

# Adaptive Forgetting Factor Fictitious Play

Michalis Smyrnakis <sup>\*1</sup> and David S. Leslie<sup>2</sup>

<sup>1</sup> School of Physics and Astronomy University of Manchester, UK.  
michalis.smyrnakis@manchester.ac.uk

<sup>2</sup> Department of Mathematics University of Bristol, UK.  
david.leslie@bris.ac.uk

**Abstract.** It is now well known that decentralised optimisation can be formulated as a potential game, and game-theoretical learning algorithms can be used to find an optimum. One of the most common learning techniques in game theory is fictitious play. However fictitious play is founded on an implicit assumption that opponents' strategies are stationary. We present a novel variation of fictitious play that allows the use of a more realistic model of opponent strategy. It uses a heuristic approach, from the online streaming data literature, to adaptively update the weights assigned to recently observed actions. We compare the results of the proposed algorithm with those of stochastic and geometric fictitious play in a simple strategic form game, a vehicle target assignment game and a disaster management problem. In all the tests the rate of convergence of the proposed algorithm was similar or better than the variations of fictitious play we compared it with. The new algorithm therefore improves the performance of game-theoretical learning in decentralised optimisation.

## 1 Introduction

Decentralised optimisation is a crucial component of sensor networks [1, 2], disaster management [3], traffic control [4] and scheduling [5]. In each of these domains a combination of computational and communication complexity render centralised optimisation approaches intractable. It is now well known that many decentralised optimisation problems can be formulated as a potential game [6–8]. Hence the optimisation problem can be recast in terms of finding a Nash equilibrium of a potential game. An iterative decentralised optimisation algorithm can therefore be considered a type of learning in games algorithm, and vice versa.

Fictitious play is the canonical example of learning in games [9]. Under fictitious play each player maintains some beliefs about his opponents' strategies, and based on these beliefs he chooses the action that maximises his expected reward. The players then update their beliefs about opponents' strategies after observing their actions. Fictitious play converges to Nash equilibrium for certain kinds of games [9, 10] but in practice this convergence can be very slow. This is

---

\* Michalis Smyrnakis is supported by The Engineering and Physical Sciences Research Council EPSRC (grant number EP/I005765/1).

because it implicitly assumes that the other players use a fixed strategy in the whole game by giving the same weight to every observed action.

In [11] this problem was addressed by using particle filters to predict opponents' strategies. The drawback of this approach is the computational cost of the particle filters that render difficult the application of this method in real life applications.

In this paper we propose an alternative method which uses a heuristic rule to adapt the weights of opponents' strategies by taking into account their recent actions. We observe empirically that this approach reduces the number of steps that fictitious play needs to converge to a solution, and hence the communications overhead between the distributed optimisers that is required to find a solution to the distributed optimisation problem. In addition the computational demand of the proposed algorithm is similar to the classic fictitious play algorithm.

The remainder of this paper is organised as follows. We start with a brief description of game theory, fictitious play and stochastic fictitious play. Section 3 introduces adaptive forgetting factor fictitious play (AFFFP). The impact of the algorithm's parameters on its performance is studied in Section 4. Section 5 presents the results of AFFFP for a climbing hill game, a vehicle target assignment game and a disaster management simulation scenario. We finish with a conclusion.

## 2 Background

In this section we introduce the relationship between potential games and decentralised optimisation, as well as the classical fictitious play learning algorithm.

### 2.1 Potential games and decentralised optimisation

A class of games which maps naturally to the decentralised optimisation framework is strategic form games. The elements of a strategic form game are [12]

- a set of players  $1, 2, \dots, I$ ,
- a set of actions  $s^i \in S^i$  for each player  $i \in I$ ,
- a set of joint actions,  $s = (s^1, s^2, \dots, s^I) \in S^1 \times S^2 \times \dots \times S^I = S$ ,
- a payoff function  $u^i : S \rightarrow \mathbf{R}$  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.

We will often write  $s = (s^i, s^{-i})$ , where  $s^i$  is the action of Player  $i$  and  $s^{-i}$  is the joint action of Player  $i$ 's opponents.

The rules that the players use to select the action that they will play in a game are called strategies. A player  $i$  chooses his actions according to a pure strategy when he selects his actions by using a deterministic rule. In the cases that he chooses an action based on a probability distribution then he acts according to a mixed strategy. If we denote the set of all the probability distributions over the

action space  $S^i$  as  $\Delta^i$ , then a mixed strategy of player  $i$  is an element  $\sigma^i \in \Delta^i$ . We define  $\Delta$  as the set product of all  $\Delta^i$ ,  $\Delta = \Delta^1 \times \dots \times \Delta^I$ . Then the joint mixed strategy  $\sigma = (\sigma^1, \dots, \sigma^I)$ , is defined as an element of  $\Delta$  and we will often write  $\sigma = (\sigma^i, \sigma^{-i})$  analogously to  $s = (s^i, s^{-i})$ . We will denote the expected utility a player  $i$  will gain if he chooses a strategy  $\sigma^i$  (resp.  $s^i$ ), when his opponents choose the joint strategy  $\sigma^{-i}$  as  $u^i(\sigma^i, \sigma^{-i})$  (resp.  $u^i(s^i, \sigma^{-i})$ ).

Many decision rules can be used by the players to choose their actions in a game. One of them is to choose their actions from a set of mixed strategies that maximises their expected utility given their beliefs about their opponents' strategies. When Player  $i$ 's opponents' strategies are  $\sigma^{-i}$  then the best response of player  $i$  is defined as:

$$BR^i(\sigma^{-i}) = \operatorname{argmax}_{\sigma^i \in \Delta^i} u^i(\sigma^i, \sigma^{-i}). \quad (1)$$

Nash [13], based on Kakutani's fixed point theorem, showed that every game has at least one equilibrium. This equilibrium is a strategy  $\hat{\sigma}$  that is a fixed point of the best response correspondence,  $\hat{\sigma}^i \in BR^i(\hat{\sigma}^{-i}) \forall i$ . Thus when a joint mixed strategy  $\hat{\sigma}$  is a Nash equilibrium then

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

Equation (2) implies that if a strategy  $\hat{\sigma}$  is a Nash equilibrium then it is not possible for a player to increase his utility by unilaterally changing his strategy. When all the players in a game select equilibrium actions using pure strategies then the equilibrium is referred as pure strategy Nash equilibrium.

A particularly useful category of games for multi-agent decision problems is the class of potential games [10, 8, 7]. The utility function of an exact potential game satisfies the following property:

$$u^i(s^i, s^{-i}) - u^i(\tilde{s}^i, s^{-i}) = \phi(s^i, s^{-i}) - \phi(\tilde{s}^i, s^{-i}) \quad (3)$$

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 actions  $s^i, \tilde{s}^i \in S^i$ . The potential function depicts the changes in the players' payoffs when they unilaterally change their actions. Every potential game has at least one pure strategy Nash equilibrium [10]. There may be more than one, but at any equilibrium no player can increase their reward, therefore the potential function, through a unilateral deviation.

Wonderful life utility [6, 7] is a method to design the individual utility functions of a potential game such that the global utility function of a decentralised optimisation problem acts as the potential function. Player  $i$ 's utility when a joint action  $s = (s^i, s^{-i})$  is performed, is the difference in global utility obtained by the player selecting action  $s^i$  in comparison with the global utility that would have been obtained if  $i$  had selected an (arbitrarily chosen) reference action  $s_0^i$ :

$$u^i(s^i, s^{-i}) = u_g(s^i, s^{-i}) - u_g(s_0^i, s^{-i}) \quad (4)$$

where  $u_g$  is the global utility function. Hence the decentralised optimisation problem can be cast as a potential game, and any algorithm that is proved to converge to Nash equilibria will converge to a joint action from which no player can increase the global reward through unilateral deviation.

## 2.2 Fictitious Play

Fictitious play is a widely used learning technique in game theory. In fictitious play each player chooses his action according to the best response to his beliefs about opponent's strategy.

Initially each player has some prior beliefs about the strategies that his opponents use to choose actions. The players, after each iteration, update their beliefs about their opponents' strategy and play again the best response according to their beliefs. More formally in the beginning of a game players maintain some arbitrary non-negative initial weight functions  $\kappa_0^j$ ,  $j = 1, \dots, I$  that are updated using the formula:

$$\kappa_t^j(s^j) = \kappa_{t-1}^j(s^j) + I_{s_t^j=s^j} \quad (5)$$

for each  $j$ , where  $I_{s_t^j=s^j} = \begin{cases} 1 & \text{if } s_t^j = s^j \\ 0 & \text{otherwise.} \end{cases}$

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

$$\sigma_t^j(s^j) = \frac{\kappa_t^j(s^j)}{\sum_{s^j} \kappa_t^j(s^j)}. \quad (6)$$

Equations (5) and (6) are equivalent to:

$$\sigma_t^j(s^j) = \left(1 - \frac{1}{t^j}\right) \sigma_{t-1}^j(s^j) + \frac{1}{t^j} I_{s_t^j=s^j} \quad (7)$$

where  $t^j = t + \sum_{s^j \in S^j} \kappa_0^j(s^j)$ . Player  $i$  chooses an action which maximises his expected payoffs given his beliefs about his opponents' strategies.

The main purpose of a learning algorithm like fictitious play is to converge to a set of strategies that are a Nash equilibrium. For classic fictitious play (7) it has been proved [9] that if  $\sigma$  is a strict Nash equilibrium and it is played at time  $t$  then it will be played for all further iterations of the game. Also 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 [14], zero sum games [15], games that can be solved using iterative dominance [16] and potential games [10]. There are also games where fictitious play does not converge to a Nash equilibrium. Instead it can become trapped in a limit cycle whose period is increasing through time. An example of such a game is Shapley's game [17].

A player  $i$  that uses the classic fictitious play algorithm uses best responses to his beliefs to choose his actions, so he chooses his actions  $s^i$  from his pure strategy space  $S^i$ . Randomisation is allowed only in the case that players are indifferent between his available actions, but it is very rare in generic payoff strategic form

games for a player to be indifferent between the available actions [18]. Stochastic fictitious play is a variation of fictitious play where players use mixed strategies in order to choose actions. This variation was originally introduced to allow convergence of players' strategies to a mixed strategy Nash equilibrium [9] but has the additional advantage of introducing exploration into the process.

The most common form of smooth best response of Player  $i$ ,  $\overline{BR}^i(\sigma^{-i})$ , is the following [9]:

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

where  $\xi$  is the randomisation parameter. When the value of  $\xi$  is close to zero, a  $\overline{BR}$  is close to  $BR$  and players exploit their action space, whereas large values of  $\xi$  result in complete randomisation [9].

Stochastic fictitious play is a modification of fictitious play under which Player  $i$  uses  $\overline{BR}^i(\sigma_t^{-i})$  to randomly select an action instead of selecting a best response action  $BR^i(\sigma_t^{-i})$ . We reinforce the fact that the difference between classic fictitious play and stochastic fictitious play is in the decision rule the players use to choose the actions. The updating rule (7) that is used to update the beliefs of the opponents' strategies are the same in both algorithms.

When Player  $i$  uses equation (7) to update the beliefs about opponents' strategies he treats the environment of the game as stationary and implicitly assumes that the actions of the players are sampled from a fixed probability distribution [9]. Therefore recent observations have the same weight as initial ones. This approach leads to poor adaptation when other players change their strategies.

A variation of fictitious play that treats the opponents' strategies as dynamic and places greater weights on recent observations while we calculate each action's probability is geometric fictitious play, introduced in [9]. According to this variation of fictitious play the estimation of each opponent's probability to play an action  $s^j$  is evaluated using the formula:

$$\sigma_t^j(s^j) = (1 - z)\sigma_{t-1}^j(s^j) + zI_{s_t^j=s^j} \quad (9)$$

where  $z \in (0, 1)$  is a constant.

In Section 3 we introduce a new variant of fictitious play in which the constant  $z$  is automatically adapted in response to the observations of opponent strategy.

### 3 Adaptive forgetting factor fictitious play

The objective of players when they maintain beliefs  $\sigma_t^{-i}$  is to estimate the mixed strategy of opponents. However consider streaming data where in each time step a new observation arrives and it belongs to one of  $J$  available classes [19]. When the objective is to estimate the probability of each class given the observed data, this objective can be expressed as the fitting of a multinomial distribution to the

observed data stream. If a fixed multinomial distribution over time is assumed, then its parameters can be estimated using the empirical frequencies of the previously observed data. This is exactly the strategy estimation described in Section 2.2. But in real life applications it is rare to observe a data stream from a constant distribution. Hence there is a need for the learning algorithm to adapt to the distribution that the data currently follow. This is similar to iterative games, where we expect that all players update their strategies simultaneously.

An approach that is widely used in the streaming data literature to handle changes in data distributions is forgetting. This suggests that recent observations have greater impact on the estimation of the algorithm’s parameters than the older ones. Two methods of forgetting are commonly used: window based methods and resetting. Salgado et.al [20] showed that when abrupt changes (jumps) are detected then the optimal policy is to reset the parameters of the algorithm. In the case of smooth changes (drift) in the data stream’s distribution, a solution is to use only a segment of the data (a window). The simplest form of window based methods uses a specific segment size constantly; there are also approaches that adaptively change the size of the window but they are more complicated. Some examples of algorithms that use window based methods are [21–23].

Another method is to introduce forgetting, which is also used in geometric fictitious play (9), to discount the old information by giving higher weights to the recent observations. When the discount parameter is fixed it is necessary to know a priori the distribution of data and the way that they evolve through time due to the fact that we must choose the forgetting factor in advance. In addition the performance of the approximation when there are changes that result from a jump or non-constant drift is poor for a fixed forgetting factor. For those reasons this methodology has serious limitations.

A more sophisticated solution is the use of a forgetting factor that takes into account the recent data and the previously estimated parameters of the model and adapts to observed changes in the data distribution. Such a forgetting factor was proposed by Haykin [24] in the case of recursive least squares filtering problems. In [24] the objective was the minimisation of the mean square error of a cost function that depends on an exponential weighting factor  $\lambda$ . This forgetting factor is then recursively updated using gradient descent of the forgetting factor,  $\lambda$ , with respect to the residual errors of the algorithm. Anagnostopoulos [19] proposed a generalisation of this method in the context of online streaming data from a generalised linear model according to which the forgetting factors are adaptively changed by using gradient ascent of the log-likelihood of the new data point.

In the streaming data context, after  $t$  time intervals we observe a sequence of data  $x_1, \dots, x_t$  and we fit a model  $f(\theta_t|x_{1:t})$ , where  $\theta_t$  are the model’s parameters at time  $t$ . Note that the parameters of the model,  $\theta_t$ , depend on the observed data stream  $x_{1:t}$  and the forgetting factors  $\lambda_t$ . Since the estimated model parameters depend on  $\lambda_t$  we will write  $\theta_t(\lambda_t)$ . The log-likelihood of the data that will arrive at time  $t+1$ ,  $x_{t+1}$ , given the parameters of the model at time  $t$  will be denoted as

$\mathcal{L}(x_{t+1}; \theta_t(\lambda_t))$ . Then the update of the forgetting factor  $\lambda_{t+1}$  can be expressed as:

$$\lambda_{t+1} = \lambda_t + \gamma \frac{\partial \mathcal{L}(x_{t+1}; \theta_t(\lambda_t))}{\partial \lambda} \quad (10)$$

where  $\gamma$  is the learning rate parameter of the gradient ascent algorithm.

As in [19], we can apply the forgetting factor of equation (10) in the case of fitting a multinomial distribution to streaming data. This will result a new update rule that players can use, instead of the classic fictitious play update rule (7), to maintain beliefs about opponents' strategies.

In classic fictitious play the weight function (5) places the same weight on every observed action. In particular  $\kappa_t^j(s^j)$  denotes the number of times that player  $j$  has played the action  $s^j$  in the game. To introduce forgetting the impact of the previously observed actions in the weight function will be discounted by a factor  $\lambda_{t-1}$ . Such a weight function can be written as:

$$\kappa_t^j(s^j) = \lambda_{t-1}^j \kappa_{t-1}^j(s^j) + I_{s_{t-1}^j=s^j} \quad (11)$$

where  $I_{s_{t-1}^j=s^j}$  is the same identity function as in (7). To normalise we set  $n_t = \sum_{s^j \in S^j} \kappa_t^j(s^j)$ . From the definition of  $\kappa_t^j(s^j)$  we can use the following recursion to evaluate  $n_t^j$

$$n_t^j = \lambda_{t-1}^j n_{t-1}^j + 1. \quad (12)$$

Then player  $i$ 's beliefs about his opponent  $j$ 's probability of playing action  $s^j$  will be:

$$\sigma_t^j(s^j) = \frac{\kappa_t^j(s^j)}{n_t^j}. \quad (13)$$

Similarly to the case of geometric fictitious play  $0 < \lambda_t \leq 1$ . Moreover when the value of  $\lambda_t$  is close to zero this results in very fast adaptation and when  $\lambda_t = 0$  the players are myopic, and thus they respond to the last action of their opponents. On the other hand when  $\lambda_t = 1$  this results in the classic fictitious play update rule.

From this point on in we will only consider inference over a single opponent mixed strategy in fictitious play. In the case of multiple opponents separate estimates are formed identically and independently for each opponent. We will therefore drop all dependence on player  $i$ , and write  $s_t$ ,  $\sigma_t$  and  $\kappa_t(s)$  for the opponent's action, strategy and weight function respectively.

The value of  $\lambda$  should be updated in order to have adaptive forgetting factors. Initially we have to evaluate the log-likelihood of the recently observed action  $s_t$  given the beliefs of the opponents strategies. The log-likelihood is of the following form:

$$\mathcal{L}(s_t; \sigma_{t-1}) = \ln \sigma_{t-1}(s_t) \quad (14)$$

When we replace  $\sigma_{t-1}(s_t)$  with its equivalent from (13) the log-likelihood can be written as:

$$\mathcal{L}(s_t; \sigma_{t-1}) = \ln \left( \frac{\kappa_{t-1}(s_t)}{n_{t-1}} \right) = \ln \kappa_{t-1}(s_t) - \ln n_{t-1} \quad (15)$$

In order to estimate the update of  $\lambda$ , equation (10), the evaluation of the log-likelihood's derivative with respect to  $\lambda$  is required. The terms  $\kappa_t$  and  $n_t$  both depend on  $\lambda$ . Hence the derivative of (15) is:

$$\frac{\partial \mathcal{L}(s_t; \sigma_{t-1})}{\partial \lambda} = \frac{1}{\kappa_{t-1}(s_t)} \frac{\partial}{\partial \lambda} \kappa_{t-1}(s_t) - \frac{1}{n_{t-1}} \frac{\partial}{\partial \lambda} n_{t-1} \quad (16)$$

Note that  $\kappa_t(s_t) = \lambda_{t-1} \kappa_{t-1}(s_t) + I_{s_{t-1}^j = s^j}$ , so

$$\frac{\partial}{\partial \lambda} \kappa_t(s) |_{\lambda = \lambda_{t-1}} = \kappa_{t-1}(s) + \lambda_{t-1} \frac{\partial}{\partial \lambda} \kappa_{t-1}(s) |_{\lambda = \lambda_{t-1}} \quad (17)$$

and similarly

$$\frac{\partial}{\partial \lambda} n_t |_{\lambda = \lambda_{t-1}} = n_{t-1} + \lambda_{t-1} \frac{\partial}{\partial \lambda} n_{t-1} |_{\lambda = \lambda_{t-1}} \quad (18)$$

We can use equations (17) and (18) to recursively estimate  $\frac{\partial}{\partial \lambda} \kappa_{t-1}(s)$  for each  $s$  and  $\frac{\partial}{\partial \lambda} n_{t-1}$  and hence calculate  $\frac{\partial}{\partial \lambda} \mathcal{L}(s_t; \sigma_{t-1})$ . Summarising we can evaluate the adaptive forgetting factor  $\lambda_t$  as follows:

$$\lambda_t = \lambda_{t-1} + \gamma \left( \frac{1}{\kappa_{t-1}(s)} \frac{\partial}{\partial \lambda} \kappa_{t-1}(s) - \frac{1}{n_{t-1}} \frac{\partial}{\partial \lambda} n_{t-1} \right) \quad (19)$$

To ensure that  $\lambda_t$  remains in  $(0, 1)$  we truncate it to this interval whenever it leaves.

After updating their beliefs players can choose their actions by choosing either a best response to their beliefs of their opponents strategies or a smooth best response. Table 1 summarises the algorithm of adaptive forgetting factor fictitious play.

At time  $t$ , each player carries out the following

1. Updates the weights  $\kappa_t^j(s^j) = \lambda_{t-1}^j \kappa_{t-1}^j(s^j) + I_{s_{t-1}^j = s^j}$
2. Update  $\frac{\partial}{\partial \lambda} \kappa_{t-1}^j(s)$  and  $\frac{\partial}{\partial \lambda} n_{t-1}^j$  using equations (17) and (18)
3. Based on the weights of step 1 each player updates his beliefs about his opponents strategies using  $\sigma_t^j(s^j) = \frac{\kappa_t^j(s^j)}{n_t^j}$ , where  $n_t^j = \lambda_{t-1}^j n_{t-1}^j + 1$ .
4. Choose an action based on the beliefs of step 3 according either to best response,  $BR$ , or to smooth best response  $\overline{BR}$
5. Observe opponent's action  $s_t^j$
6. Update the forgetting factor using:  $\lambda_t^j = \lambda_{t-1}^j + \gamma \left( \frac{1}{\kappa_{t-1}^j(s)} \frac{\partial}{\partial \lambda} \kappa_{t-1}^j(s) - \frac{1}{n_{t-1}^j} \frac{\partial}{\partial \lambda} n_{t-1}^j \right)$

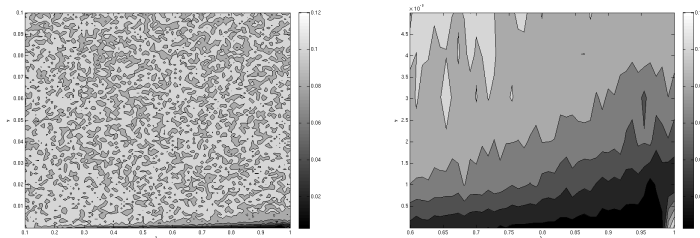
**Table 1.** Adaptive forgetting factor fictitious play algorithm



## 4 AFFFP parameters

The adaptive rule that we choose to update the forgetting factor  $\lambda$  is based on the gradient ascent algorithm. It is well known that different initial values of an algorithm's parameter and learning rates  $\gamma$  can lead to poor results of the gradient ascent algorithm [25]. This is because very small values of  $\gamma$  lead to poor exploration of the space and thus the gradient ascent algorithm can be trapped in an area where the solution is not optimal, whereas large values of  $\gamma$  can result in big jumps that will lead the algorithm away from the area of the optimum solution. Thus we should evaluate the performance of adaptive forgetting factor fictitious play for different combinations of the step size parameter  $\gamma$  and initial values  $\lambda_0$ .

We employed a toy example, where a single opponent chooses his actions using a mixed strategy which has a sinusoidal form, a situation which corresponds to smooth changes in the data distribution of online streaming data. The opponent uses a strategy of the following form over the  $t = 1, 2, \dots, 1000$  iterations of the game:  $\sigma_t(1) = \frac{\cos \frac{2\pi t}{\beta} + 1}{2} = 1 - \sigma_t(2)$ , where  $\beta = 1000$ . We repeated this example 100 times for each combination of  $\gamma$  and  $\lambda_0$ . Each time we measured the mean square error of the estimated strategy against the real one. The range of  $\gamma$  and  $\lambda_0$  was  $10^{-6} \leq \gamma \leq 10^{-1}$  and  $10^{-1} \leq \lambda_0 \leq 1$  respectively.



(a) Contour plot of mean square error when the range of  $\gamma$  and  $\lambda$  is  $10^{-6} \leq \gamma \leq 10^{-1}$  and  $10^{-1} \leq \lambda \leq 1$  respectively.  
 (b) Contour plot of mean square error when the range of  $\gamma$  and  $\lambda$  is  $10^{-6} \leq \gamma \leq 5 \times 10^{-3}$  and  $0.6 \leq \lambda \leq 1$  respectively.

**Fig. 1.** Contour plot of mean square error

The average mean square error for all the combinations of  $\gamma$  and  $\lambda_0$  is depicted on Figure 1. The mean square error is minimised in the dark area of the contour plot. In Figure 1(a) we observe that when  $\gamma$  is less than  $10^{-3}$  and  $\lambda_0$  is greater than 0.6 the mean square error is minimised. In Figure 1(b) we reduce the range of  $\gamma$  and  $\lambda_0$  to be  $10^{-6} \leq \gamma \leq 5 \times 10^{-3}$  and  $0.6 \leq \lambda_0 \leq 1$ , respectively. Values of  $\lambda_0$  greater than 0.75 result in estimators with small mean square error, for certain values of  $\gamma$ , and as the value of  $\lambda_0$  approaches 0.95 so it is minimised for a wider range of learning rates. We also observe that when  $\lambda_0$  is greater than 0.98 and

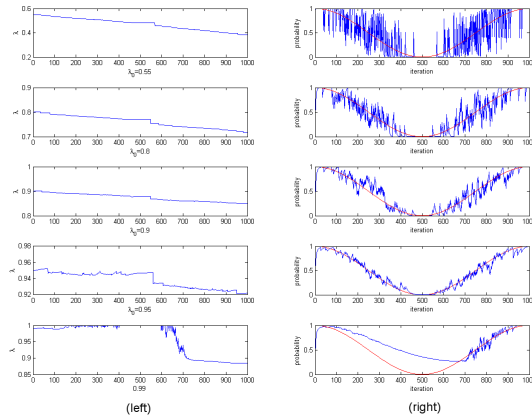
as we approach 1 then the mean square error increases. This suggests that for values of  $\lambda_0$  greater than 0.98 the value of  $\lambda$  approaches 1 very fast and thus the algorithm behaves like the classic fictitious play update rule. In contrast when  $\lambda_0$  is less than 0.75 then we introduce big discounts to the previously observed actions from the beginning of the game and the players easily use strategies that are reactions to their opponent's randomisation. In addition, independently from the initial value of  $\lambda_0$ , when the learning rate  $\gamma$ , is greater than 0.001 the algorithm results in poor estimations of the opponent's strategy. This is because for  $\gamma$  greater than 0.001 the step that a player moves towards the maximum of the log-likelihood is very large and that results in values of  $\lambda$  which are close either to zero or to one. So the player uses either the classic fictitious play update rule or responds to his opponent's last observed action.

We further examine the relationship between the performance of adaptive forgetting factor fictitious play and the sequence of  $\lambda$ 's in the drift toy example with respect to the initial values of the parameters  $\lambda_0$  and  $\gamma$ . We use two instances of the drift toy example. In the first one we set a fixed value of parameter  $\gamma = 10^{-4}$  and examine the performance of our algorithm and the evolution of  $\lambda$  during the game for different values of  $\lambda_0 = \{0.55, 0.8, 0.9, 0.95, 0.99\}$ . In the second one we fix  $\lambda_0$  and examined the results of the algorithm for different values of  $\gamma = \{10^{-6}, 10^{-5}, 5 \cdot 10^{-4}, 10^{-4}, 10^{-3}\}$ . Figures 2 and 3 depict the results of the case of fixed  $\gamma$  and  $\lambda_0$ , respectively. Each row of these figures consists of two plots for the same set of parameters,  $\gamma$  and  $\lambda_0$ . The left figure shows the evolution of  $\lambda$  during the game and the right one depicts the pre-specified strategy of the opponent and its corresponding prediction.

As we observe in Figure 2, when we set  $\lambda_0 = 0.55$  the tracking of the opponent's strategy was affected by his randomisation. The value of  $\lambda$  constantly decreases which results in giving higher weights to the recently observed actions even if they are a consequence of randomisation. When we increase the value of  $\lambda_0$  to 0.8 the results are improving. When we increase  $\lambda_0$  to 0.90 or 0.95 the resulting sequence of  $\lambda$ 's does not affect the tracking of opponent's strategy. On the other hand when we increase the value of  $\lambda_0$  to 0.99 the value of  $\lambda$  is very close to 1 for many iterations which result in poor approximation for the same reasons that the classic fictitious play update rule fails to capture smooth changes in opponent's strategy. When  $\lambda_0$  is decreased to 0.9 the approximation of opponent's strategy improves significantly.

Figure 3 depicts the results when  $\lambda_0 = 0.95$  for different values of parameter  $\gamma$ . We observe that high values of  $\gamma$  ( $\gamma = 10^{-3}, 5 \cdot 10^{-4}$ ) result in big changes in the value of  $\lambda$  and that affects the quality of the approximation. On the other hand when we use very small values of  $\gamma$ ,  $\gamma = 10^{-5}$ , or  $\gamma = 10^{-6}$ , it leads to very small deviations from  $\lambda_0$ . The good approximation results of opponent's strategy that we observe for those two values of  $\gamma$  are because of the initial value  $\lambda_0$ . In this scenario if we fix the value of  $\lambda = 0.95$  during the whole game we will also have a good approximation. But in real life applications it is impossible to choose so efficiently the value of  $\lambda_0$ . When  $\gamma = 10^{-4}$  we observe changes in the

values of  $\lambda$ , which are not so sudden as the ones for  $\gamma = 10^{-3}$  or  $\gamma = 5 \cdot 10^{-4}$ , that lead to good approximation of opponent's strategy.

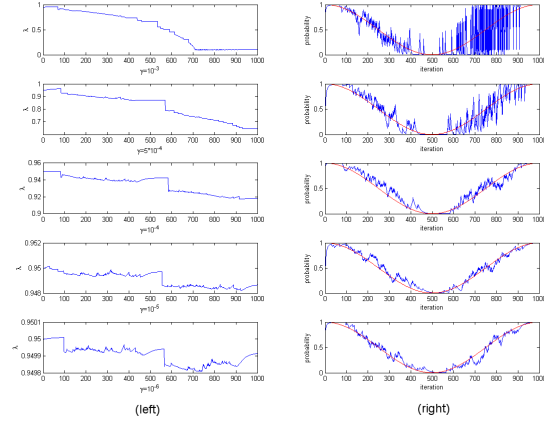


**Fig. 2.** Evolution of  $\lambda$  and tracking of  $\sigma_t(1)$  when the true strategies are mixed when  $\gamma$  is fixed. The pre-specified strategy of the opponent and its prediction are depicted as the red and blue line respectively.

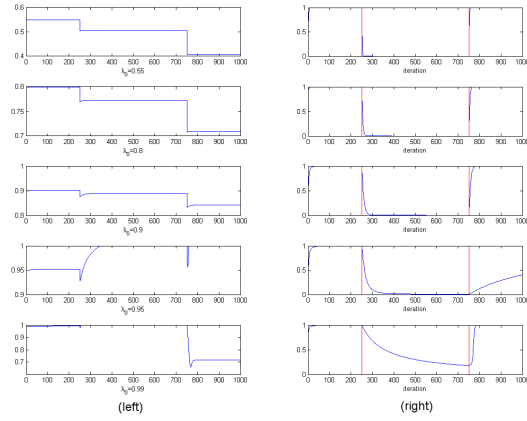
We also performed simulations for a case where jumps occur. In this example we used a game of 1000 iterations and two available actions, 1 and 2. The opponent played action 1 with probability  $\sigma_t^2(1) = 1$  during the first 250 and the last 250 iterations of the game and for the remaining iterations of the game  $\sigma_t^2(1) = 0$ . The probability of the second action can be calculated by using  $\sigma_t^2(2) = 1 - \sigma_t^2(1)$ . The results for the case of fixed  $\gamma$  and  $\lambda_0$  are depicted in Figures 4 and 5, respectively.

When abrupt changes occur the different values of  $\gamma$  do not affect the performance of the algorithm as we observe in Figure 5. On the contrary the initial value of  $\lambda$  affects the estimation of the jumps in the opponent's strategies. As we observe in Figure 4 the opponent's strategy approximation and the evolution of  $\lambda$  are similar when  $\lambda_0$  is equal to 0.55, 0.8 and 0.9 respectively. In those three cases when a jump is observed, there is a drop in the value of  $\lambda$ , then  $\lambda$  slightly increases and finally it remains constant until the next jump occurs.

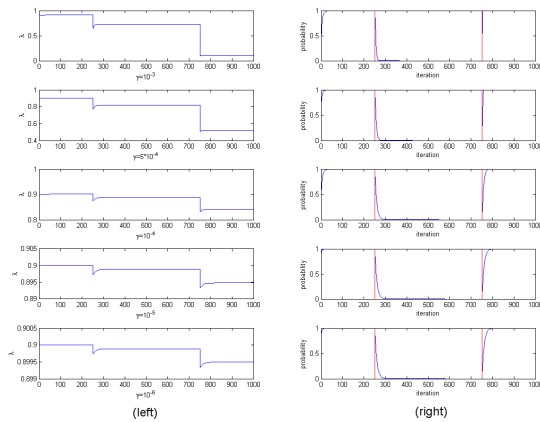
The sequences of  $\lambda$  and the approximation results of the last two cases,  $\lambda_0 = 0.95$  and  $\lambda_0 = 0.99$  are different from the 3 cases we described above. In both of them the opponent's strategy tracking is good at the first 250 iterations, but afterwards these two examples have the opposite behaviour. In the example where  $\lambda_0 = 0.95$  the opponent's strategy is correctly approximated when  $\sigma_t(1) = 0$ . Because of the high weights of the previously observed actions, the likelihood needs a large number of iterations to become constant and thus  $\lambda$  becomes equal to 1. Then the adaptive forgetting factor fictitious play process becomes identical



**Fig. 3.** Evolution of  $\lambda$  and tracking of  $\sigma_t(1)$  when the true strategies are mixed when  $\lambda_0$  is fixed. The pre-specified strategy of the opponent and its prediction are depicted as the red and blue line respectively



**Fig. 4.** Evolution of  $\lambda$  and tracking of  $\sigma_t(1)$  when the true strategies are pure when  $\gamma$  is fixed. The pre-specified strategy of the opponent and its prediction are depicted as the red and blue line respectively.



**Fig. 5.** Evolution of  $\lambda$  and tracking of  $\sigma_t(1)$  when the true strategies are pure when  $\lambda_0$  is fixed. The pre-specified strategy of the opponent and its prediction are depicted as the red and blue line respectively.

to classic fictitious play and fails to adapt to the second jump. When  $\lambda_0$  is equal to 0.99 adaptive forgetting factor fictitious play fails to adapt the estimation of opponent's strategy to the first jump. But when the second jump occurs, the likelihood of action 1 is small since action 2 is played for 500 consecutive iterations, and a drop in the value of  $\lambda$  is observed which resulted in adaptation to the change of opponent's strategy.

By taking into account the above results we observe that  $\gamma = 10^{-4}$  and  $0.8 \leq \lambda_0 \leq 0.9$  leads to useful approximations when we consider cases where both smooth and abrupt changes in opponent's strategy are possible to happen. In the remainder of the article we set  $\gamma = 10^{-4}$  and  $\lambda_0 = 0.8$ .

## 5 Results

### 5.1 Climbing hill game

We initially compared the performance of the proposed algorithm with the results of geometric and classic stochastic fictitious play in a three player climbing hill game. This game which is depicted in Table 2, generalises the climbing hill game that was presented in [26] and exhibits a long best response path from the risk-dominant joint mixed action (D,D,U) to the Nash equilibrium.

We present the results of 1000 replications of a learning episode of 1000 iterations for each game. 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.

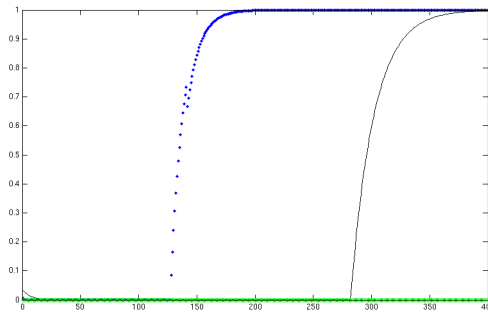
The value of the learning parameter  $z$  of geometric fictitious play was set to 0.1. We selected this value of  $z$  on the premise that the algorithm has the

	U	M	D	U	M	D	U	M	D
U	0	0	0	-300	70	80	<b>100</b>	-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 2.** Climbing hill game with three players. Player 1 selects rows, Player 2 selects columns, and Player 3 selects the matrix. The global reward depicted in the matrices, is received by all players. The unique Nash equilibrium is in bold.

best results in the tracking experiment with pre-specified opponents strategy that we used in Section 4 to select the parameters of AFFFP. Thus we will use this learning rate in all the simulations that we present in the rest of this article. For all algorithms we used smooth best responses (8) with randomisation parameter  $\xi$  in the smooth best response function equal to 1, allowing the same randomisation for all algorithms.

Adaptive forgetting factor fictitious play performed better than both geometric and stochastic fictitious play. The overall mean global payoff was 95.26 for AFFFP whereas the respective payoffs for geometric and stochastic fictitious play were 91.7 and 70.3. Stochastic fictitious play didn't converge to the Nash equilibrium after 1000 replications. Also when we are concerned about the speed of convergence the proposed variations of fictitious play outperform geometric fictitious play. This can be seen if we reduce the iterations of the game to 200. Then the overall mean payoffs of AFFFP is 90.12 when for geometric fictitious play it is 63.12. This is because adaptive forgetting factor fictitious play requires approximately 100 iterations to reach the Nash equilibrium, when geometric fictitious play needs at least 300. This difference is depicted in Figure 6.



**Fig. 6.** Probability of playing the (U,U,D) equilibrium for one run of each of AFFFP (blue dot line), geometric fictitious play (black solid line) and stochastic fictitious play (green diamond line) for the three player climbing hill game.

## 5.2 Vehicle target assignment game

We also compared the performance of the proposed variation of fictitious play against the results of geometric fictitious play in the vehicle target assignment game that is described in [7]. In this game agents should coordinate to achieve a common goal which is to maximise the total value of the targets that are destroyed. In particular in a specific area we place  $I$  vehicles and  $J$  targets. For each vehicle  $i$ , its available actions are simply the targets that are available to engage. Each vehicle can choose only one target to engage but a target can be engaged by many vehicles. The probability that player  $i$  has to destroy a target  $j$  is  $p_{ij}$  if it chooses to engage target  $j$ . We assume that the probability each agent has to destroy a target is independent of the actions of the other agents, and the target is destroyed if any one agent successfully destroys it, so the probability a target  $j$  is destroyed by the vehicles that engage it is  $1 - \prod_{i:s^i=j} (1 - p_{ij})$ . Each of the targets has a different value  $V_j$ . The expected utility that is produced from the target  $j$  is the product of its value  $V_j$  and the probability it has to be destroyed by the vehicles that engage it. More formally we can express the utility that is produced from target  $j$  as:

$$U_j(s) = V_j(1 - \prod_{i:s^i=j} (1 - p_{ij})) \quad (20)$$

The global utility is then the sum of the utilities of each target:

$$u_g(s) = \sum_j U_j(s). \quad (21)$$

Wonderfull life utility was used to to evaluate each vehicle's payoff. Then the utility that a vehicle  $i$  receives after engage a target  $j$ ,  $s^i = j$ , is

$$u_i(s^i, s^{-i}) = U_j(s^i, s^{-i}) - U_j(s_0^i, s^{-i}) = \quad (22)$$

where  $s_0^i$  was set to be the greedy action of player  $i$ :  $s_0^i = \operatorname{argmax}_j V_j p_{ij}$ .

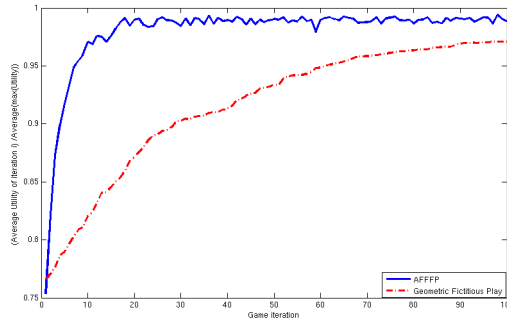
In our simulations we used thirty vehicles and thirty targets that were placed uniformly at random in a unit square. The probability of a vehicle  $i$  to destroy a target  $j$  is proportional to the inverse of its distance from this target  $1/d_{ij}$ . The values of the targets are independently sampled from a uniform distribution with range in  $[0 100]$ .

The vehicles had to “negotiate” with the other vehicles (players) for a fixed number of negotiation steps before they choose a target to engage. A negotiation step begins with each player choosing a target to engage and it ends by the agents exchanging this information with the others, and updating their beliefs about their opponents' strategies based on this information. The target that each vehicle will choose in the game will be his action at the final negotiation step.

Figure 7 depicts the average results for 100 instances of the game for the two algorithms, AFFFP and geometric fictitious play. For each instance, both

algorithms run for 100 negotiation steps. To be able to average across the 100 instances we normalise the scores of an instance by the highest observed score for that instance (since some instances will have greater available score than others). As in the strategic form game, we set the randomisation parameter  $\xi$  in the smooth best response function equal to 1 for both algorithms.

In Figure 7 we observe that AFFFP result in a better solution on average than geometric fictitious play. Furthermore geometric fictitious play needs more iterations to reach the area where its reward is maximised than AFFFP.



**Fig. 7.** Utility of AFFFP (dotted line) and geometric fictitious play (dashed line) for the vehicle target assignment game.

### 5.3 Disaster management scenario

Finally we test our algorithm in a disaster management scenario as described in [27]. Consider the case where a natural disaster has happened (an earthquake for example) and because of this  $N_I$  simultaneous incidents occurred in different areas of a town. In each incident  $j$ , a different number of people  $N_p(j)$  are injured. The town has a specific number of ambulances  $N_{amb}$  available that are able to collect the injured people. An ambulance  $i$  can be at the area of incident  $j$  in time  $T_{ij}$  and has capacity  $c_i$ . We will assume that the total capacity of the ambulances is larger than the number of injured people. Our aim is to allocate the ambulances to the incidents in such a way that the average time  $\frac{1}{N_{amb}} \sum_{i:s^i=j} T_{ij}$  that the ambulances need to reach the incidents is minimised while all the people that are engaged in the incident will be saved. Then we can formulate this scenario as follows. Each of the  $N_{amb}$  players should choose one of the  $N_I$  incidents as actions. The utility to the system of an allocation is:

$$u_g(s) = -\frac{1}{N_{amb}} \sum_{j=1}^{N_I} \sum_{i:s^i=j} T_{ij} - \sum_{j=1}^{N_I} \max(0, N_p(j) - \sum_{i:s^i=j} c_i) \quad (23)$$



where  $i = 1, \dots, N_{amb}$  and  $s^i$  is the action of player  $i$ . The emergency units have to “negotiate” with each other and choose the incident which they will be allocated to, using a variant of fictitious play.

The first component of the utility function expresses the first aim, to allocate the ambulance to an incident as fast as possible. Thus the agents have to choose an incident with small  $T_{ij}$ . The second objective, which is to save all the injured people that are engaged in an incident, is expressed as the second component of the utility function. It is a penalty factor that adds to the average time the number of the people that were not able to be saved. Like the vehicle target assignment game each player can choose only one incident to help, but in each incident more than one player can provide his help.

We follow [27] and consider simulations with 3 and 5 incidents, and 10, 15 and 20 available ambulances. We run 200 trials for each of the combinations of ambulances and incidents, for each algorithm. Since this scenario is NP-complete [27] our aim is not to find the optimal solution, but to reach a sufficient or a near optimal solution. Furthermore the algorithm we present here is “any-time”, since the system utility generally increases as the time goes on, and therefore interruption before termination results in good, if not optimal actions. In each of the 200 trials the time  $T_{ij}$  that an ambulance needs to reach an incident is a random number uniformly distributed between zero and one. The capacity of each ambulance is an integer uniformly distributed between one and four. Finally the total number of injured people that are involved in each incident is a uniformly distributed integer between  $\frac{c_t}{2 \cdot N_I}$  and  $\frac{c_t}{N_I}$ , where  $c_t$  is the total capacity of the emergency units  $c_t = \sum_{i=1}^{N_{amb}} c_i$ .

In each trial we allow 200 negotiation steps. In this scenario because of the utility function structure, a big randomisation parameter  $\xi$  in the smooth best response function can easily lead to unnecessary randomisation. For that reason we set  $\xi$  to 0.01 for both algorithms which results in a decision rule which approximates best response. The learning rate and the initial value of  $\lambda$  in adaptive forgetting factor is set to  $10^{-4}$  and 0.8 respectively.

We use the same performance measures as [27] to test the performance of our algorithms. We compared the solution of our algorithm against the centralised solution which can be obtained using binary integer programming. In particular we compared the solution of our algorithm against the one we obtain by using Matlab’s *bintprog* algorithm, which uses a branch and bound algorithm that is based on linear programming relaxation [28–30]. To compare the result of these two algorithms we use the ratio  $\frac{f_{fp}}{f_{opt}}$ , where  $f_{fp}$  is the utility that the agents could gain if they used the variations of fictitious play we propose and  $f_{opt}$  is the utility that the agents should gain if they were using the solution of *bintprog*. Thus values of the ratio smaller than one mean that the proposed variations of fictitious play perform better than *bintprog*, and values of the ratio larger than one mean that the proposed variations of fictitious play perform worst than *bintprog*. Furthermore we measured the percentage of the instances in which all the casualties are rescued, and the overall percentage of people that are rescued.

		%complete	% saved	$f_{fp}/f_{opt}$
3 incidents	10 ambulances	82.0	92.84	1.2702
	15 ambulances	77.5	89.88	1.2624
	20 ambulances	81.0	90.59	1.2058
5 incidents	10 ambulances	91.0	98.54	1.6631
	15 ambulances	90.5	98.28	1.5088
	20 ambulances	83.0	93.6	1.4251

**Table 3.** Results of adaptive forgetting factor fictitious play after 200 negotiation steps for the three performance measures.

		%complete	% saved	$f_{fp}/f_{opt}$
3 incidents	10 ambulances	95,5	99,74	1.2970
	15 ambulances	74,5	88.24	1.2965
	20 ambulances	60.55	87.9	1.2779
5 incidents	10 ambulances	94.5	99.68	1.8587
	15 ambulances	79.0	88.28	1.7443
	20 ambulances	48.50	86.54	1.8545

**Table 4.** Results of geometric fictitious play after 200 negotiation steps for the three performance measures.

Tables 3 and 4 present the results we have obtained in the last step of negotiations between the ambulances for the disaster management scenario when they use adaptive forgetting factor and geometric fictitious play respectively, to coordinate.

The total percentage of the people that were saved and the ratio of  $f_{fp}/f_{opt}$  were similar within the groups of 3 and 5 incidents when the adaptive forgetting factor fictitious play algorithm were used. Regarding the percentage of the trials in which all people were saved, we can observe that as we increase the complexity of the scenario, hence the number of ambulances, the performance of adaptive forgetting factor fictitious play is decreasing.

When we compare the results of the two algorithms we can observe that in both cases of 3 and 5 incidents respectively adaptive forgetting factor fictitious play perform better than geometric fictitious play when the scenarios included more ambulances, and therefore were more complicated. Especially in the case of the 20 ambulances the difference when we consider the number of the cases where all the casualties were collected from the incidents, was greater than 20%.

The differences we can observe from *bintprog*'s centralised solution, for both algorithms, can be explained from the structure of the utility function (23). The first component of the utility is a number between zero and one since it is the average of the times,  $T_{ij}$ , that the ambulances need to reach the incidents. On the other hand the penalty factor, even in the cases where only one person is not collected from the incidents, is greater than the first component of the utility. Thus a local search algorithm like the variations of fictitious play we propose

initially searches for an allocation that collects all the injured people, so the penalty component of the utility will be zero, and afterwards for the allocation that minimises also the average time that the ambulances needed to reach the incidents. It is therefore easy to become stuck in local optima.

We have also examined how the results are influenced by the number of iterations we use in each of the 200 trials of the game. For that reason we have compared the results that we would have obtained if in each instance of the simulations we had stopped the negotiations between the emergency units after 50, 100, 150 and 200 iterations for both algorithms. Tables 5-7 and 8-10 depict the results for adaptive forgetting factor fictitious play and geometric fictitious play respectively.

		Iterations			
		50	100	150	200
3 incidents	10 ambulances	74.5	81.0	82.0	82.0
	15 ambulances	73.0	75.5	77.5	77.5
	20 ambulances	73.5	78.0	80.0	81.0
5 incidents	10 ambulances	80.0	87.5	90.0	91.0
	15 ambulances	76.5	84.5	89.5	90.5
	20 ambulances	62.0	77.0	80.0	83.0

**Table 5.** Percentage of solutions in which the capacity of the ambulance in every incident was enough to cover all injured people for different stopping times of the negotiations, 50, 100, 150 and 200 iterations of the adaptive forgetting factor fictitious play algorithm.

		Iterations			
		50	100	150	200
3 incidents	10 ambulances	91.44	92.35	92.95	92.84
	15 ambulances	88.65	90.28	89.97	89.88
	20 ambulances	89.09	89.33	90.44	90.59
5 incidents	10 ambulances	96.39	97.80	98.51	98.54
	15 ambulances	94.84	97.22	98.03	98.29
	20 ambulances	87.61	91.90	92.81	93.61

**Table 6.** Average percentage of injured people collected for different stopping times of the negotiations, 50, 100, 150 and 200 iterations of the adaptive forgetting factor fictitious play algorithm.

We can see from tables 5-7 that the performance of adaptive forgetting factor fictitious play, in all the measures that we used, is similar for 100, 150 and 200 negotiation steps. In particular when we consider the percentage of the instances

		Iterations			
		50	100	150	200
3 incidents	10 ambulances	1.2791	1.2603	1.2669	1.2702
	15 ambulances	1.2701	1.2452	1.2319	1.2624
	20 ambulances	1.2142	1.1971	1.1943	1.2058
5 incidents	10 ambulances	1.6989	1.6827	1.6772	1.6631
	15 ambulances	1.5569	1.5352	1.5304	1.5088
	20 ambulances	1.5298	1.4413	1.4306	1.4251

**Table 7.** Average percentage of the ratio  $f_{fp}/f_{opt}$  for different stopping times of the negotiations, 50, 100, 150 and 200 iterations of the adaptive forgetting factor fictitious play algorithm.

		Iterations			
		50	100	150	200
3 incidents	10 ambulances	94.0	94.0	94.7	95.5
	15 ambulances	71.0	73.0	73.3	74.5
	20 ambulances	60.5	61.3	62.0	63.0
5 incidents	10 ambulances	86.0	92.0	93.3	94.5
	15 ambulances	79.0	78.0	79.0	82.0
	20 ambulances	42.0	49.0	47.3	48.5

**Table 8.** Percentage of solutions in which the capacity of the ambulance in every incident was enough to cover all injured people for different stopping times of the negotiations, 50, 100, 150 and 200 iterations of the geometric fictitious play algorithm.

		Iterations			
		50	100	150	200
3 incidents	10 ambulances	99.62	99.65	99.69	99.74
	15 ambulances	88.20	88.21	88.21	88.24
	20 ambulances	86.65	87.41	88.23	89.90
5 incidents	10 ambulances	97.20	99.54	99.61	99.68
	15 ambulances	94.84	97.22	98.03	98.29
	20 ambulances	85.33	86.38	86.391	86.54

**Table 9.** Average percentage of injured people collected for different stopping times of the negotiations, 50, 100, 150 and 200 iterations of the geometric fictitious play algorithm.

		Iterations			
		50	100	150	200
3 incidents	10 ambulances	1.2957	1.2928	1.3039	1.2970
	15 ambulances	1.2965	1.3039	1.2801	1.2740
	20 ambulances	1.2779	1.2744	1.2723	1.2722
5 incidents	10 ambulances	1.8587	1.8540	1.8550	1.8519
	15 ambulances	1.7443	1.7205	1.7085	1.7074
	20 ambulances	1.8545	1.7845	1.7744	1.7727

**Table 10.** Average percentage of the ratio  $f_{fp}/f_{opt}$  for different stopping times of the negotiations, 50, 100, 150 and 200 iterations of the geometric fictitious play algorithm.

that the ambulances collected all the injured people and the ratio  $f_{fp}/f_{opt}$  the difference in the results after 100 and 200 negotiation steps is between 1% and 4.5%. The differences become even smaller for the percentage of the people that were saved which was less than 2%.

Geometric fictitious play was trapped in an area of a local minimum after few iterations since the results are similar after 50, 100, 150 and 200 iterations. This is reflected in the results where geometric fictitious play performed worse than adaptive forgetting factor fictitious play especially in the complicated cases where the negotiations where between 20 ambulances.

Adaptive forgetting factor fictitious play performed also better than the Random Neural Network (RNN) presented in [27], when we consider the percentage of the cases in which all the injured people are collected and the overall percentage of people that are rescued. The percentage of instances where the proposed allocations by the RNN could collect all the casualties were from 25 to 69 percent. The corresponding results of adaptive forgetting factor fictitious play are from 77.5 to 94.5. The overall percentage of people that are rescued by the RNN algorithm are similar to the ones of adaptive forgetting factor fictitious play, between 85 and 98.5 percent. The ratio  $\frac{f_{fp}}{f_{opt}}$  reported by [27] is better than that shown here. However in [27] only the examples in which all the casualties were collected were included to evaluate the ratio. Cases with high penalties, since the uncollected casualties introduce higher penalties than the inefficient allocation, were excluded from the ratio evaluation. Thus artificially improve their metric, especially when one considers that in many instances less than 40% of their solutions were included.

## 6 Conclusions

Fictitious play is a classic learning algorithm in games, but it is formed on an (incorrect) stationarity assumption. Therefore we have introduced a variation of fictitious play, adaptive forgetting factor fictitious play, which address this problem by giving higher weights to the recently observed actions using a heuristic rule from the streaming data literature.

We examined the impact of adaptive forgetting factor fictitious play parameters  $\lambda_0$  and  $\gamma$  on the results of the algorithm. We showed that these two parameter should be chosen carefully since there are combinations of  $\lambda_0$  and  $\gamma$  that induce very poor results. An example of such combination is when high values of the learning rate  $\gamma$  are combined with low values of  $\lambda_0$ . This is because values of  $\lambda_0 < 0.6$  assign small weights to the previously observed actions and this results in volatile estimations that are influenced by opponents' randomisation. High values of the learning rate  $\gamma$ , mean that  $\lambda_0$  is driven still lower, exacerbating the problem further. From the simulation results we have seen that a satisfactory combination of parameters  $\lambda_0$  and  $\gamma$  is  $0.8 \leq \lambda_0 \leq 0.9$  and  $\gamma = 10^{-4}$ .

Adaptive forgetting factor performed better than the competitor algorithms in the climbing hill game. Moreover it converged to the a better solution than geometric fictitious play in the vehicle target assignment game. In the disaster management scenario the performance of the proposed variation of fictitious play compared favorably with that of geometric fictitious play and a pre-planning algorithm that uses neural networks [27].

Our empirical observations indicate that adaptive forgetting factor fictitious play converges to a solution that is at least as good as that given by the competitor algorithms. Hence by slightly increasing the computational intensity of fictitious play less communication is required between agents to quickly coordinate on a desirable solution.

## References

1. Kho, J., Rogers, A., Jennings, N.R.: Decentralized control of adaptive sampling in wireless sensor networks. *ACM Trans. Sen. Netw.* **5**(3) (2009) 1–35
2. Lesser, V., Ortiz, C., Tambe, M., eds.: *Distributed Sensor Networks: A Multiagent Perspective*. Kluwer Academic Publishers, Boston (May 2003)
3. Kitano, H., Todokoro, S., Noda, I., Matsubara, H., Takahashi, T., Shinjou, A., Shimada, S.: Robocup rescue: Search and rescue in large-scale disasters as a domain for autonomous agents research. In: *Proc. of IEEE Conf. on System, Man and Cybernetics*. (1999)
4. van Leeuwen, P., Hesselink, H., Rohlinga, J.: Scheduling aircraft using constraint satisfaction. *Electronic Notes in Theoretical Computer Science* **76** (2002) 252 – 268
5. Stranjak, A., Dutta, P.S., Ebden, M., Rogers, A., Vytelingum, P.: A multi-agent simulation system for prediction and scheduling of aero engine overhaul. In: *AA-MAS '08: Proceedings of the 7th International joint Conference on Autonomous Agents and Multi-Agent Systems*. (2008) 81–88
6. Tumer, K., Wolpert, D.: *A survey of collectives*. In: *Collectives and the Design of Complex Systems*, Springer (2004) 1–42
7. Arslan, G., Marden, J., Shamma, J.: Autonomous vehicle-target assignment: A game theoretical formulation. *Journal of Dynamic Systems, Measurement, and Control* **129** (2007) 584–596
8. Chapman, A.C., Rogers, A.C., Jennings, N.R., Leslie, D.S.: A unifying framework for iterative approximate best response algorithms for distributed constraint optimisation problems. *The Knowledge Engineering Review* (forthcoming).

9. Fudenberg, D., Levine, D.: *The Theory of Learning in Games*. The MIT Press (1998)
10. Monderer, D., Shapley, L.: Potential games. *Games and Economic Behavior* **14** (1996) 124–143
11. Smyrnakis, M., Leslie, D.S.: Dynamic Opponent Modelling in Fictitious Play. *The Computer Journal* (2010) 2308–2324
12. Fudenberg, D., Tirole, J.: *Game Theory*. MIT Press (1991)
13. Nash, J.: Equilibrium points in n-person games. In: *Proceedings of the National Academy of Science, USA*. Volume 36. (1950) 48–49
14. Miyasawa, K.: On the convergence of learning process in a 2x2 non-zero-person game (1961)
15. Robinson, J.: An iterative method of solving a game. *Annals of Mathematics* **54** (1951) 296–301
16. Nachbar, J.H.: Evolutionary selection dynamics in games: Convergence and limit properties. *International Journal of Game Theory* **19**(1) (1990) 59–89
17. Shapley, L.: *Advances in Game Theory*. Princeton University Press, Princeton (1964)
18. Fudenberg, D., Kreps, D.M.: Learning mixed equilibria. *Games and Economic Behavior* **5** (1993) 320–367
19. Anagnostopoulos, C.: *A Statistical Framework for Streaming Data Analysis*. PhD thesis, Imperial College London (2010)
20. Salgado, M.E., Goodwin, G.C., Middleton, R.H.: Modified least squares algorithm incorporating exponential resetting and forgetting. *International Journal of Control* **47** (1988) 477–491
21. Black, M., Hickey, R.J.: Maintaining the performance of a learned classifier under concept drift. *Intelligent Data Analysis* **3**(6) (1999) 453 – 474
22. Muhlbaier, M., Polikar, R.: An ensemble approach for incremental learning in nonstationary environments. In: *Multiple Classifier Systems*. (2007) 490–500
23. Aggarwal, C., Han, J., Wang, J., Yu, P.: On demand classification of data streams. In: *Proceedings of the tenth ACM SIGKDD international conference on Knowledge Discovery and Data Mining*. (August 2004) 503–508
24. Haykin, S.: *Adaptive Filter Theory*. Prentice Hall (1996)
25. Bishop, C.M.: *Neural Networks for Pattern Recognition*. Oxford University Press (1995)
26. Claus, C., Boutilier, C.: The dynamics of reinforcement learning in cooperative multiagent systems. In: *AAAI '98/IAAI '98: Proceedings of the fifteenth national/tenth conference on Artificial intelligence/Innovative applications of artificial intelligence*. (1998) 746–752
27. Gelenbe, E., Timotheou, S.: Random neural networks with synchronized interactions. *Neural Computation* **20** (2008) 2308–2324
28. Wolsey, L.A.: *Integer Programming*. John Wiley & Sons (1998)
29. Nemhauser, G.L., Wolsey, L.A.: *Integer and Combinatorial Optimization*. John Wiley & Sons (1988)
30. Hillier, F.S., Lieberman, G.: *Introduction to Operations Research*. McGraw-Hill (2001)