Multi-model Adaptive Learning for Robots Under Uncertainty

Michalis Smyrnakis[1][a], Hongyang Qu[2][b], Dario Bauso[3][c] and Sandor Veres[2][d]
[1] *Science and Technology Facilities Council, Daresboury, U.K.*
[2] *Department of Automatic Control and Systems Engineering, University of Sheffield, Sheffield, U.K.*
[3] *Jan C. Willems Center for Systems and Control ENTEG, Faculty of Science and Engineering University of Groningen, Nijenborgh, Groningen, The Netherlands*

Keywords: Multi-model Adaptive Learning, Fictitious Play, Robotic Teams, Task Allocation.

Abstract: This paper casts coordination of a team of robots within the framework of game theoretic learning algorithms. A novel variant of fictitious play is proposed, by considering multi-model adaptive filters as a method to estimate other players' strategies. The proposed algorithm can be used as a coordination mechanism between players when they should take decisions under uncertainty. Each player chooses an action after taking into account the actions of the other players and also the uncertainty. In contrast, to other game-theoretic and heuristic algorithms for distributed optimisation, it is not necessary to find the optimal parameters of the algorithm for a specific problem a priori. Simulations are used to test the performance of the proposed methodology against other game-theoretic learning algorithms.

1 INTRODUCTION

Teams of robots can be used in many domains such as mine detection (Zhang et al., 2001), medication delivery in medical facilities (Evans and Krishnamurthy, 1998), formation control (Raffard et al., 2004; Smyrnakis et al., 2016) and exploration of unknown environments (Madhavan et al., 2004). A common feature shared by these applications is that robots should either minimise a cost function or maximise a utility function in a distributed fashion. Thus, the resulting problem can be formulated as an distributed optimization one. Distributed optimisation arises also in several applications such as in smart grids (Ayken and Imura, 2012), disaster management (Kitano et al., 1999; Smyrnakis and Galla, 2015), robot team coordination (Semsar-Kazerooni and Khorasani, 2009), sensor networks (Kho et al., 2009).

In each of the aforementioned applications, the agents need to coordinate to achieve a common goal. If the desired task requires distributed optimisation of a utility or cost function, then the resulting problem turns into a game where each agent optimizes a portion of the common objective function based on local information. In such a scenario, game theory provides formal tools to assess the quality of the solution obtained. When the same game is repeated over time, allowing players to learn their opponents' strategies solutions based on reinforcement learning (Fujita et al., 2016; Verstaevel et al., 2017) and adaptive multi agent systems (Frasheri et al., 2018) can be considered.

If the robots do not have access to the other robots' states, or if the communication channel is noisy or involves faulty sensors, we say that the game has imperfect information, and the robots have to make a decision under uncertainty. Uncertainty can lead to wrong decision. For example, wrong decisions could be made when the positions of other robots are inferred by noisy odometry or noisy camera input. Another example in which noisy observation can have impact on the coordination process of a robot team is the case of a fleet of Unmaned Aerial Vehicles (UAVs) that need to take images of an area in order to identify the growth of the crops. The images should be taken from various angles. However, it is not always possible for a UAV to know the exact position and bearing of the other UAVs, and therefore, to make correct decisions about when to change photo-shooting angles.

Another important factor which can introduce uncertainty is the state of the environment/ team of robots. Therefore, in cases where the information

[a] https://orcid.org/0000-0003-1416-3727
[b] https://orcid.org/0000-0002-1643-8926
[c] https://orcid.org/0000-0001-9713-677X
[d] https://orcid.org/0000-0003-0325-0710

Smyrnakis, M., Qu, H., Bauso, D. and Veres, S.
Multi-model Adaptive Learning for Robots Under Uncertainty.
DOI: 10.5220/0008927700500061
In *Proceedings of the 12th International Conference on Agents and Artificial Intelligence (ICAART 2020) - Volume 1*, pages 50-61
ISBN: 978-989-758-395-7; ISSN: 2184-433X

about either the environment's or the team's state is not shared with everyone, this can lead to non-optimal decisions. If this is the case, then the coordination problem can be formulated as a Bayesian game. Bayesian games can be used to deal with distributed optimisation problems under incomplete information. A sequence of Bayesian games can also be used to describe partially observable games and decentralised partial observable Markov decision process (dec-POMDPs)(Emery-Montemerlo et al., 2005).

In this paper, we propose a novel game-theoretic learning algorithm, which takes into account the aforementioned types of uncertainty. Therefore, it can be used as a coordination mechanism among robots playing either complete information games with noisy observations or Bayesian games. In detail, it is a synchronous algorithm, where Extended Kalman Filters Fictitious Play (EKFFP) (Smyrnakis and Veres, 2016) is combined with multi-model adaptive filters (MMAFs) (Brown and Hwang, 1997). The novelty of our algorithm is that the joint distribution of the uncertainty and the observed actions of other players' action are used to make decisions. Robots use multiple models to solve their optimisation task. Each model is either a probabilistic representation of the noisy observations model of the other players' states or of the state of the world. Each model of MMAF represents a part of uncertainty and the final decision making is based on a weighted average over all the models.

Another advantage of our algorithm, is that in contrast to EKFFP and various heuristic distributed optimisation algorithms, there is no need to tune any parameters. Therefore, there is no need to decide in advance the value of the EKFFP parameters. Instead, random valuations of these parameters can be used simultaneously. Each valuation predicts other robots' strategy from a different angle, which is represented by different models. Therefore, it is easier to adapt to evolution of other robots' strategy.

Distributed learning under noisy observation was considered in (Kostelnik et al., 2002). Particle swarm algorithms subjected to intrinsic noise was applied in (Di Mario et al., 2016). In (Hennig, 2013), noisy observations in a different context from this work were investigated, as no directly any knowledge about the noisy observation was used, such as their probability distribution. In (Chapman et al., 2012), the uncertainty was dealt within a game theoretic framework under a simplified assumption that players use the same strategy through the iterations of the game.

Various approaches have been adopted to solve Bayesian games including: Bayesian action graph games (Jiang and Leyton-Brown, 2010), Multi-agent influence diagrams (Koller and Milch, 2003) and Newton method (Govindan and Wilson, 2003). The difference between these approaches and the proposed algorithm is that their goal is to find the optimal strategy. However, the search for an optimal solution in cooperative Bayesian games is an NP-hard problem (Tsitsiklis and Athans, 1985), and thus, is not tractable. On the other hand, in our algorithm players take into account the other players' actions and their possible types when updating their desired action until they reach to a commonly accepted solution, which is usually an equilibrium point to the problem.

The rest of the paper is organised as follows. In Section 2, a brief description of the game-theoretic notions that will be used in the rest of the paper are presented. In Section 3, the learning algorithm EKFFP is presented. Section 4 describes our fictitious play based algorithm, which integrates multi-model adaptive filters and Fictitious play. Section 5 discusses some implementation details of our algorithm and game-theoretic learning algorithms in general. In Section 6, we evaluate our algorithm in several case studies. In Section 7, we summarise our findings and present our future work.

2 GAME-THEORETIC DEFINITIONS

This section contains a brief description of some game theoretic definitions that will be used in the rest of the paper.

2.1 Normal Form Games

A game Γ in normal form is defined as a tuple

$$\Gamma = \langle I, \{A^i\}_{i \in I}, \{r^i\}_{i \in I} \rangle,$$

where

- I is the set of indices of all players;
- A^i is the set of all possible actions of player i and the set product $A = \times_{i \in I} A^i$ is the set of all joint actions;
- $r^i : A \to \mathbb{R}$ is the utility (reward) function of player i, which computes the reward that the player gains after a joint action is selected.

Joint action a, can be written as $a = (a^i, a^{-i})$, where a^{-i} is the joint action of all players but i. A strategy of player i is a probability distribution over its action space, and let Δ^i denote the set of all the probability distributions over A^i. Each player uses a strategy $\sigma^i \in \Delta^i$ to choose its action. Similarly to actions, a joint strategy $\sigma \in \Delta$ is defined as an element of the set product $\Delta = \times_{i \in I} \Delta^i$, and σ^{-i} is a joint strategy of all players but i.

In this paper we consider iterative games. In these games, a game is repeatedly played along a discrete sequence of time instances called rounds or iterations when each player chooses their actions based on their strategies, the history of the observed joint actions in the played iterations of the game and the rewardse allocated to them.

A player uses a *pure* strategy when it deterministically chooses actions, therefore it puts all its mass function in a single action $a^i \in A^i$ such that $\sigma^i(a^i) = 1$. When this is not the case, we call such a strategy, *mixed* strategy. The expected reward of player i, given its opponents' strategies σ^{-i}, is denoted by $r^i(\sigma^i, \sigma^{-i})$. If σ^i is a pure strategy with $\sigma^i(a^i) = 1$, the expected reward can be written as $r^i(a^i, \sigma^{-i})$.

The most common deterministic decision rule in game theory is the so-called *best response* (*BR*) by which players choose the actions which maximise their expected rewards. Formally, the action that a player i will choose, given its opponents strategies σ^{-i}, is

$$BR^i(\sigma^{-i}) = \underset{a^i \in A^i}{\operatorname{argmax}} \quad r^i(a^i, \sigma^{-i}). \qquad (1)$$

A joint strategy $\tilde{\sigma} = (\tilde{\sigma}^i, \tilde{\sigma}^{-i})$ that satisfies

$$r^i(\tilde{\sigma}^i, \tilde{\sigma}^{-i}) \geq r^i(\sigma^i, \sigma^{-i}) \quad \forall i \in I, \forall \sigma^i \in \Delta^i$$

is a Nash equilibrium (Nash, 1950). Nash in (Nash, 1950) showed that every game has at least one equilibrium, i.e., there is at least one strategy $\tilde{\sigma}$ where players do not benefit from deviating from it unilaterally. A Nash equilibrium can be either mixed or pure if the strategy $\tilde{\sigma}$ is a mixed or a pure strategy respectively.

2.2 Bayesian Games

A Bayesian game, or game of incomplete information, is defined as a tuple

$$\mathcal{G} = \langle I, \{\Theta^i\}_{i \in I}, \{A^i\}_{i \in I}, \{p(\theta^i)\}_{\theta^i \in \Theta, i \in I}, \{r^i\}_{i \in I} \rangle,$$

where

- I is the set of player indices;
- Θ^i is the set of types belonging to player i and $\Theta = \times_{i \in I} \Theta^i$;
- A^i is set of possible actions of player i;
- $r^i : A \to \mathbb{R}$ is the utility function of player i.

Each type of a player represents a possible internal state of the player. At any time, a player can only be in one of its types. The type of a player constitutes its private information, in the sense that each player i knows the type that it is in. In contrast, the other players only know the probability that player i can be in a certain type at that moment. If $\theta^i \in \Theta^i$ is considered as the state of the environment, i.e., the type of a player is the state of the environment (world), then all players have the same types and thus $\Theta^i = \Theta^j$, $\forall i, j \in I$. The expected reward of a player in a Bayesian game is then estimated as:

$$r^i(\sigma^i, \theta^i) = \sum_{\theta^{-i} \in \Theta^{-i}} p(\theta^{-i}) r^i(\sigma^i, \theta^i, \sigma^{-i}, \theta^{-i}). \qquad (2)$$

A Bayesian Nash equilibrium is defined as

$$\sigma^i \in \underset{a^i \in A}{\operatorname{argmax}} \quad p(\theta^i | \theta^{-i}) r^i(\sigma^i, \theta^i, \sigma^{-i}, \theta^{-i})$$

Hence a Bayesian Nash equilibrium is a Nash equilibrium of the expanded game in which each player action space of pure strategies is the set of maps from Θ^i to A^i.

Table 1: Reward matrices of a two players Bayesian game.

Type A			Type B		
task	difficult	easy	task	difficult	easy
difficult	10,6	5,10	difficult	9,10	5,4
easy	8,5	6,8	easy	8,6	7,5

As an example, consider a task allocation scenario: two robots 1 and 2 need to collaborate to finish two tasks: *easy* and *difficult*. The *difficult* task can be performed efficiently only if both robots work together on it. Robot 1 can do both tasks at the same efficiency, and hence it has only one type. Robot 2 has two types A and B. In type A, robot 2 can do the *easy* task with greater efficiency than the difficult task, while in type B, it performs both tasks with the same efficiency. Furthermore, robot 2 always knows its type, while robot 1 only knows that with probability p robot 2 is in type A, and with probability $1 - p$ is in type B. In this game, the world has a unique state and thus does not affect robots' types. Table 1 illustrates an example of the utility function in this Bayesian game.

Each matrix of Table 1 represents the rewards that robots receive. The single type robot is the row player of the game, and the other robot is the column player. If robot 2 is in type A, then both robots receive the rewards in the left matrix, while when it is type B, they receive the reward in the right matrix. For example, the entry $5, 10$ of the left matrix means that when robot 1 performs the *difficult* task and robot 2 does the *easy* task and is in type A. In this case, robot 1 receives 5 unit of reward and robot 2 receives 10 units. We reinforce here that at any time of game playing, robot 2 knows exactly which reward matrix is used, while robot 1 makes decisions based on the probability distribution of types of robot 2: p and $1-p$ for the left and right matrix respectively.

3 LEARNING ALGORITHMS

A distributed optimisation task can be cast as a game. However, the formulation of the optimisation task as a game does not directly provide a solution to the game. A coordination mechanism between the robots is needed especially in cases where autonomy is a desirable property of the robot team. Game-theoretic learning algorithms can be used by robots to choose a joint action to solve the game. The canonical example of game-theoretic learning algorithms is fictitious play (FP). Fictitious play is an iterative learning algorithm. In each iteration t, each player estimates other players' strategies, and based on these estimates, chooses an action using the best response decision rule. At the initial iteration, i.e., $t = 0$, every player i maintains some arbitrary, non-negative weights $\kappa_t^{i\to j}$ for each other player j as the estimation of their strategy. In particular, $\kappa_t^{i\to j}(a^j)$ is the weight for action $a^j \in A^j$ of player j. At successive iterations, players update their weight functions based on other players' chosen actions. The update of player i's weight function for player j is computed as follows (Fudenberg and Levine, 1998):

$$\kappa_t^{i\to j}(a^j) = \kappa_{t-1}^{i\to j}(a^j) + \begin{cases} 1 & \text{if} \quad a^j = a_{t-1}^j \\ 0 & \text{otherwise} \end{cases} \tag{3}$$

where a_{t-1}^j is the action that player j chooses at iteration $t-1$. Based on these weights, player i then estimates player j's strategy using the following equation:

$$\sigma_t^j(a^j) = \frac{\kappa_t^{i\to j}(a^j)}{\sum_{a^j\in A^j}\kappa_t^{i\to j}(a^j)}. \tag{4}$$

Fictitious play converges to the Nash equilibrium in many classes of games, such as 2×2 games with generic payoffs (Miyasawa, 1961), zero sum games (Robinson, 1951), games that can be solved using iterative dominance (Nachbar, 1990), and potential games (Monderer and Shapley, 1996). However, this convergence can be very slow (Fudenberg and Levine, 1998) because of the implicit assumption that all players use the same strategy throughout the game. In (Smyrnakis and Leslie, 2010) and (Smyrnakis and Veres, 2016), variants of fictitious play were proposed, which were based on particle filters and extended Kalman filters respectively. These algorithms assume that players adapt their strategies through the iterations of the game. In both variants, the fictitious play process is described as a hidden Markov model (HMM). Each player maintains some unconstrained propensities[1] , which are responsible for their strategies. In each iteration of game playing, each player aims to predict other players' propensities, i.e., hidden layer of the HMM, by using the history of other players' actions, i.e., observations layer of the HMM.

More formally, let $x^i(a^i)$ denote the propensity of player i to play action a^i, and a_t^i the action of player i at the t-th iteration of the fictitious play process. The probability at which each player estimates about the propensity of other players is:

$$p(x^j(a^j)|a_0^j, a_1^j, \ldots, a_t^j) \quad \forall j \in I\setminus\{i\}, \forall a^j \in A^j. \tag{5}$$

This probability was evaluated using particle filters in (Smyrnakis and Leslie, 2010) and extended Kalman filters in (Smyrnakis and Veres, 2016). In this paper, we only consider the variant of fictitious play based on extended Kalman filters. But the same methodology can be easily applied to the variant with particle filters. As each player estimates the propensity of every other individual player separately, only inference over a single "opponent" player, say player j, will be presented in the rest of the paper.

3.1 Extended Kalman Filter Fictitious Play (EKFFP)

This variant of fictitious play is based on the assumption that players have no prior knowledge about other players' strategies, and thus, an autoregressive model can be used to propagate the propensities (Smyrnakis and Leslie, 2010). In addition, inspired from the sigmoid functions that are used in neural networks to connect the weights and the observations, a Boltzman formula is used to relate the propensities with other players' strategies (Bishop, 1995). The following state space model is used to describe EKFFP:

$$\begin{aligned} x_t^j(a^j) &= x_{t-1}^j(a^j) + \xi_{t-1}^j \\ I_{a_t^j=a^j}(a^j) &= h(x_t^j(a^j)) + \zeta_t^j \end{aligned} \tag{6}$$

where $I_{a_t^j=a^j}(a^j)$ is the measurement equation, which relates the propensities to the actions of the players. The noise of the propensity process, $\xi_{t-1}^j \sim N(0,\Xi)$, which comprises the internal states, has zero *mean* and *covariance matrix* Ξ. The error, $\zeta_t \sim N(0,Z)$, of

[1]The strategies are probability distributions. Thus, when a dynamical model is used to propagate them, new estimates are not necessary to lay in the probability distributions space. For that reason, the intentions of players to choose an action, namely propensities, which are not bounded to probability distribution spaces, are used (Smyrnakis and Leslie, 2010).

the observations has zero *mean* and *covariance matrix Z*. This error occurs because a discrete 0-1 process, such as the best response in Equation (1) is represented through the continuous Boltzmann formula $h(\cdot)$ in which τ is a "temperature parameter". The components of the vector h, are evaluated as:

$$h(x^j(a^j)) = \frac{exp(x^j(a^j)/\tau)}{\sum_{a^k \in A^j} exp(x^j(a^k)/\tau)}. \qquad (7)$$

The behaviour of the EKFFP algorithm at the t-th iteration of a game can be described as follows. At first, player i uses the EKF process, which is based on the state space in Equation (6), to predict other players' propensities. Player i then using these estimates, evaluates player j's strategy of choosing an action $a^j \in A^j$, $\sigma_t^j(a^j)$, as follows:

$$\sigma_t^j(a^j) = \frac{exp(\bar{x}_t^j(a^k)/\tau)}{\sum_{a^k \in A^j} exp(\bar{x}_t^j(a^k)/\tau)}, \qquad (8)$$

where $\bar{x}_t^j(a^k)$ is player i's prediction of the propensities of player j in order to choose action $a^k \in A^j$ based on the state equations in Equation (6) and using observations up to time $t-1$. Player i then uses the estimates in Equation (8) to choose an action using best response in Equation (1). After all players have chosen an action, they use the EKF update process to correct their estimates about other players' strategies in the light of the recently observed actions. Then, the next iteration of EKFFP starts with $t = t+1$. The EKF estimations can be computed by any standard textbook procedure, such as in (Sakka, 2013). Algorithm 1 summarises the fictitious play algorithm when EKF is to predict other robots' strategies.

Algorithm 1: Extended Kalman filter fictitious play (Smyrnakis and Veres, 2016).

```
1: while $t < max\ iterations$ do
2:     for all $j \in I \setminus \{i\}$ do
3:         Predict other players' propensities for the
           next iteration $t+1$ using the state equations
           in Equation (6).
4:     Use the beliefs about other players' strategies
       in Equation (8) and choose an action using BR
       in Equation (1).
5:     Observe other players' actions
6:     for all $j \in I \setminus \{i\}$ do
7:         Update estimates of player $j$'s propensities
           using extended Kalman Filtering to obtain
           $\bar{x}_t^j(a^k)$.
8:     $t = t+1$
```

4 MULTI-MODEL ADAPTIVE FILTER EKFFP

4.1 Multi-model Adaptive Filters

The EKFFP process requires the definition of the covariance matrices Ξ and Z for the random variables ξ and ζ respectively. The performance of the learning algorithm is affected by the values of these covariance matrices. In (Smyrnakis and Veres, 2016) specific values were proposed for those covariance matrices, although these values are not optimal for all games. In this work, we propose a new approach that uses many models, each of which represents a pair of covariance matrices Ξ and Z. This approach then uses a weighted sum of these models in order to obtain an estimate of other players' propensities, instead of estimating the propensities from a single pair of covariance matrices. For Bayesian games, players can have a propensity estimate for each state of the nature or each type of other players.

The framework that allows many models to be considered under the EKFFP process is multiple model adaptive filters (Brown and Hwang, 1997; Blair and Bar-Shalom, 1996; Caputi, 1995). Let L be the set of all models that are used. Instead of estimating the propensity in Equation (5), each player should estimate

$$p(x^j(a^j), l | a_0^j, a_1^j, \ldots, a_t^j), \qquad (9)$$

where $l \in L$ is one of the possible models, each of which either refers to a pair of covariance matrices for potential games, or the state of the nature or another player's type Θ^i in Bayesian games.

To simplify notations, we use $\tilde{a}_t^j$ to denote $(a_0^j, a_1^j, \ldots, a_t^j)$. The estimate of other players' propensities in Equation (9) can be written as:

$$p(x^j(a^j), l|\tilde{a}_t^j) = p(l|\tilde{a}_t^j) p(x^j(a^j)|l, \tilde{a}_t^j), \qquad (10)$$

where $p(x^j(a^j)|l, \tilde{a}_t^j)$ for a given l is the standard EKF estimate of other player's propensity and $p(l|\tilde{a}_t^j)$ can be seen as the weight factor of each model.

Using Bayes rule, the probability $p(l|\tilde{a}_t^j)$ can be written as:

$$p(l|\tilde{a}_t^j) = \frac{p(\tilde{a}_t^j|l)p(l)}{\sum_{l \in L} p(\tilde{a}_t^j|l)p(l)}, \qquad (11)$$

where $p(l)$ is the prior distribution of the model l. The probability $p(\tilde{a}_t^j|l)$ can be written as

$$\begin{aligned} rlp(\tilde{a}_t^j|l) = & \quad p(a_t^j, a_{t-1}^j, \ldots, a_0^j|l) \qquad (12)\\ = & \quad p(a_t^j, a_{t-1}^j, \ldots, a_1^j|a_0^j, l)p(a_0^j|l) \\ & \quad \vdots \\ = & \quad p(a_t^j|\tilde{a}_{t-1}^j, l)p(a_{t-1}^j|\tilde{a}_{t-2}^j, l)\cdots p(a_0^j|l). \end{aligned}$$

The propensities are described by the hidden Markov model, so they are conditionally independent, and thus, Equation (12) can be written as:

$$p(\tilde{a}_t^j|l) = \prod_{q=0}^{q=t} p(a_q^j|l). \qquad (13)$$

4.2 Multi-model Adaptive Filters EKFFP (MMAF-EKFFP)

Let $|L|$ denote the cardinality of set L, and $x_{t,l}^j(a^j)$ the propensity of player j, playing action a^j at the t^{th} iteration under model l. In the multi-model adaptive filters EKFFP process, each player uses $|L|$ models for the propensity of each other player. In particular, each model is a state model:

$$\begin{aligned} x_{t,l}^j(a^j) &= x_{t-1,l}^j(a^j) + \xi_{t-1,l}^j \\ I_{a_t^j=a^j}^l(a^j) &= h(x_{t,l}^j(a^j)) + \zeta_{t,l}^j. \end{aligned} \qquad (14)$$

For each of these models, player i uses the EKF process to predict player j's propensity, i.e., $\bar{x}_{t,l}^j(a^k)$, in order to choose an action $a^k \in A^j$ under model l. This prediction is weighted using Equation (11). The estimate of player j's propensity to choose action $a^k \in A^j$ is then the sum of the weighted estimates of each model:

$$\bar{x}_{t,a^k}^j = \sum_{l \in L} p(l|a^k)\bar{x}_{t,l}^j(a^k), \qquad (15)$$

where $p(l|a^k)$ is evaluated using Equation (11) and (13). Then Equation (8) can be applied to evaluate player j's strategy. Each model of the estimates of player j's propensity are updated using the standard EKF process under the light of the new action player j has chosen. Algorithm 2 and Figure 1 summarise the MMAF-EKFFP algorithm.

5 IMPLEMENTATION DETAILS

In this section, we discuss some implementation details in robotics of the MMAF-EKFFP algorithm and of game-theoretic learning algorithms in general.

Algorithm 2: MMAF-EKFFP.

```
1:  while $t < max\ iterations$ do
2:    for all $j \in I \setminus \{i\}$ do
3:      for all $l \in L$ do
4:        For each model $l$ predict other players' propensities for the next iteration $t+1$, using Equation (14).
5:      Evaluate $\bar{x}_t^j(a^k)$ using Equation (15)
6:    Compute the beliefs about other players' strategies using Equation (8), and choose an action using BR in Equation (1)
7:    Observe other players' actions
8:    for all $j \in I \setminus \{i\}$ do
9:      for all $l \in L$ do
10:       Update each model's estimate of other players' propensities using extended Kalman filter to obtain $\bar{x}_t^j(a^k)$
11:   $t = t + 1$
```

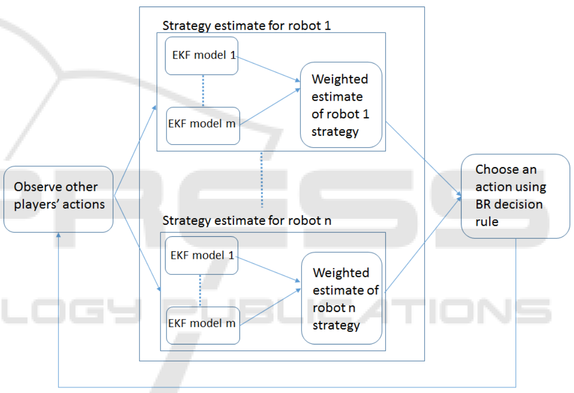
Strategy estimate for robot 1
EKF model 1
Weighted estimate of robot 1 strategy
EKF model m
Observe other players' actions
Choose an action using BR decision rule
Strategy estimate for robot n
EKF model 1
Weighted estimate of robot n strategy
EKF model m

Figure 1: The MMAF EKFP process.

5.1 Robots' Decisions

As any iterative learning algorithm, our algorithm assumes that a specific game can be iteratively be played and a final decision is reached either when the algorithm converges to an equilibrium or when the maximum number of iterations is reached. In robotics, this can be seen as the following coordination mechanism. The robots in each iteration choose an action which they intent to play and makes aware the other robots for its intentions, i.e., by communicating this intention to the other robots. Based on this information, they update their estimates about other players strategies and update the action they are intended to choose. The action that the team of robots will execute will be the joint action of the final iteration of the learning algorithm.

Table 2: Reward matrices of the Bayesian game for the remaining action.

Type A

	do	not do
do	10,10	-5,0
not do	-5,0	0,0

Type B

	do	not do
do	10,5	-5,0
not do	0,-10	0,0

5.2 Complexity

The computational complexity of the EKF algorithm is upper bounded by the complexity of inverting a matrix $O(n^3)$ (Bonato et al., 2009), where n is the rank of the inverted matrix. Therefore the additional computational complexity of EKFFP when it is compared with classic FP is upper bounded by $O((|I|-1)|A^k|^3)$, where A^k ($k \in I$) is the largest set of actions among all players. Similarly, for MMAF-EKFFP it is $O(|\mathcal{M}|(|I|-1)|A^k|^3)$, where $|\mathcal{M}|$ denotes the number of models which are used. The difference of the two algorithms is of a multiplicative magnitude of $\mathcal{M}$. This computational difference can be vanished if the computations of each model $m \in \mathcal{M}$ are executed in parallel.

5.3 Example of a Sequence of Bayesian Games

In (Emery-Montemerlo, 2005), it was shown that dec-POMDPs can be cast as a sequence of Bayesian games. In this section, we present a process of implementing a sequence of Bayesian games to solve a coordination task. Consider the task allocation problem between two robots with rewards shown in Table 1 in Section 2. Depending to the type of robot 2, there are two pure Nash equilibria where robots can converge. The first one is $(easy, easy)$ when robot 2 is of type A and $(difficult, difficult)$ when robot 2 is of type B. In order to accomplish their mission robots should finish both the easy and the difficult task. Independently of the action the robots will choose for this game, they will have accomplish only half of the necessary tasks. Therefore the players will have to play a sequence of games in order to finish both tasks, easy and difficult. The first game is the one with rewards depicted in Table 1. Another game should be defined in order to complete the unselected task, after making a decision based on the first game. An example of such a game is defined in Table 2. There the robots should choose if they will both try to finish the remaining task or only one one should try or none of them should try. Note here that the game depicted in Table 2 makes the two robots coordinate and choose to do the remaining task together. Nonetheless, depending on the nature of the problem another reward matrix can be used in order to allow robots having different behaviours.

6 SIMULATION RESULTS

6.1 Results in Potential Games

In this section the performance of the proposed algorithm, MMAF-EKFFP, is compared against the one of EKFFP in a resource allocation task. This is the vehicle-target assignment game which was introduced in (Arslan et al., 2007). In this potential game, N robots and M targets are placed in an area. The goal of each robot is to engage a target in order to destroy it. The actions of each robot are simply the choice of a target to engage. Each robot can choose only one target to engage, but a target can be engaged by many robots. The probability robot i has to destroy a target m is assumed to be independent of the probability another robot j has to destroy the same target. The probability that a target m will be destroyed is computed as:

$$1 - \prod_{i:a^i=m} (1-p_{im}),$$

where p_{im} is the probability at which the robot i can destroy target m.

The reward that robots will share is the sum of the rewards each target m will produce if a specific joint action a is selected:

$$r_{global}(a) = \sum_{m \in M} r_m(a), \qquad (16)$$

where $r_m(a)$, is defined as the product of target's m value V_m and the probability it can be destroyed by the robots which engage it. More formally, we can express the utility that is produced by target m as follows.

$$r_m(a) = V_m(1 - \prod_{i \in I:a^i=m} (1-p_{im})). \qquad (17)$$

Multiple cases of the vehicle target assignment game were considered, with varying number of robots. In particular 20 targets and $N = \{20, 30, 40, 50, 60\}$ robots were considered. The simulations were run in a computer with dual Intel Xeon E5-2643 v2 processors (3.50GHz, 6 cores) and 384 GB memory. The probability that each robot i can destroy a target m was set to be proportional to their euclidean distance. For each case, we run MMAF-EKFFP with six and twelve models respectively. Each model represents a pair of covariance matrices Ξ and Z. In all cases, the values of Ξ and Z were uniformly sampled from the interval $(0, 0.1]$. Comparisons are also made with the classic EKFFP with Ξ and Z defined as in (Smyrnakis and Veres, 2016). We reinforce here that there does not exist a universal combination of Ξ and Z which maximises the performance of EKFFP for all games

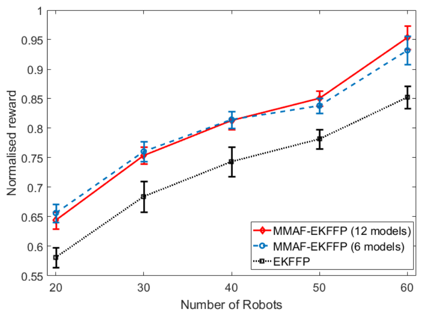

Figure 2: Average reward over 200 runs of the vehicle target assignment game.

(Smyrnakis and Veres, 2016). The parameters which are reported in (Smyrnakis and Veres, 2016) are suggestive. Therefore, the question which arises is which pair of parameters Ξ and Z maximise the performance of EKFFP for the games of interest. MMAF-EKFFP provides a solution to this problem, since many combinations of Ξ and Z can be used as part of different models. Then the players will select actions based on the weighted sum of these models. The results presented in this paper are averaged over 200 runs for each case. In each run, target values V_m were uniformly chosen from $(0,10]$. The target and the robot positions were uniformly chosen from $[0,1]$. This results in different scales of reward function. In order to make comparison feasible, the following normalised version of the global utility was used:

$$r_{total} = \frac{r_{global}}{\sum_m V_m}.$$

As it is depicted in Fig. 2, MMAF-EKFFP with multiple models performs better than the classic EKFFP, which uses only a single model. In addition, MMAF-EKFFP with 6 and 12 models perform similarly when cases with up to 40 robots are considered. When more players are considered, having more models improves the results of the algorithm.

It is expected that MMAF-EKFFP has heavier computational cost than EKFFP as the number of models is increased. Nonetheless as it is shown in Table 3, the running time of MMAF-EKFFP is in the same level as EKFFP because it is implemented using parallel computing, taking advantage of the multiple cores which many development platforms already have in robotics. In this experiment, each model is processed by an individual thread, which then runs in an individual physical computation core.

6.2 Games with Noisy Rewards

This section contains the simulation results in a scenario where the robots receive noisy observations of the other robots' actions. Noisy observations denote observations which for some reason is incorrect. Consider the case where two UAVs monitor a part of a field in order to decide if fertilisation is needed. The best results are obtained when one UAV flies at the top of the area and the other flies at the sides of the area. This scenario can be modelled as a two player game, which is depicted in Table 4. If both UAVs choose to go at the top of the area of interest, they will collide and they will receive some negative rewards. On the other hand, if they both choose to fly at the side of the area, then they will gain no reward because the quality of the images they gather will be poor. Finally, some positive reward is generated only if the two UAvs make different decisions. Note that the natural choice of the two UAVs is to choose to fly at the side of the area. They change their decision only when they believe with probability greater than 0.8 that the other UAV will choose to go at the side of the area of interest. In addition, each UAV can observe correctly the intention of the other UAV with probability p in every iteration of the coordination process.

Remark. In this paper, we assume that each robot knows the probability distribution of noisy observations, either by having some prior knowledge about its sensors' specification or using some methods to estimate that distribution as the one proposed in (Rosen and Leonard,) to estimate this distribution.

Figure 3 shows the results of MMAF-EKFFP and EKFFP for the game with rewards depicted in Table 4. As it is shown here the percentage of times that MMAF-EKFFP converged to a Nash equilibrium is always greater than 50%. This is significantly better than the results reported in (Chapman et al., 2012), where the results of Generalised Weakened fictitious play (GWFP) (Leslie and Collins, 2006) and Filtered Fictitious Play (FFP) (Chapman et al., 2012). The probability of converging to a Nash equilibrium reached zero when the 50% or more of the observations were faulty for FFP. For the case of GWFP the probability of converging to a Nash equilibrium reached zero when the 30% or more of the observations were faulty. Note here that even EKFFP performs better than FFP and GWFP since the probability of converging to a Nash equilibrium reached zero when the 80% or more of the observations were faulty.

The minimum probability of convergence to a Nash equilibrium for the MMAF-EKFFP algorithm is observed when the 50% of the observations were faulty. This is because the observations had exactly the same chance to be correct or faulty. When play-

Table 3: Time in seconds that 20 iterations needed for various number of models EKF fictitious play.

	20 robots	30 robots	40 robots	50 robots
EKFFP	0.0649 ± 0.0042	0.1133 ± 0.0130	0.1548 ± 0.0109	0.2321 ± 0.0227
MMAF-EKFFP (6 models sequential)	0.3243 ± 0.0263	0.5909 ± 0.2247	1.1428 ± 0.1618	1.6509 ± 0.3116
MMAF-EKFFP (12 models sequential)	0.7773 ± 0.0327	1.3294 ± 0.1102	2.2283 ± 0.2177	3.0378 ± 0.2063
MMAF-EKFFP (6 models parallel)	0.0568 ± 0.0013	0.0906 ± 0.0034	0.1326 ± 0.0059	0.1918 ± 0.0105
MMAF-EKFFP (12 models parallel)	0.0616 ± 0.0030	0.1091 ± 0.0065	0.1619 ± 0.0071	0.2326 ± 0.0192

Table 4: UAVs' rewards for the game with noisy observations.

	Top	Sideways
Top	-4,-4	0,1
Sideways	1,0	0,0

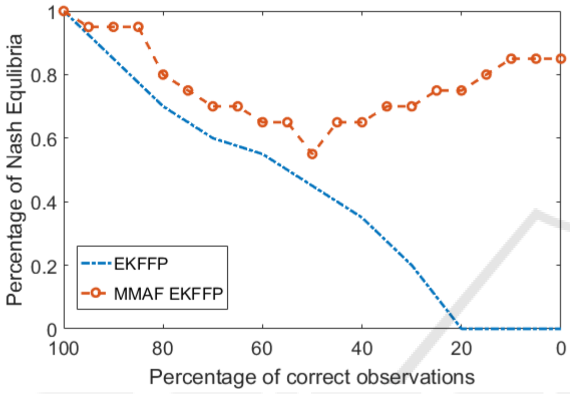

Figure 3: Probability to converge to Nash equilibrium as a function of the percentage of the correctly observed opponents' actions.

ers know that the probability of a faulty observation is greater than a correct one, they use this information and increase their chances to converge to an equilibrium point.

6.3 Results in Bayesian Games

The majority of the methods that are used to solve Bayesian games, such as Agent Security via Approximate Policies (ASAP) (Paruchuri et al., 2007a), Mixed Integer Programming Nash (MIP Nash) (Sandholm et al., 2005), brute force search methods (Oliehoek et al., 2010) or Multiple Linear Programs (Conitzer and Sandholm, 2006) tries to find the Bayes Nash equilibrium with the highest reward. In order to solve the Bayesian game in Table 1, we can transform it into a strategic form game using Harsanyi's transformation (Harsanyi and Selten, 1972), and search for the Nash equilibrium of this new game. An example of Harsanyi's transformation is depicted in Table 5, where the Bayesian game of Table 1 has been cast as a strategic form game, with probabilities p and $1-p$ of the second player being of type A or B. In this game, robot 1 is the row player and robot 2 the column player.

Note here that as the second robot can be of any of the two types in the strategic form game, its actions consist of all the possible combinations of its actions in the two games. Therefore, in the strategic form representation of Table 1, the second robot has four possible actions $E_AD_B, E_AE_B, D_AD_B, D_AE_B$, where E_AD_B denotes selecting the easy task if it is of type A and the difficult task if it is of type B etc.

The game with rewards in Table 5 have three Bayes-Nash equilibria. Two pure Bayes-Nash equilibria and one mixed. The joint actions $(difficult, D_AD_B)$ and $(easy, D_AE_B)$ are pure Bayes-Nash equilibria. The mixed Bayes-Nash equilibrium exists when $p > \frac{1}{5}$: robot 1 chooses the difficult task with probability $\frac{5p-1}{5-p}$ and robot 2 chooses D_AD_B with probability $\frac{2}{3}$. Nonetheless, even finding the Bayes-Nash equilibria does not answer to the question which action the robots should choose if they are playing any of the two games. For example if the type of robot 2 is A, then the equilibrium of the actual game which will be played is easy;easy, as it can be seen from Table 1. On the other hand, if the type of the robot 2 is of type B then the equilibrium of the actual game which will be played is difficult,difficult.

On the other hand, the proposed algorithm allows robots to learn what the other robots are doing, and evaluate the probability of choosing a particular sequence of actions given that they are of a specific type. Then based on this knowledge, they choose the action that maximises their expected reward conditional to the possible types of the other players.

Now we study the performance of MMAF-EKFFP in two games with incomplete information. The first game had at least one pure strategy Nash equilibrium, and the second has only a mixed strategy Nash equilibrium.

The first game is the one depicted in Tables 1 and 2. MMAF-EKFFP always converged to the pure Nash equilibrium of the game, and therefore, the two robots always chose to work on the same task (easy or difficult) jointly.

The second example which is considered comes from a security problem (Paruchuri et al., 2007b). In security games, a group of security robots try to secure some areas of interest and an attacker robot tries to invade these areas. In (Beard and McLain, 2003; Ruan et al., 2005), the security robots cannot be phys-

Table 5: Strategic form game's rewards, of the Bayesian game of Table 1 as a function of p.

		Robot 2			
		E_AD_B	E_AE_B	D_AD_B	D_AE_B
Robot 1	difficult	$9+p, 10-4p$	$5+5p, 4+2p$	$9-4p$, 10	$5, 4+6p$
	easy	$8, 6-p$	$7+p, 5$	$8+2p, 6-2p$	$7-p, 5+3p$

ically present in all the areas of interest at the same time. Instead, they can choose among various patrol routes. Security robots choose the route and the areas they will patrol based on the importance of the areas or the likelihood at which an attacker robot will be appear etc. The security problem can be cast as a two player game (Paruchuri et al., 2007b). If there are are $\mathcal{M}$ areas of interest, then the action of the security robots will be the d-tuple ($d \leq m$) of the areas which the robots will patrol. The order by which the robots visit the areas is also taking into account. For instance, in the case of three areas $\{1,2,3\}$, each petrol order, e.g., $1 \leftarrow 2 \leftarrow 3$ or $1 \leftarrow 3 \leftarrow 2$, is a different action. The attacker robot can choose any of the $\mathcal{M}$ available areas to invade. Assume that the security robots can choose from $\mathcal{K}$ patrolling routes. The reward of the security robots $r_i(k)$ ($k \in \mathcal{K}$) and that of attacking robot $r_j(m)$ ($m \in \mathcal{M}$) are estimated respectively as follows.

$$r_i(k) = \begin{cases} -u_i(a^i) \text{ if } a^j \notin a^i \\ p_{a^i}c_i + (1-p_{a^i})(-u_i) \text{ if } a^j \in a^i, \end{cases} \tag{18}$$

where $u_i(a^i)$ is the value of area a^i to the security robots, p_{a^i} is the probability that the security robots can catch the attacker in the a^i area, c_i is the reward to the security robots if the attacker is caught:

$$r_j = \begin{cases} u_j \text{ if } a^j \notin a^i \\ -p_{a^j}c_j + (1-p_{a^j})(u_j) \text{ if } a^j \in a^i, \end{cases} \tag{19}$$

where $u_i(a^j)$ is the value of area a^j to the attacker robot, p_{aj} is the probability that the security robots can catch the attacker when patrolling area a^j, and c_j is the cost to the attacker robot if it is caught.

Consider the case where two areas are available and the actions available to security robot are $1 \leftarrow 2$ and $2 \leftarrow 1$ respectively. In addition, assume that the attacking robot can be of two types A and B. The above reward function, when similar parameters to (Paruchuri et al., 2007b) are used, can be modelled as a Bayesian game as it is depicted in Table 6.

When the attacker is type A, the optimal policy for the security robots, which leads to a Nash equilibrium, is to choose a mixed strategy and play action $1 \leftarrow 2$ with probability 0.58 and $2 \leftarrow 1$ with probability 0.42. The mixed strategy for the attacking robot is to choose area 1 with probability 0.375 and area 2 with probability 0.625. If attacker is of type B, the security robots' optimal policy is to choose a mixed strategy by playing action $1 \leftarrow 2$ with probability 0.35 and $2 \leftarrow 1$ with probability 0.65. The mixed strategy for the attacking robot is to choose area 1 with probability 0.39 and area 2 with probability 0.61. The security robots assume that the attacking robot is of type A with probability 0.9 and of type B with probability 0.1. Figure 4 illustrates the probability with which the attacking robot, chose Area 1 when it was of type A. The performance of MMAF-EKFFP was compared with EKFFP and the asynchronous best response algorithm which proposed in (Emery-Montemerlo et al., 2005) in order to solve Bayesian games. As it can be seen from Figure 4, MMAF-EKFP converged to the Nash equilibrium of the game while the two other algorithms failed to converged to a value close to the Nash equilibrium. Similarly, Figure 5 depicts the probability with which the security robot chose action $2 \leftarrow 1$. As it can be observed from Figure 4, MMAF-EKFP converged to the Nash equilibrium, while the other algorithms failed to converged to a value close to the Nash equilibrium. Similar results were obtained for various combinations of the probabilities with which the attacking robot could be of type A or B. In particular, the differences in the results between the case where the attacking robot is of type A with probability 0.9 and the case when it is of type B with probability 0.1, are less than 0.01 from the reported results in Figures 4 and 5.

Table 6: Reward matrices of the attacking and security robot for two different types of attacking robot. The top table is the game played when the attacking robot is of type A.

	Areas 1,2	Areas 2,1
Area 1	-1,0.5	-0.125,-0.125
Area 2	-0.375,0.125	-1,0.5

	Areas 1,2	Areas 2,1
Area 1	-1.2,0.4	-0.025,-0.025
Area 2	-0.275,0.125	- 1.2,0.4

7 CONCLUSIONS AND FUTURE WORKS

A new game-theoretic learning algorithm based on EKFFP and multi-model adaptive filters has been proposed. This new algorithm can take into account var-

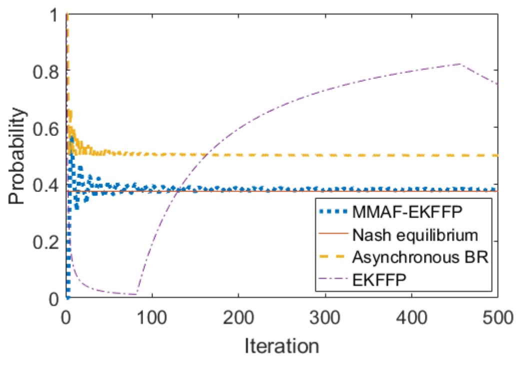

Figure 4: Probability of the attacking robot to choose area 1 when it is of type A as a function of the number of iterations.

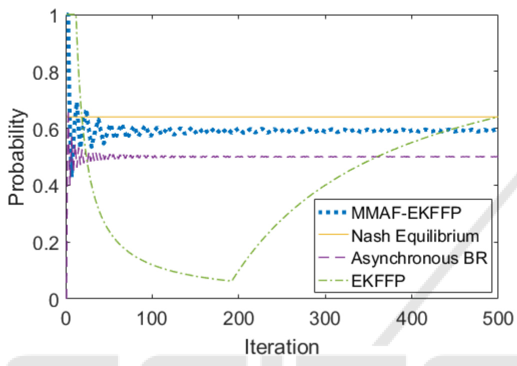

Figure 5: Probability of the security robot to choose action $2 \leftarrow 1$ when the attacking robot is of type B as a function of the number of iterations.

ious forms of uncertainty. In particular, uncertainty due to noisy observations and different types of other robots were considered. The performance of our algorithm is demonstrated via two case studies in robotics and two Bayesian games. The experimental results showed that it can provide better solution than the classic EKFFP algorithm and can be run as fast as the latter if implemented using parallel computing. In the future, we will apply this algorithm to more case studies to investigate the relationship between the average rewards and the number of models, as in Fig. 2, the average rewards gained by MMAF-EKFFP with 12 models is not significantly different from that from 6 models. We also intend to implement it in GPUs to further improve its performance.

REFERENCES

Arslan, G., Marden, J. R., and Shamma, J. S. (2007). Autonomous vehicle-target assignment: A game-theoretical formulation. *Journal of Dynamic Systems, Measurement, and Control*, 129(5):584–596.

Ayken, T. and Imura, J.-i. (2012). Asynchronous distributed optimization of smart grid. In *SICE Annual Conference (SICE), 2012 Proceedings of*, pages 2098–2102. IEEE.

Beard, R. W. and McLain, T. W. (2003). Multiple uav cooperative search under collision avoidance and limited range communication constraints. In *Proceedings of 42nd IEEE Conference on Decision and Control.*, volume 1, pages 25–30. IEEE.

Bishop, C. M. (1995). *Neural Networks for Pattern Recognition*. Oxford University Press.

Blair, W. and Bar-Shalom, T. (1996). Tracking maneuvering targets with multiple sensors: Does more data always mean better estimates? *Aerospace and Electronic Systems, IEEE Transactions on*, 32(1):450–456.

Bonato, V., Marques, E., and Constantinides, G. A. (2009). A floating-point extended kalman filter implementation for autonomous mobile robots. *Journal of Signal Processing Systems*, 56(1):41–50.

Brown, R. G. and Hwang, P. Y. (1997). Introduction to random signals and applied kalman filtering: with matlab exercises and solutions. *Introduction to random signals and applied Kalman filtering: with MATLAB exercises and solutions, by Brown, Robert Grover.; Hwang, Patrick YC New York: Wiley, c1997.*, 1.

Caputi, M. J. (1995). A necessary condition for effective performance of the multiple model adaptive estimator. *IEEE transactions on aerospace and electronic systems*, 31(3):1132–1139.

Chapman, A. C., Williamson, S. A., and Jennings, N. R. (2012). Filtered fictitious play for perturbed observation potential games and decentralised pomdps. *CoRR*, abs/1202.3705.

Conitzer, V. and Sandholm, T. (2006). Choosing the best strategy to commit to. In *ACM Conference on Electronic Commerce*.

Di Mario, E., Navarro, I., and Martinoli, A. (2016). Distributed learning of cooperative robotic behaviors using particle swarm optimization. In *Experimental Robotics*, pages 591–604. Springer.

Emery-Montemerlo, R. (2005). *Game-Theoretic Control for Robot Teams*. PhD thesis, The Robotics Institute. Carnegie Mellon University.

Emery-Montemerlo, R., Gordon, G., Schneider, J., and Thrun, S. (2005). Game theoretic control for robot teams. In *Robotics and Automation, 2005. ICRA 2005. Proceedings of the 2005 IEEE International Conference on*, pages 1163–1169. IEEE.

Evans, J. and Krishnamurthy, B. (1998). Helpmate®, the trackless robotic courier: A perspective on the development of a commercial autonomous mobile robot. In *Autonomous Robotic Systems*, volume 236, pages 182–210. Springer London.

Frasheri, M., Cürüklü, B., and Ekström, M. (2018). Comparison between static and dynamic willingness to interact in adaptive autonomous agents. In *Proceedings of the 10th International Conference on Agents and Artificial Intelligence - Volume 1: ICAART,*, pages 258–267.

Fudenberg, D. and Levine, D. (1998). *The theory of Learning in Games*. The MIT Press.

Fujita, W., Moriyama, K., ichi Fukui, K., and Numao, M. (2016). Adaptive two-stage learning algorithm for repeated games. In *Proceedings of the 8th International Conference on Agents and Artificial Intelligence - Volume 1: ICAART,*, pages 47–55.

Govindan, S. and Wilson, R. (2003). A global newton method to compute nash equilibria. *Journal of Economic Theory*, 110(1):65–86.

Harsanyi, J. C. and Selten, R. (1972). A generalized nash solution for two-person bargaining games with incomplete information. *Management Science*, 18(5-part-2):80–106.

Hennig, P. (2013). Fast probabilistic optimization from noisy gradients. In *ICML (1)*, pages 62–70.

Jiang, A. X. and Leyton-Brown, K. (2010). Bayesian action-graph games. In *Advances in Neural Information Processing Systems*, pages 991–999.

Kho, J., Rogers, A., and Jennings, N. R. (2009). Decentralized control of adaptive sampling in wireless sensor networks. *ACM Transactions on Sensor Networks (TOSN)*, 5(3):19.

Kitano, H., Tadokoro, S., Noda, I., Matsubara, H., Takahashi, T., Shinjou, A., and Shimada, S. (1999). Robocup rescue: Search and rescue in large-scale disasters as a domain for autonomous agents research. In *Systems, Man, and Cybernetics, 1999. IEEE SMC'99 Conference Proceedings. 1999 IEEE International Conference on*, volume 6, pages 739–743. IEEE.

Koller, D. and Milch, B. (2003). Multi-agent influence diagrams for representing and solving games. *Games and Economic Behavior*, 45(1):181–221.

Kostelnik, P., Hudec, M., and Šamulka, M. (2002). Distributed learning in behaviour based mobile robot control. *Intelligent Technologies-Theory and Applications*.

Leslie, D. S. and Collins, E. J. (2006). Generalised weakened fictitious play. *Games and Economic Behavior*, 56(2):285–298.

Madhavan, R., Fregene, K., and Parker, L. (2004). Distributed cooperative outdoor multirobot localization and mapping. *Autonomous Robots*, 17(1):23–39.

Miyasawa, K. (1961). On the convergence of learning process in a 2x2 non-zero-person game.

Monderer, D. and Shapley, L. (1996). Potential games. *Games and Economic Behavior*, 14:124–143.

Nachbar, J. (1990). Evolutionary' selection dynamics in games: Convergence and limit properties. *International Journal of Game Theory*, 19:59–89.

Nash, J. (1950). Equilibrium points in n-person games. In *Proceedings of the National Academy of Science, USA*, volume 36, pages 48–49.

Oliehoek, F. A., Spaan, M. T., Dibangoye, J. S., and Amato, C. (2010). Heuristic search for identical payoff bayesian games. In *Proceedings of the 9th International Conference on Autonomous Agents and Multiagent Systems: volume 1-Volume 1*, pages 1115–1122. International Foundation for Autonomous Agents and Multiagent Systems.

Paruchuri, P., Pearce, J. P., Tambe, M., Ordonez, F., and Kraus, S. (2007a). An efficient heuristic for security against multiple adversaries in stackelberg games. In *AAAI Spring Symposium: Game Theoretic and Decision Theoretic Agents*, pages 38–46.

Paruchuri, P., Pearce, J. P., Tambe, M., Ordonez, F., and Kraus, S. (2007b). An efficient heuristic for security against multiple adversaries in stackelberg games. In *AAAI Spring Symposium: Game Theoretic and Decision Theoretic Agents*, pages 38–46.

Raffard, R. L., Tomlin, C. J., and Boyd, S. P. (2004). Distributed optimization for cooperative agents: Application to formation flight. In *Decision and Control, 2004. CDC. 43rd IEEE Conference on*, volume 3, pages 2453–2459. IEEE.

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

Rosen, D. M. and Leonard, J. J. Nonparametric density estimation for learning noise distributions in mobile robotics.

Ruan, S., Meirina, C., Yu, F., Pattipati, K. R., and Popp, R. L. (2005). Patrolling in a stochastic environment. In *10 International Symposium on Command and Control*.

Sakka, S. (2013). *Bayesian Filtering and Smoothing*. Cambridge University Press.

Sandholm, T., Gilpin, A., and Conitzer, V. (2005). Mixed-integer programming methods for finding nash equilibria. In *Proceedings of the National Conference on Artificial Intelligence*, volume 20.

Semsar-Kazerooni, E. and Khorasani, K. (2009). Multi-agent team cooperation: A game theory approach. *Automatica*, 45(10):2205 – 2213.

Smyrnakis, M. and Galla, T. (2015). Decentralized optimisation of resource allocation in disaster management. In *City Evacuations: An Interdisciplinary Approach*, pages 89–106. Springer.

Smyrnakis, M., Kladis, G. P., Aitken, J. M., and Veres, S. M. (2016). Distributed selection of flight formation in uav missions. *Journal of Applied Mathematics and Bioinformatics*, 6(3):93–124.

Smyrnakis, M. and Leslie, D. S. (2010). Dynamic Opponent Modelling in Fictitious Play. *The Computer Journal*, 53:1344–1359.

Smyrnakis, M. and Veres, S. M. (2016). Fictitious play for cooperative action selection in robot teams. *Eng. Appl. of AI*, 56:14–29.

Tsitsiklis, J. N. and Athans, M. (1985). On the complexity of decentralized decision making and detection problems. *Automatic Control, IEEE Transactions on*, 30(5):440–446.

Verstaevel, N., Boes, J., Nigon, J., d'Amico, D., and Gleizes, M.-P. (2017). Lifelong machine learning with adaptive multi-agent systems. In *Proceedings of the 9th International Conference on Agents and Artificial Intelligence - Volume 2: ICAART,*, pages 275–286.

Zhang, Y., Schervish, M., Acar, E., and Choset, H. (2001). Probabilistic methods for robotic landmine search. In *Proceedings of the 2001 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS '01)*, pages 1525 – 1532.