where the sequence $se_T = (s_1, a_1, ..., s_T, a_T)$ represents a trial, and $R(s, a)$ is the reward function (to simplify the formalism, we do not take into account the other agents yet).

After performing a trial (or several trials) by following the current greedy policy with some exploration (not detailed here), we update the parameters of the policy function by climbing the gradient of performance:

$$\begin{aligned}\Delta w_i &= \alpha \frac{\partial V_\pi(se_T)}{\partial w_i} \\ &= \alpha \sum_{t=1}^{T}\Big(\sum_{j=1}^{t}\frac{\partial \ln \pi(s_j, a_j)}{\partial w_i}\Big)P_{\pi,t}(s_t)\gamma^t R(s_t, a_t),\end{aligned} \tag{12}$$

$$\text{with } P_{\pi,t}(s_t) = \prod_{j=1}^{t}\pi(s_j, a_j)$$

where α is the learning rate, γ is the discount factor, and $P_{\pi,t}(s_t)$ is the probability to be in s_t after t steps while following π from a single start sate s_1, assumed to be fixed.

5.4 Adaptation to the adversary

The main challenge of the application of a mono-agent learning technique to a multi-agent problem is the adaptation to the other learning agents which make the world not stationary. In this article, we deal with the simplest case of adaptation to an only adversary. Hence, the policy update is reformulated as follows:

$$w_i \leftarrow w_i + \alpha \sum_{t=1}^{T} \Big[\Big(\sum_{j=1}^{t} \frac{\partial \ln \pi_1(s_{1,j}, a_{1,j})}{\partial w_i} \Big) P_{\pi_1,\pi_2,t}(s_{1,t}, s_{2,t}) \gamma^t R(s_{1,t}, s_{2,t}, a_{1,t}, a_{2,t}) \Big] \quad (13)$$

where the subscript 1 indicates the agent which makes the update, and the subscript 2 indicates the adversary.

Although one of the agents is "blind", during an off-line learning process using simulations, we can provide perfect information to all the agents (see 4.2). The agents' state is therefore common, and the policy update is still reformulated as follows:

$$w_i \leftarrow w_i + \alpha \sum_{t=1}^{T} \Big[\Big(\sum_{j=1}^{t} \frac{\partial \ln \pi_1(s_j, a_{1,j})}{\partial w_i} \Big) P_{\pi_1,\pi_2,t}(s_t) \gamma^t R(s_t, a_{1,t}, a_{2,t}) \Big] \quad (14)$$

A well-known method to adapt to a learning adversary is "*Win or Learn Fast*" proposed by [2], which suggests that the learning rate should be small when the policy π is better than an average policy $\overline{\pi}$, and should be big otherwise, as follows:

$$\overline{\pi}_1(s,a) \leftarrow \overline{\pi}_1(s,a) + \frac{1}{N_{trial}} (\pi_1(s,a) - \overline{\pi}_1(s,a)) \quad (15)$$

$$\alpha = \begin{cases} \alpha_{min} & \text{if } V_{\pi_1,\pi_2}(se_T) > V_{\overline{\pi}_1,\pi_2}(se_T) \\ \alpha_{max} & \text{otherwise} \end{cases} , \quad (16)$$

with $V_{\pi_1,\pi_2}(se_T) =$

$$\sum_{t=1}^{T} \Big[\prod_{j=1}^{t} \pi_1(s_j, a_{1,j}) \pi_2(s_j, a_{2,j}) \Big] \gamma^t R(s_t, a_{1,t}, a_{2,t})$$

where N_{trial} is the number of performed trials.

6 Experimental evaluations

6.1 Learning

In the perfect information setting, the optimal deterministic playing of the one-dimensional PEG introduced in 2.2 can be informally specified as follows:

- At first, the evader begins to play and does not change its position;
- The pursuer chooses its "launching" moment, then follows the evader by maximally accelerating;
- Based on the "launching" moment of the pursuer, the evader calculates the interception moment t_f and the end game's starting moment $t_f - \theta_{endgame}$ (see 2.2);
- At the right time, the pursuer starts the end game by maximally accelerating in an unique direction.

This solution can be presented in the agent's reduced state space (θ, Z) (see 2.2), as illustrated in Figure 1, which moreover shows the discretization result of this game in the case where $\langle n, q \rangle = \langle 32 \times 4, 12 + 4 \rangle$ (see 5.2). The evader's action a_e (lateral acceleration) is also discretized into three regions $[-(a_e)_{max}, -0.87(a_e)_{max})$, $[-0.87(a_e)_{max}, 0.81(a_e)_{max})$, $[0.81(a_e)_{max}, (a_e)_{max}]$. And a similar discretization is obtained for the pursuer: $[-(a_p)_{max}, -0.89(a_p)_{max})$, $[-0.89(a_p)_{max}, 0.93(a_p)_{max})$, $[0.93(a_p)_{max}, (a_p)_{max}]$. We obtain similar results for $n = 32 \times 4, 50 \times 4, 72 \times 4$ and $q = 12 + 4, 16 + 4, 24 + 4$.

Starting from these results, we build 9 pairs of q-parameter policy functions $\langle \pi_e^{'}(\theta, Z, a_e), \pi_p^{'}(\theta, Z, a_p) \rangle$ for the associated intermediate game (defined in 4.1).

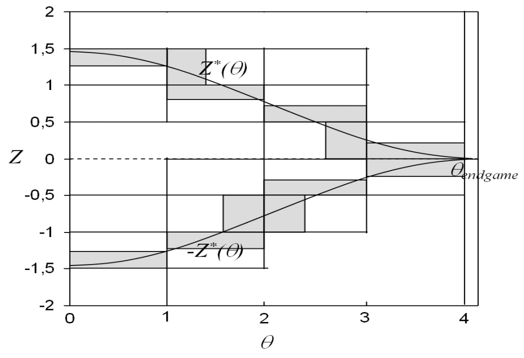

Figure 1. Discretization result of the (θ, Z) space ($n = 32 \times 4$, $q = 12 + 4$).

To learn $\pi_e^{'}$ and $\pi_p^{'}$, we use a specialized defense simulation environment. We choose $\Delta\tau = 1s$, $\alpha_{max} = 0.25$ and $\alpha_{min} = 0.015$. $R_e(\theta, Z, a_e, a_p)$ and $R_p(\theta, Z, a_e, a_p)$ are respectively estimated by using the statistical avoidance probability of the evader and the statistical interception probability of the pursuer. These probabilities are initially set to 0.5, then accumulated and linearly interpolated after each trial. After performing 1000 trials for each pair $\langle n, q \rangle$, $n = 32 \times 4, 50 \times 4, 72 \times 4$, $q = 12 + 4, 16 + 4, 24 + 4$, we obtain 9 pairs of policies $\langle \pi_e^{'}, \pi_p^{'} \rangle$.

We now illustrate the Monte Carlo estimation of policies for the associated imperfect information game. Because a "blind" evader does not know the positions and the "launching" moment of the pursuer, its policy can be formulated as

$$\pi_e(t, a_e) = \int Pr(\theta, Z \mid t) \pi_e^{'}(\theta, Z, a_e) dt$$

where t is the continuous playing time calculated from the game's beginning. The pursuer does not choose a fixed "launching" moment, so its policy can be reformulated as

$$\pi_p(t, Z, a_p) = \int Pr(\theta \mid t)\pi_p^{'}(\theta, Z, a_p)dt$$

If we rewrite $Pr(\theta, Z \mid t) = Pr(\theta \mid t)Pr(Z \mid t)$, we will directly estimate only two probabilities $Pr(\theta \mid t)$ and $Pr(Z \mid t)$ by using Monte Carlo experiments and the linear interpolation. Indeed, for each pair of policies $\langle\pi_e^{'}, \pi_p^{'}\rangle$, we perform 1000 experiments, with the pursuer's choice of "launching" moment following a uniform probability distribution. In this way, we obtain 9 pairs of policies $\langle\pi_e, \pi_p\rangle$ for the imperfect information game.

6.2 Results

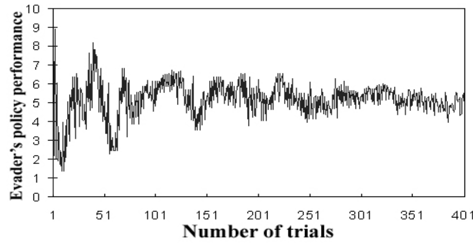

Figure 2. Evader's current policy performance while learning ($n = 72 \times 4$, $q = 24 + 4$, $\Delta\tau = 1s$).

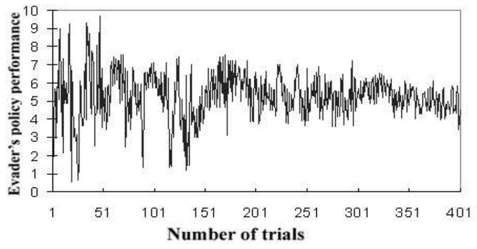

Figure 3. Evader's current policy performance while learning ($n = 72 \times 4$, $q = 24 + 4$, $\Delta\tau = 0.5s$).

Figure 2 and Figure 3 present the variation of the evader's current policy performance *while learning* (see 5.3) where $n = 72 \times 4, q = 24+4, \Delta\tau = 1s, 0.5s$. Both the graphs show clear convergence tendencies. By comparing these two, we recognize that the smaller decision-making time step $\Delta\tau$ is, the more the current policy performance varies, and as a result, the longer the learning time required is. We use from now $\Delta\tau = 1s$.

We evaluate any pair of policies $\langle\pi_e, \pi_p\rangle$, by performing 1000 Monte Carlo experiments. The avoidance probability of the evader as well as the interception probability of the pursuer are all estimated.

n	$q = 12 + 4$	$q = 16 + 4$	$q = 24 + 4$
32×4	0.412	0.432	0.459
50×4	0.477	0.508	0.514
72×4	0.525	0.574	0.632

Table 1. Avoidance probability obtained after learning.

Table 1 shows the avoidance probability of the evader for 9 pairs of learned policies. We can see that n still significantly influences the obtained solution even if the "indirect" discretization is already optimized (see 5.1). On the other hand, because of the evader's stronger lateral acceleration capacity, the bigger n and q are, the freer the evader is, and therefore, the more it can avoid the pursuer's interception.

To compare solution methods for the PEG, we implement two kinds of player, based on traditional analytical methods: (1) a deterministic pursuer that always follows the evader by maximally accelerating; (2) a deterministic end-game evader that estimates the end game's starting moment according to a uniform probability distribution, then plays the end game by maximally accelerating in an unique direction.

$\langle n, q\rangle$	LE *vs.* LP	LE *vs.* DP	EE *vs.* LP
$\langle 50 \times 4, 16 + 4\rangle$	0.492	0.082	0.723
$\langle 72 \times 4, 24 + 4\rangle$	0.368	0.056	0.643

Table 2. Interception probability according to the kind of player.

Table 2 shows the probability of interception of the pursuer for the three cases: a learned evader (LE) *vs.* a learned pursuer (LP), a LE *vs.* a deterministic pursuer (DP), a deterministic end-game evader (EE) *vs.* a LP. The LE and LP's policies are learned with $\langle n, q\rangle = \langle 50 \times 4, 16 + 4\rangle, \langle 72 \times 4, 24 + 4\rangle$. These results show that, while playing against a LP, a LE always gets a lower interception probability than a EE. Similarly, while playing against a LE, a LP always gets a higher interception probability than a DP. Moreover, in comparison with the analytical results given by Shinar and Silberman [11] and Lipman et *al.* [6], our learning-based method gives better solutions in the sense that when a LE plays against a LP it always gets a better avoidance probability.

7 Conclusion

Finding stochastic policies for imperfect information DGs has faced a big problem of computational complexity. In this paper, we have proposed the *resolution-based policy search*, which approximates such stochastic policies. The learning performance depends on several parameters, *e.g.* the number of "indirect" discretization state-action areas n, the number of parameters of a policy function q, and the decision-making time step $\Delta\tau$. The bigger n, p and $1/\Delta\tau$ are, the closer to optimality the solution obtained gets, but the more slowly the learning process converges. However, experimental results have shown that we can choose these parameters' values so that we obtain good solutions after a reasonable number of trials (without loosing the convergence property). Concerning the PEG, with some appropriate learning parameter values, the learned pursuer and evader have proven their greater power, in comparison to two typical kinds of non-learned agent.

Future work will provide more investigation into the experimental behavior of the learning method presented, and into the non-uniform discretization of the decision-making time, as suggested by Munos [8].

A *q* state-action areas

Input A set of state-action areas $(\Delta s)_i, i = 1, ..., n$ generated by a discretization sampling.

Output The q most significant state-action areas $(\Delta s)_j, j = 1, ..., q$.

1. For each $(\Delta s)_i, i = 2, ..., n$, define $rep_{1,i} \triangleq (\Delta s)_i$ and calculate

$$avrg_{1,i} = \frac{1}{3}\Big[(\Delta s)_i + \frac{1}{2}\Big(\frac{(\Delta s)_{i-1} + (\Delta s)_i}{2} + \frac{(\Delta s)_i + (\Delta s)_{i+1}}{2}\Big) + \frac{(\Delta s)_{i-1} + (\Delta s)_i + (\Delta s)_{i+1}}{3}\Big]$$

2. For each pair $\langle(\Delta s)_i, (\Delta s)_{i+1}\rangle, i = 2, ..., n-1$, define $rep_{2,i} \triangleq min((\Delta s)_i, (\Delta s)_{i+1})$ and calculate

$$avrg_{2,i} = \frac{1}{3}\Big[\frac{1}{2}((\Delta s)_i + (\Delta s)_{i+1}) + \frac{(\Delta s)_i + (\Delta s)_{i+1}}{2} + \frac{1}{2}\Big(\frac{(\Delta s)_{i-1} + (\Delta s)_i + (\Delta s)_{i+1}}{3} + \frac{(\Delta s)_i + (\Delta s)_{i+1} + (\Delta s)_{i+2}}{3}\Big)\Big]$$

3. For each triplet $\langle(\Delta s)_i, (\Delta s)_{i+1}, (\Delta s)_{i+2}\rangle, i = 1, ..., n-2$, define $rep_{3,i} \triangleq min((\Delta s)_i, (\Delta s)_{i+1}, (\Delta s)_{i+2})$ and calculate

$$avrg_{3,i} = \frac{1}{3}\Big[\frac{1}{3}((\Delta s)_i + (\Delta s)_{i+1} + (\Delta s)_{i+2}) + \frac{1}{2}\Big(\frac{(\Delta s)_i + (\Delta s)_{i+1}}{2} + \frac{(\Delta s)_{i+1} + (\Delta s)_{i+2}}{2}\Big) + \frac{(\Delta s)_i + (\Delta s)_{i+1} + (\Delta s)_{i+2}}{3}\Big]$$

4. For each pair $\langle rep_{r,t}, rep_{l,g}\rangle$ satisfying $rep_{r,t} \equiv rep_{l,g}$ and $avrg_{r,t} \leq avrg_{l,g}$, remove $rep_{l,g}$ from $REP \triangleq \{rep_{i,j} | \forall i, j\}$.

5. In the set REP, choose the q elements $rep_{i,k}$ whose $avrg_{i,k}$ are the smallest; these elements represent the q most significant areas $(\Delta s)_j, j = 1, ..., q$.

References

[1] L. Baird and A. W. Moore. Gradient descent for general reinforcement learning. In *NIPS'98*, pages 968–974, Cambridge, MA, USA, 1998.

[2] M. H. Bowling and M. M. Veloso. Multiagent learning using a variable learning rate. *Artificial Intelligence*, 136(2):215–250, 2002.

[3] M. H. Bowling and M. M. Veloso. Scalable learning in stochastic games. In *AAAI Workshop Proceedings on Game Theoretic and Decision Theoretic Agents*, Edmonton, Canada, 2002.

[4] S. Gutman. On optimal guidance for homing missiles. *Guidance and Control*, 2:296–300, August 1979.

[5] R. Isaacs. *Differential Games: A Mathematical Theory with Applications to Warfare and Pursuit, Control and Optimization*. J.Wiley and Sons, Toronto, 1965.

[6] Y. Lipman, J. Shinar, and Y. Oshman. Stochastic analysis of the interception of maneuvering antisurface missiles. *Guidance, Control and Dynamics*, 20(4):707–714, July–August 1997.

[7] M. L. Littman. Markov games as a framework for multi-agent reinforcement learning. In *ICML'94*, pages 157–163, New Brunswick, NJ, 1994.

[8] R. Munos. Policy gradient in continuous time. *Machine Learning*, 7:771–791, 2006.

[9] R. Munos and A. W. Moore. Variable resolution discretization for high-accuracy solutions of optimal control problems. In *IJCAI'99*, pages 1348–1355, 1999.

[10] J. W. Sheppard. Colearning in differential games. *Machine Learning*, 33(2-3):201–233, 1998.

[11] J. Shinar and G. Silberman. A discrete dynamic game modelling anti-missile defense scenarios. *Dynamics and Control*, 5(1):55–67, 1995.

[12] W. T. B. Uther and M. M. Veloso. Tree based discretization for continuous state space reinforcement learning. In *AAAI/IAAI'98*, pages 769–774, 1998.