

# Stochastic Fictitious Play using Particle Filters to update the beliefs of opponents strategies

Michail Smyrnakis  
Department of Mathematics  
University of Bristol  
United Kingdom  
m.smyrnakis@bris.ac.uk

David Leslie  
Department of Mathematics  
University of Bristol  
United Kingdom  
david.leslie@bris.ac.uk

## ABSTRACT

Distributed optimization can be formulated as an  $n$  player coordination game. One of the most common learning techniques in game theory is fictitious play and its variations. However fictitious play is founded on an implicit assumption that opponents' strategies are stationary. In this paper we present a new variation of fictitious play in which players predict opponents' strategy using a particle filter algorithm. This allows us to use a more realistic model of opponent strategy.

We used pre-specified opponents' strategies to examine if our algorithm can efficiently track the strategies. Furthermore we have used these experiments to examine the impact of different values of our algorithm parameters on the results of strategy tracking. We then compared the results of the proposed algorithm with those of stochastic and dynamic fictitious play in a potential game and two climbing hill games, one with two players and the other with three players.

Our algorithm converges more quickly to the optimum than both the competitor algorithms. Hence by placing a greater computational demand on the individual agents, less communication is required between the agents.

## Categories and Subject Descriptors

I.2.6 [Artificial Intelligence]: Learning

; I.2.11 [Artificial Intelligence]: Distributed artificial intelligence—*multiagent systems*

## General Terms

Game theory, Distributed optimization

## 1. INTRODUCTION

A multiagent system consists of  $n$  agents that have to interact with each other and coordinate to accomplish a specific task. When the objective is optimization this is naturally

formulated as an  $n$ -player game. Many different learning techniques have been used to solve multiagent optimization problems such as Q-learning [5], minimax-Q learning [11], opponent modeling [17], WOLF [3] and others. However very few of these have theoretical convergence results. On the other hand game-theoretic algorithms such as adaptive play and fictitious play have been proved to converge although in practice this convergence can be very slow [7, 13]. In this paper we introduce a variation of fictitious play which attempts to predict the other players strategies using particle filters. This reduces the number of the steps that fictitious play needs to converge, and hence the communications overhead between the distributed optimizers.

The remainder of this paper is organized as follows. We start with a brief description of game theory, different categories of games, fictitious play and stochastic fictitious play. Section 3 introduces a particle filter based algorithm to update the beliefs about opponents strategy and the results of a tracking scenario. Section 4 presents the results of the method described in Section 3 when incorporated in stochastic fictitious play, comparing with the results of classic stochastic fictitious play and dynamic stochastic fictitious play. We finish with a conclusion.

## 2. BACKGROUND

### 2.1 Games

Game theory is the field of science that deals with the way someone should take a decision when there is interaction between the decision makers. Games can be divided into two general categories: strategic form games and extensive form games. The main difference between strategic and extensive form games is that in strategic form games the actions are made simultaneously but in extensive form games the decision makers might act with knowledge of the action of their opponents. For that reason a basic element of extensive form games is the order of the moves. Generally we can say that extensive form games are a form of multi-player decision trees. The rest of this paper will focus on strategic form games.

The elements of a strategic form game are [8]

- a set of players  $i \in 1, 2, \dots, I$ ,
- a set of actions  $s^i \in S^i$  for each player  $i$ ,
- a set of joint actions,  $s = (s^1, s^2, \dots, s^I) \in S^1 \times S^2 \times \dots \times S^I = S$ ,

- the payoff function  $u^i : S \rightarrow \mathbb{R}$  for each player  $i$ ,

where  $u^i(s)$  is the utility that player  $i$  will gain after a specific joint action  $s$  has been played. The rules that the players are using to select the action that is going to be played in the game are called strategies. When an action of player  $i$  is selected from his set of available actions with probability equal to one then the player acts according to a pure strategy. In the cases that he chooses an action based on a probability distribution  $\sigma^i$  over the available actions he acts according to a mixed strategy. We will use  $\sigma = (\sigma^1, \dots, \sigma^I)$  to denote a joint mixed strategy.

Different kinds of strategic form games can be classified according to their payoff functions. In particular we separate games into coordination and non-coordination games. Zero-sum games are a class of non-coordination games where  $\sum_i u^i(s) = 0$ . In this kind of game when player  $i$  obtains some positive utility  $u^i$  his opponents  $-i$  obtain a negative utility  $-u^i$ . An example of the Zero-sum games is the matching pennies game which has payoffs depicted in Table 1. Another class of non-coordination games are

	H	T
H	1,-1	-1,1
T	-1,1	1,-1

Table 1: matching pennies

the constant sum games where  $\sum_i u^i(s) = c$ , where  $c$  is a constant. In these games there is not the restriction that Player  $i$ 's opponents will lose the same amount of utility units that Player  $i$  will gain. Generally we can characterize non-cooperative games as competitive games since the utilities that a player gains is his opponent's loss. Although non-coordination games are the most-studied games, they are the least applicable to distributed coordination.

On the other hand in coordination games the players' payoffs are simultaneously maximized and agents must collaborate to choose the action that maximizes their payoffs. A sub-category of coordination games are potential games. In a potential game utilities follows the formula:

$$u^i(s^i, s^{-i}) - u^i(\tilde{s}^i, s^{-i}) = \phi(s^i, s^{-i}) - \phi(\tilde{s}^i, s^{-i}) \quad (1)$$

where  $\phi$  is a potential function and the above equality stands for every player  $i$  for every action  $s^{-i} \in S^{-i}$  and for every pair of action  $s^i, \tilde{s}^i \in S^i$ , where  $S^i$  and  $S^{-i}$  represent the set of all available actions for player  $i$  and his opponents respectively.

A distributed optimization problem can be cast as a game in which all players receive the same reward, namely the global utility of the joint action [18]. This global reward then acts as a potential. However, it is often not possible for all players to know this global utility function. An alternative approach that results in local utility functions, and that results in a potential game, uses Wonderful Life Utilities [1]. For the remainder of the paper we concentrate on learning in potential games.

A set of pure strategies satisfying the property that, when

the players choose these actions none of them can do better by unilaterally changing his or her strategy, is called a pure strategy Nash equilibrium. In general we can define the Nash equilibrium as a joint mixed strategy  $\sigma^*$  that for each player  $i$  satisfies

$$u^i(\sigma^{i*}, \sigma^{-i*}) \geq u^i(s^i, \sigma^{-i*}) \quad \text{for all } s^i \in S^i \quad (2)$$

where strategy profile  $\sigma^{i*}$  is the strategy of player  $i$  in the game, and  $\sigma^{-i*}$  is the strategy of player  $i$ 's opponents, and  $u^i(\sigma^{i*}, \sigma^{-i*})$  (resp.  $u^i(s^i, \sigma^{-i*})$ ) is the utility which is gained from Player 1 when he plays the action  $\sigma^{i*}$  (resp.  $s^i$ ) and the opponents' strategy is  $\sigma^{-i*}$ . Every potential game has at least one (there may be more than one) pure strategy Nash equilibrium [13]. A sub case of the pure equilibria are the strict equilibria. A pure equilibrium is strict if each player has a unique best response to his opponents actions.

An important category of equilibria which are of particular interest in distributed optimization are the Risk Dominant Equilibria. This can be seen from the Stag-Hunt game in Table 2, which depicts a potential game where there is a risk dominant strategy. In this game the players will gain

	L	R
U	9,9	0,8
D	8,0	7,7

Table 2: Stag-Hunt Game

greater utilities if they collaborate and play the combination (U,L), yet this is a risky choice since if your "opponent" fails to cooperate you receive nothing (the task cannot be completed by you alone). On the other hand if Player 1 plays D he will always will gain some utility. This payoff will be less than 9 utility units which is the maximum payoff but he will earn at least 7 utility units whatever the action of Player 2 will be. The same also stands for Player 2, if he chooses to play R. For that reasons the two players will be able to coordinate to the (U,L) equilibrium only if Player 1 believes that Player 2 will play L with probability greater than 0.875 and Player 2 believes that Player 1 will play U with probability greater than 0.875. This level of confidence is difficult to obtain in a distributed setting.

## 2.2 Fictitious Play

We have seen that distributed optimization can be framed as the task of finding a Nash equilibrium in a potential game. Game-theoretical learning algorithms solve this problem by evolving each players initial strategy in response to the strategies selected by the opponents. There is then a strong correspondence between learning in games and iterative distributed optimization.

One of the most widely used learning techniques in game theory is fictitious play. According to fictitious play each player chooses his action according to the best response to his beliefs about the strategy of his opponent. Where best response is defined as

$$BR = \underset{s^i \in S^i}{\operatorname{argmax}} u^i(s^i, \sigma^{-i}) \quad (3)$$

Initially each player has some prior beliefs about the probability with which his opponent plays each action. After each turn the players update their beliefs about their opponent strategy and play again the best response according to their beliefs. More formally for a two player strategic form game with finite strategy space  $S^1$  and  $S^2$  with payoff functions  $u^1$  and  $u^2$  we have that an arbitrary non-negative initial weight functions  $\kappa_0^i$  are updated using the formula:

$$\kappa_t^i(s^{-i}) = \kappa_{t-1}^i(s^{-i}) + \begin{cases} 1 & \text{if } s_{t-1}^{-i} = s^{-i} \\ 0 & \text{otherwise} \end{cases} \quad (4)$$

The mixed strategy of the opponent is estimated from the following formula:

$$\sigma_t^{-i}(s^{-i}) = \frac{\kappa_t^i(s^{-i})}{\sum_{s^{-i} \in S^{-i}} \kappa_t^i(s^{-i})} \quad (5)$$

An alternative to that rule which maximizes the likelihood of the observations is the usage of a Bayesian approach. The player starts with a prior belief for the set of the distributions over the actions and updates his beliefs using the Bayes rules. If the prior is chosen to be Dirichlet distributed then the two approaches are equivalent [7].

Player  $i$  then chooses the action which maximizes his payoffs according to his beliefs about his opponents' play as they have been constructed using the formulas (4) and (5). These updating methods treat the environment of the game as stationary and implicitly assume that the actions of the players are sampled from a fixed probability distribution. A variation of fictitious play that treats the opponents' strategies as dynamic and gives bigger weights to the recent observations while we calculate each action's probability is dynamic fictitious play [7]. According to this variation of fictitious play we estimate each opponents probability to play an action  $i$  using the formula:

$$\sigma_t^j(s^j) = (1 - z)\sigma_{t-1}^j(s^j) + zI_{s_t^j=s^j} \quad (6)$$

where  $z$  is a constant and  $I_{s_t^j=s^j} = \begin{cases} 1 & \text{if } s_t^j = s^j \\ 0 & \text{otherwise.} \end{cases}$

The main aim for a learning algorithm like fictitious play is to converge to a set of strategies that are a Nash equilibrium. We define convergence of fictitious play to occur when the marginal empirical distributions have converged. The marginal empirical distribution has the following form:

$$d_t^j(s^j) = \frac{\kappa_t(s^j) - \kappa_0(s^j)}{t} \quad (7)$$

It has been proved [7] that if  $\sigma$  is a strict Nash equilibrium and it is played at time  $t$  then it will be played for all the other iterations of the game. Also that any steady state of fictitious play is a Nash equilibrium. Furthermore it has been proved that fictitious play converges for  $2 \times 2$  games with generic payoffs [12], zero sum games [15], games that can be solved using iterative dominance [14] and potential games [13]. There are also games where the marginal empirical distribution do not converge. An example of such game is Shapley's game [16]

## 2.3 Stochastic Fictitious Play

A player  $i$  of a game who is selecting his actions  $s^i$  using fictitious play is actually selecting his actions from his strategy space  $S^i$  according to his beliefs about his opponent's actions. Randomization is allowed only in the case that the player is indifferent between his available actions, but it is very rare in generic payoff strategy games for a player to be indifferent between the available actions [6]. Stochastic fictitious play is a modification of fictitious play that allows players to play the available actions according to a mixed strategy  $\sigma^i$ . This was originally introduced to allow convergence of strategies to a mixed strategy Nash equilibrium [7] but has the added advantage of introducing exploration into the process. There are two equivalent formulations of stochastic fictitious play. The first one is based on Harsanyi's purification theorem [10], which allows randomization and chooses actions according to mixed strategies using randomly distributed perturbations of the players' estimated payoffs. The latter, and the one which is used in our algorithm, is based on randomization even when the players are not indifferent about the opponents strategy. Each player  $i$  chooses his actions according to a smooth best response to the estimated strategies of the other players. The most common smooth best response to  $\sigma^{-i}$  is the distribution over player  $i$ 's actions which satisfies:

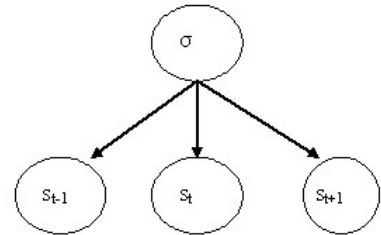
$$\overline{BR}(\sigma^{-i})(s^i) = \frac{\exp(u^i(s^i, \sigma^{-i})/\lambda)}{\sum_{\tilde{s}^i} \exp(u^i(\tilde{s}^i, \sigma^{-i})/\lambda)} \quad (8)$$

where  $\lambda$  is the randomization parameter. Values of  $\lambda$  close to zero allow very little randomization, whereas large values of  $\lambda$  result in complete randomization [7].

## 3. USING PARTICLE FILTERS TO TRACK OPPONENTS' STRATEGY.

### 3.1 Introduction

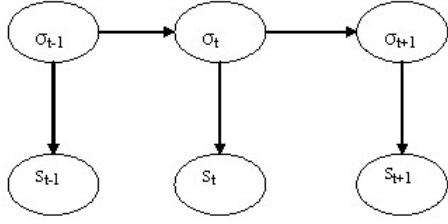
In this section we only consider inference over opponent mixed strategy in fictitious play. Separate estimates will be formed identically and independently for each opponent. We therefore consider only one opponent, drop all dependence on player  $i$ , and write  $s_t$  and  $\sigma_t$  for the opponent's action and strategy respectively. In fictitious play, as depicted in Figure 1, we are assuming that Player  $i$ 's opponent has a fixed mixed strategy  $\sigma$  over time, from which all of their actions arose.



**Figure 1: Graphical model of the standard fictitious play assumption**

A more rational assumption about the strategies of Player  $i$ 's opponents is that they are changing over time, instead of having a fixed value. According to this assumption, at every time  $t$  Player  $i$ 's opponent has a different strategy  $\sigma_t$ .

Figure 2 depicts fictitious play when opponent strategy is changing over time.



**Figure 2: Graphical model for a more realistic fictitious play model**

For the tracking scenario Player  $i$ 's objective is to predict efficiently his opponent's strategy  $\sigma_t$ , using the information he has from the actions  $s_{1:t-1}$  he has observed and his beliefs about his opponents strategies till that time  $\sigma_{1:t-1}$ . This objective can be represented as a Hidden Markov Model (HMM). HMMs try to predict the value of an unobserved variable  $x_t$  using the observations of another variable  $z_t$ . In control theory this kind of problem is referred to as the filtering problem. Particle filters are widely used to solve this sort of problem.

From the family of particle filter algorithms the Sequential Importance Sampling (SIS) filter with resampling was used. According to this method we sample  $N$  particles from a distribution  $q(\cdot|\cdot)$  (importance function) which has similar characteristics to the density function of the data and we associate some weights  $w_t$  to each sample. At the initial step we set equal weights  $w_t = 1/N$  for all the particles because we have no observations to support other values for the weights. We sample from the importance function instead of sampling from the density function of the data  $p(\cdot|\cdot)$  because in most cases it is difficult to sample directly from  $p$ . A general algorithm for SIS filters is depicted in Table 3.

Basic SIS algorithm
<ul style="list-style-type: none"> <li>• Draw <math>N</math> samples from the proposal distribution of <math>x_t</math> where <math>x_t^n \sim q(x_t^n x_{t-1}^n, z_t)</math> and <math>0 &lt; n &lt; N</math></li> <li>• Calculate the weights <math>w_t</math> where <math>w_t^n = w_{t-1}^n \cdot \frac{p(z_t x_t^n)p(x_t^n x_{t-1}^n)}{q(x_t^n x_{t-1}^n)}</math></li> <li>• Normalize the weights <math>w_t^n = \frac{w_t^n}{\sum_{n=1}^N (w_t^n)}</math></li> <li>• Compute the value of <math>N_{eff}</math> <math>N_{eff} = \frac{1}{\sum_{n=1}^N (w_t^n)^2}</math></li> <li>• If <math>N_{eff} &lt; N_{threshold}</math> resample according to the weights using systematic resampling [2].</li> </ul>

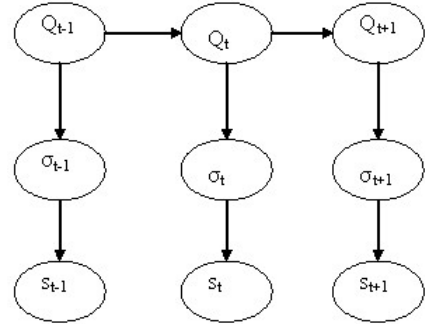
**Table 3: SIS algorithm [9]**

The reason for resampling is that after some iterations most of the weights will be approximately zero with very few of

them large enough to influence the results. Using resampling avoids this phenomenon since it multiplies particles with high weights and discards particles with small weights.

### 3.2 An SIS algorithm to update fictitious play beliefs

In this section we describe how to use the particle filter within fictitious play. The first challenge is that the opponent's strategy is a probability distribution and for that reason there are constraints that make it difficult to sample directly from it. A way to avoid this problem is to introduce a new layer to Figure 2 with unconstrained propensities of our opponent and sample from them, instead of the strategies. Figure 3 depicts the propagation of fictitious play using the propensities, where  $Q_t$ ,  $\sigma_t$  and  $s_t$  are the propensity, the strategy and the action of Player's  $i$  opponents at time  $t$  respectively.



**Figure 3: Players  $i$  opponents actions during the time**

We assume that  $Q(s_0) \sim MVN(0, I)$  and that  $Q(s_t)$  depends only on its previous value  $Q(s_{t-1})$  according to the equation:

$$Q(s_t) = Q(s_{t-1}) + \eta_t \quad (9)$$

where  $\eta_t \sim MVN(0, v^2 I)$ . We can represent the algorithm of SIS with resampling (Table 3) using the fictitious play notation, where  $Q_t$  and  $s_t$  will replace  $x_t$  and  $z_t$  respectively. So in every iteration of the particle filter we can use equation (9) to sample the  $N$  particles from our opponent's propensity  $Q_t(s_t)$  and use a form of Boltzmann distribution to update the beliefs about the opponent's actions given the opponents propensities. For an action  $s_t \in S$ :

$$P(s_t|Q_t) = \frac{e^{Q_t(s_t)/\tau}}{\sum_{\tilde{s} \in S} e^{Q_t(\tilde{s})/\tau}} \quad (10)$$

where  $\tau$  is a temperature constant.

We can observe in Table 3 that the weights are propagated using the following formula:  $w_t = w_{t-1} \cdot \frac{p(z_t|x_t)p(x_t|x_{t-1})}{q(x_t|x_{t-1})}$ . From the assumptions we have made for the distribution of  $Q$ , we know that the distribution of  $Q_t|Q_{t-1}$  is a normal distribution. Since it is easy to sample from a normal distribution we can draw our  $N$  samples from the normal distribution in such way where  $p(x_t|x_{t-1}) = q(x_t|x_{t-1})$  so we will be able to simplify the formula which is used to evaluate the weights. The simplified formula can be written  $w_t = w_{t-1} \cdot p(z_t|x_t)$ . Using fictitious play notation we can

write  $w_t = w_{t-1} \cdot P(s_t|Q_t)$ . Table 4 summarizes the SIS algorithm with resampling using fictitious play notation for tracking.

<ul style="list-style-type: none"> <li>• Propagate N particles from the model of <math>Q_t</math> where <math>Q_t^n = Q_{t-1}^n + \eta_t^n</math></li> <li>• Calculate the weights <math>w_t</math> where <math>w_t^n = w_{t-1}^n \cdot P(s_t Q_t^n) = w_{t-1}^n \cdot \frac{e^{Q_t^n(s_t)/\tau}}{\sum_{\bar{s} \in S} e^{Q_t^n(\bar{s})/\tau}}</math></li> <li>• Normalize the weights <math>w_t^n = \frac{w_t^n}{\sum_{n=1}^N (w_t^n)}</math></li> <li>• Compute the value of <math>N_{eff}</math> <math>N_{eff} = \frac{1}{\sum_{n=1}^N (w_t^n)^2}</math></li> <li>• If <math>N_{eff} &lt; N_{threshold}</math> resample according to the weights using systematic resampling.</li> </ul>
--

Table 4: SIS with resampling for fictitious play algorithm

### 3.3 Experiments to test the impact of algorithm parameters

Initially simulations were employed to examine if the proposed model is able to track sufficiently a pre-specified opponent's strategy over time. Also these experiments were implemented to monitor the impact of the temperature parameter  $\tau$  and innovation variance parameter  $v$  in the results of our algorithm. The experiments can be divided into two categories. The first category contains experiments where our opponent chooses his actions from a pure strategy space. In that category the opponent played action 2 from iteration 251 to 750 and action 1 in all the other iterations of the game. The later category allows randomization and the opponent chooses his actions from a mixed strategy space. The opponent strategy had the following form over the  $t$  iterations of the game:  $\sigma_t(1) = \frac{\cos \frac{2\pi t}{n} + 1}{2} = 1 - \sigma_t(2)$ .

Figures 4 and 5 depict the result of the tracking for different values of the standard deviation  $v$  with  $\tau = 1$ . In the pure strategy space we can observe that large values of  $v$  result in faster tracking as we can see in Figure 4. This happens because if we use large values of  $v$  then our algorithm actually has little memory and just follows the previous action of our opponent. This is an effective tactic in a steady environment like that of the pure strategy space. However when with large  $v$  we allowed mixed strategies then the results were too noisy and the tracking of our opponent policy was poor as we can observe in Figure 5. Note that in Figure 5 all the plots approximately track the shape of the opponents strategy, with more or less noise according to the value of  $v$ .

Figures 6 and 7 depict the results of the game for different values of  $\tau$  and fixed value of  $v = 0.1$  Both in pure and mixed strategy space (Figures 6 and 7 respectively) we can observe that small values of  $\tau$  makes the changes in the shape of the predicted distribution sharper and allow predictions close to 0 and 1. On the other hand very big values of  $\tau$  i.e.

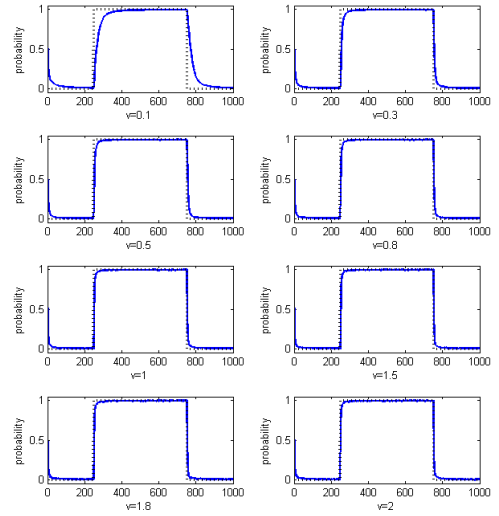


Figure 4: Tracking of opponent strategy at a pure strategy space for different values of  $v$  for action 1. Where the pre-specified strategy of the opponent and its prediction are depicted as the dot and solid line respectively

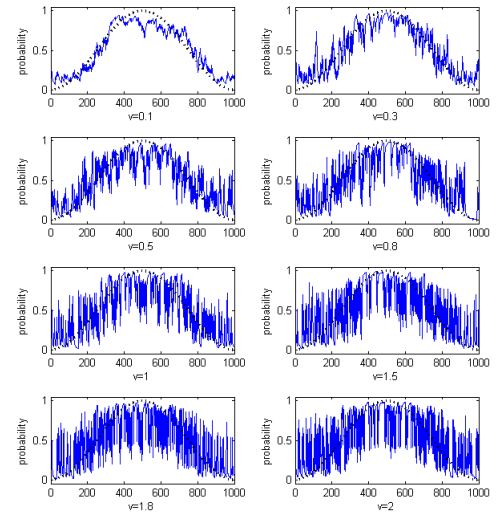
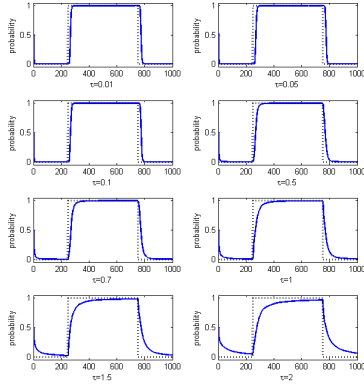
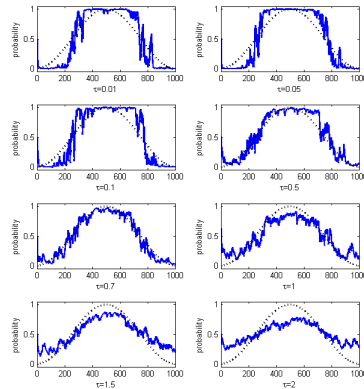


Figure 5: Tracking of opponent strategy at a mixed strategy space for different values of  $v$  for action 1. Where the pre-specified strategy of the opponent and its prediction are depicted as the dot and solid line respectively

$\tau = 2$  make the prediction too smooth and the prediction never reaches 1. The value of the parameter  $\tau$  doesn't affect directly the memory of our system, unlike the parameter  $v$ . This can be seen from Figure 7 where all the plots have almost the same noise level.



**Figure 6: Tracking of opponent strategy at a mixed strategy space for different values of  $\tau$  for action 1. Where the pre-specified strategy of the opponent and its prediction are depicted as the dot and solid line respectively**



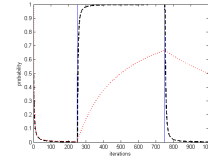
**Figure 7: Tracking of opponent strategy at a mixed strategy space for different values of  $\tau$  for action 1. Where the pre-specified strategy of the opponent and its prediction are depicted as the dot and solid line respectively**

The result of the proposed algorithm were compared with those of fictitious play when we estimate opponents strategy. Figures 8 and 9 depict the fictitious play estimate of the opponent strategy and the opponents real strategy in contrast with the predictions of the proposed algorithm. The parameters which were used for our algorithm were 0.3 and 0.7 for  $v$  and  $\tau$  respectively. In both figures we can observe that the results of the proposed algorithm outperforms the results of the tracking that the fictitious play algorithm achieves.

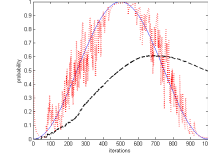
## 4. IMPLEMENTATION IN GAMES

### 4.1 Particle Filters Stochastic Fictitious Play

Until now we have managed to show that the proposed method of updating the beliefs of fictitious play performs better in tracking opponents strategy than classic fictitious



**Figure 8: Comparing the tracking of the particle filter algorithm (dash line) with fictitious play (dot line) at a pure strategy space**



**Figure 9: Comparing the tracking of the particle filter algorithm (dot line) with fictitious play (dash line) at a mixed strategy space**

play. We will combine the algorithm in Table 4 with decision making using smooth best response to introduce a new version of stochastic fictitious play. The change to the algorithm of Table 4 is that after propagating the  $Q$  values, we use these to predict opponent strategy and thus select an action. Specifically, we calculate the estimate

$$\sigma_t(s_t) = \sum_{n=1}^N w_t^n \cdot \frac{e^{Q_t^n(s_t)/\tau}}{\sum_{\bar{s} \in S} e^{Q_t^n(\bar{s})/\tau}} \quad (11)$$

then select an action using  $\overline{BR}(\sigma_t)$  as defined in (8). A compact representation of that algorithm is depicted in Table 5.

- Draw  $N$  particles from the model of  $Q_t$
- Calculate the new  $\sigma_t$
- Compute smooth best response ( $\overline{BR}$ )
- Choose an action according to ( $\overline{BR}$ )
- Observe opponent's action
- Update and normalize the weights
- Resample if necessary

**Table 5: Stochastic fictitious play using particle filters**

### 4.2 Results

In this section the performance of the particle filter stochastic fictitious play algorithm in three coordination games was examined in contrast with the results of classic stochastic fictitious play and dynamic fictitious play. These games are depicted in Table 6, 7 and 8. Table 6 depicts a simple coordination game with asymmetric penalties for off-diagonal play. Table 7 introduces the climbing hill game used by Claus and Boutilier [4] to demonstrate the slow convergence of algorithms that estimate action values based on the entire

	Stag	Hare
Stag	8,8	0,-7
Hare	-7,0	7,7

**Table 6: Stag hunt game**

	U	M	D
U	11,11	-30,-30	0,0
M	-30,-30	7,7	6,6
D	0,0	0,0	5,5

**Table 7: Climbing hill game with two players**  
[4]

history of play, in which we expect dynamic fictitious play and our new algorithm to significantly outperform classical stochastic fictitious play. Table 8 presents a more extreme version of this game in which three players must climb up a utility function; in this scenario each agent estimates the strategy of each opponent independently.

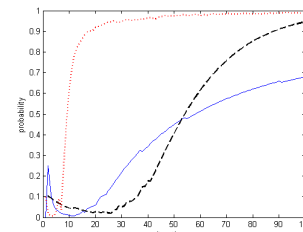
The results which are presented are for 1000 replications of a learning episode of 1000 iterations for each game. We selected the values of the parameters for each algorithm on the premise that the algorithms with these parameters have the best results in the tracking experiment with pre-specified opponents strategy. The values of the parameters for the particle filter tracking algorithm were 0.3 and 0.7 for  $v$  and  $\tau$  respectively. The value of the learning parameter  $z$  of dynamic fictitious play was set 0.05. For the all of the algorithms we set the randomization parameter  $\lambda$  in the smooth best response function equal to 1, allowing the same randomization for all algorithms. For the Stag hunt game the particle filter algorithm converged to the Pareto efficient equilibrium (Stag, Stag) in 92% of the replications. For each replication of 1000 iterations we computed the mean payoff. After the end of the 1000 replications the overall mean of the 1000 payoff means was computed. The overall mean payoffs for player 1 and 2 were 7.78 and 7.74 respectively. The classic stochastic fictitious play converged to the (Stag, Stag) equilibrium in all replications but very slowly with overall mean payoff 7.53 for both players. The results for dynamic fictitious play were better than our algorithm since it converged always to the Pareto efficient equilibrium with mean payoff 7.9 for both players.

In the Climbing hill game with two players our algorithm failed to reach the equilibrium (U,U) where the players gain the highest utility and it converged to the (M,M) equilib-

	U	M	D	U	M	D	U	M	D
U	0	0	0	-300	70	80	100	-300	90
M	0	50	40	-300	60	0	0	0	0
D	0	0	30	0	0	0	0	0	0
	U			M			D		

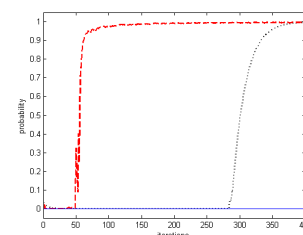
**Table 8: Climbing hill game with three players.** Player 1 selects rows, Player 2 selects columns, and Player 3 selects the matrix.

rium. The overall mean payoffs for player 1 and 2 were 6.98. In this game the stochastic and dynamic fictitious play also converged to the (M,M) equilibrium. The overall mean payoffs for player 1 and 2 were 6.95 and 6.96 respectively. The results are very close because both algorithms converge quickly to the equilibrium point. If we take a game with 100 iterations then the results are not so close. The mean payoff for stochastic fictitious play is 6.46, for dynamic fictitious play 6.53 and for the proposed algorithm is 6.85. The difference we observe is because our algorithm converges faster to the equilibrium than the other two algorithms. We can observe this difference in Figure 10.



**Figure 10: Probability of playing the (M,M) equilibrium for the particle filter algorithm (dot line), stochastic fictitious play (solid line) and dynamic fictitious play (dash line) for the row player climbing hill game**

In the more complicated environment of the Climbing hill game with three players our algorithm also outperformed both dynamic and stochastic fictitious play. The overall mean payoffs for player 1,2 and 3 were 98.6 when payoffs for dynamic and stochastic fictitious play were 91.7 and 70.3 respectively. Stochastic fictitious play didn't converged to the Nash equilibrium after 1000 replications. Also when we are concerned about the speed of convergence of the particle filter algorithm and dynamic fictitious play our algorithm outperforms the dynamic fictitious play. This can be seen if we reduce the iterations of the game to 200. Then the overall mean payoff of our algorithm is 93.2 utility units when for dynamic fictitious play is 63.12. This is because our algorithm needs 50 replications to reach the Nash equilibrium when dynamic fictitious play needs at least 300. This difference is depicted in Figure 11.



**Figure 11: Probability of playing the (U,U,D) equilibrium for the particle filter algorithm (dash line), stochastic fictitious play (solid line) and dynamic fictitious play (dot line) for the three player climbing hill game**

## 5. CONCLUSIONS

Distributed optimization problems can be formulated as a game. It is therefore natural to try to find optima by employing game-theoretical learning algorithms. The classic learning algorithm is fictitious play, but this is formed on an (incorrect) stationarity assumption. Therefore we have introduced a modification of fictitious play that uses Hidden Markov Models for the beliefs about opponent strategies.

We have examined the impact that the two parameters  $\tau$  and  $v$  have in particle filter algorithm results. Parameter  $v$  controls the memory of our algorithm. The larger  $v$  is the less memory our algorithm has. For values of  $v$  greater than 1.5 then our algorithm mimics our opponent's previous action. This can be derived in the way we propagate our beliefs about our opponent's strategy:  $Q_t = Q_{t-1} + \eta_t$ , where  $Q_0 \sim N(0, I)$  and  $\eta_t \sim N(0, v^2 I)$ . Introducing a noise greater than the scale of the  $Q$  values means that the values of  $Q_{t-1}$  lose their importance and the algorithm chooses all the  $Q_t$  values to give large  $\sigma_t(s_t)$ .

The other parameter is the temperature  $\tau$  of the Boltzman equation. This parameter has greatest impact on the range and the sharpness our estimation of the opponent's strategy will have. Our prediction of the opponent's strategy is computed based on the following formula:

$$\frac{e^{Q_t(s_t)/\tau}}{\sum_{\tilde{s} \in A} e^{Q_t(\tilde{s}_t)/\tau}}.$$

The value of  $\tau$  affects the value that the exponential will have and so the amplitude and the shape of our prediction. Large values of  $\tau$  make the differences between the probabilities of the actions smaller than they are, and small values of  $\tau$  have exactly the opposite result.

Finally, the last parameter of the algorithm  $\lambda$  has an effect on the randomization of stochastic fictitious play. Large values of  $\lambda$  allows randomization even if the player is not indifferent between the actions [7]. The player can choose an action which is not the best response to his beliefs with bigger probability. When this parameter tends to zero there is no randomization.

Stochastic fictitious play with particle filters to update its beliefs about the opponents strategy performs better in these three games than classic fictitious play. Although it was outperformed by dynamic fictitious play in the potential game, the speed of convergence to the equilibrium point in both variations of the climbing hill game is faster when using particle filters. We believe that for distributed optimization problems of realistic scale the particle filter algorithm will result in even faster convergence rates, through the ability to predict the future play of the other players instead of reacting to past play.

## 6. ACKNOWLEDGMENTS

The authors gratefully acknowledge BAE SYSTEMS and ESPSRC funding for the ALADDIN project (EP/C548051/1)

## 7. REFERENCES

- [1] G. Arslan, J. Marden, and J. Shamma. Autonomous vehicle-target assignment: A game theoretical formulation, October 2006.

- [2] S. Arulampalam, S. Maskell, N. Gordon, and T. Clapp. A tutorial on particle filters for online nonlinear/non-gaussian bayesian tracking. *IEEE Transactions on Signal Processing*, 50(2):174–188, 2002.
- [3] M. Bowling and M. Veloso. Multiagent learning using a variable learning rate. *Artificial Intelligence*, 136:215–250, 2002.
- [4] C. Claus and C. Boutilier. The dynamics of reinforcement learning in cooperative multiagent systems. In *Proceedings of the fifteenth nationalon Artificial intelligence*, 1998.
- [5] R. H. Crites and A. Barto. Improving elevator performance using reinforcement learning. In *Advances in Neural Information Processing Systems 8*, 1996.
- [6] D. Fudenberg and D. M. Kreps. Learning mixed equilibria. *Games and Economic Behavior*, 5:320–367, 1993.
- [7] D. Fudenberg and D. Levine. *The theory of Learning in Games*. The MIT Press, 1998.
- [8] D. Fudenberg and J. Tirole. *Game Theory*. MIT Press, 1991.
- [9] N. J. Gordon, D. J. Salmond, and A. Smith. Novel approach to nonlinear/non-gaussian bayesian state estimation. *IEEE Proceedings in Radar, Sonar and Navigation*, 140(2):107–113, 1993.
- [10] J. Harsanyi. Games with randomly distributed payoffs: a new rationale for mixed-strategy equilibrium points. *International Journal of Game Theory*, 2:1–23, 1973.
- [11] M. Littman. Markov games as a framework for multiagent reinforcement learning. In *Proceedings of the Eleventh International Conference of Machine learning*, 1994.
- [12] K. Miyasawa. On the convergence of learning process in a 2x2 non-zero-person game, 1961.
- [13] D. Monderer and L. Shapley. Potential games. *Games and Economic Behavior*, 14:124–143, 1996.
- [14] J. Nachbar. Evolutionary' selection dynamics in games: Convergence and limit properties. *International Journal of Game Theory*, 19:59–89, 1990.
- [15] J. Robinson. An iterative method of solving a game. *Annals of Mathematics*, 54:296–301, 1951.
- [16] L. Shapley. *In advances in Game theory*. Princeton: Princeton University, 1964.
- [17] W. Uther and M. Veloso. Adversarial reinforcement learning. Technical report, Carnegie Mellon University, 1997.
- [18] D. H. Wolpert. What information theory says about best response, binding contracts , and collective intelligence. In *Proceedings of WEHIA*, 2004.