Available online at www.sciencedirect.com
ScienceDirect
Engineering Applications of Artificial Intelligence 00 (2016) 1–28
Journal Logo

Fictitious play for cooperative action selection in robot teams

M. Smyrnakis[a], S. Veres[a]

[a]*Department of Automatic Control and Systems Engineering The University of Sheffield Mappin Street Sheffield, S1 3JD United Kingdom*

Abstract

A game theoretic distributed decision making approach is presented for the problem of control effort allocation in a robotic team based on a novel variant of fictitious play. The proposed learning process allows the robots to accomplish their objectives by coordinating their actions in order to efficiently complete their tasks. In particular, each robot of the team predicts the other robots' planned actions while making decisions to maximise their own expected reward that depends on the reward for joint successful completion of the task. Action selection is interpreted as an n-player cooperative game. The approach presented can be seen as part of the *Belief Desire Intention* (BDI) framework, also can address the problem of cooperative, legal, safe, considerate and emphatic decisions by robots if their individual and group rewards are suitably defined. After theoretical analysis the performance of the proposed algorithm is tested on four simulation scenarios. The first one is a coordination game between two material handling robots, the second one is a warehouse patrolling task by a team of robots, the third one presents a coordination mechanism between two robots that carry a heavy object on a corridor and the fourth one is an example of coordination on a sensors network.

Keywords: Robot team coordination; Fictitious play; Extended Kalman filters; Game theory; Distributed optimisation

1. Introduction

Recent advances in industrial automation technology often require distributed optimisation in a multi-agent system where each agent controls a machine. An application of particular interest, addressed in this paper through a game-theoretic approach, is the coordination of robot teams. Teams of robots can be used in many domains such as mine detection [46], medication delivery in medical facilities [13], formation control [30] and exploration of unknown environments [34, 35, 23]. In these cases teams of intelligent robots should coordinate in order to accomplish a desired task. When autonomy is a desired property of a multi-robot system then self-coordination is necessary between the robots of the team. Applications of these methodologies also include wireless sensor networks [24, 20, 45, 21], smart grids [40, 2], water distribution system optimisation [44] and scheduling problems [37].

Game theory has also been used to design optimal controllers when the objective is coordination, see e.g. [33]. Using this approach, in [33, 32] the agents/robots eventually reach the Nash equilibrium of a coordination game. Another approach is presented in [3], based on agents' cost functions, which use local components and the assumption that the states of the other agents are constant.

Fictitious play is an iterative learning process where players choose an action that maximises their expected rewards based on their beliefs about their opponents' strategies. The players update these beliefs after observing their opponents' actions. Even though fictitious play converges to the Nash equilibrium for certain categories of games [31, 25, 27, 26, 15], this convergence can be very slow because of the assumption that players use a fixed strategy

Email addresses: m.smyrnakis@sheffield.ac.uk (M. Smyrnakis), s.veres@sheffield.ac.uk (S. Veres)

in the whole game [15]. Speed up of the convergence can be facilitated by an alternative approach, which was presented in [36], where opponents' strategies vary through time and players use particle filters to predict them. Though providing faster convergence, this approach has the drawback of high computational costs of the particle filters. In applications where the computational cost is important, as the coordination of many UAVs, the particle filters approach is intractable. An alternative, that we propose here, is to use extended Kalman filters (EKF) instead of predicting opponents' strategies using particle filters. EKFs have much smaller computational costs than the particle filter variant of fictitious play algorithm that has been proposed in [36]. Moreover, in contrast to [36], we provide a proof of convergence to Nash equilibrium of the proposed learning algorithm for potential games. Potential games are of particular interest as many distributed optimisation tasks can be cast as potential games. Therefore convergence of an algorithm to the the Nash equilibrium of a potential game is equivalent to convergence to the global or local optimum of the distributed optimisation problem.

Thus the proposed learning process can be used as a design methodology for cooperative control based on game theory, which overlaps with solutions in the area of distributed optimisation [10]. Each agent i strives to maximise a global control reward, as negative of the control cost, through minimising its private control cost, which is associated with the global one. The private cost function of an agent i incorporates terms that not only depend on agent i, but also on costs associated with the actions of other agents. As the agents strive to minimise a common cost function through their individual ones, the problem addressed here can also be seen as a distributed optimisation problem. In this work we enable the agents to learn how they minimise their cost function through communication and interaction with other agents, instead of finding the Nash equilibrium of the game, which is not possible in polynomial time for some games [12]. In the proposed scheme robots learn and change their behaviour according to the other robots' actions. The learning algorithm, which is based on fictitious play [9], serves as the coordination mechanism of the controllers of team members. Thus, in the proposed cooperative control methodology there is an implicit coordination phase where agents learn other agents' policies and then they use this knowledge to decide on the action that minimises their cost functions. Additionally the proposed control module can be seen as a part of the BDI framework since agents update their beliefs about their opponents' strategies given the state of the environment.

The remainder of this paper is organised as follows. We start with a brief description of relevant game-theoretic definitions. Section 3 present some background material about rational agents. Section 4 introduces the learning algorithm that we use in our cooperative game-based robot cooperation controller, Section 5 contains the main theoretical results and Section 6 contains the simulation results in order to define the parameters of the proposed algorithm. Section 7 presents simulation results before conclusions are drawn.

2. Game theoretical definitions

In this section we will briefly present some basic definitions from game theory, since the learning block of our controller is based on these. A game Γ is defined by a set of players $\mathcal{I}$, $i \in \{1, 2, \ldots, \mathcal{I}\}$, who can choose an action, s^i, from a finite discrete set S^i. We then can define the joint action s, $s = (s^1, \ldots, s^{\mathcal{I}})$, that is played in a game as an element of the product set $S = \times_{i=1}^{i=\mathcal{I}} S^i$. Each player i receives a reward, r^i, after choosing an action s^i. The reward, also called the utility, is a map from the joint action space to real numbers, $r^i : S \rightarrow R$. 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. When players select their actions using a probability distribution they use mixed strategies. The mixed strategy of a player i, σ^i, is an element of the set Δ^i, where Δ^i is the set of all the probability distributions over the action space S^i. The joint mixed strategy, σ, is then an element of $\Delta = \times_{i=1}^{i=\mathbb{I}} \Delta^i$. A strategy where a specific action is chosen with probability 1 is referred to as pure strategy. Analogously to the joint actions we will write $\sigma = (\sigma^i, \sigma^{-i})$ for mixed strategies. The expected utility a player i will gain if it chooses a strategy σ^i (resp. s^i), when its opponents choose the joint strategy σ^{-i}, is denoted by $r^i(\sigma^i, \sigma^{-i})$ (resp. $r^i(s^i, \sigma^{-i})$).

A game, depending on the structure of its reward functions, can be characterised either as competitive or as a coordination game. In competitive games players have conflicted interest and there is not a single joint action where all players maximise their utilities. Zero sum games are a representative example of competitive games where the reward of a player i is the loss of other players . An example of a zero-sum game is presented in Table 1. On the other hand in coordination games players either share a common reward function or their rewards are maximised in the same joint action. A very simple example of a coordination game where players share the same rewards, is depicted in Table 2. Even though competitive games are the most studied games, we will focus our work on coordination games

	Head	Tails
Head	1,-1	-1,1
Tails	-1,1	1,-1

Table 1. Rewards of two players in a zero sum game as function of the outcome of throwing a coin: matching pennies game.

	L	R
U	1,1	0,0
D	0,0	1,1

Table 2. Rewards of two players in a simple coordination game as function of joint moves to the left & up (L,U) or right & down (R,D).

because they naturally formulate a solution to distributed optimisation and coordination.

2.1. Best response and Nash Equilibrium

A common decision rule in game theory is best response. Best response is defined as the action that maximises players' expected utility given their opponents' strategies. Thus for a specific mixed strategy σ^{-i} we evaluate the best response as:

$$\hat{\sigma}^i_{pure}(\sigma^{-i}) = \underset{s^i \in S}{\operatorname{argmax}} \quad r^i(s^i, \sigma^{-i}) \tag{1}$$

A joint mixed strategy $\hat{\sigma} = (\hat{\sigma}^i, \hat{\sigma}^{-i})$ is called a Nash equilibrium. Nash in [28] showed that every game has at least one equilibrium which satisfies:

$$r^i(\hat{\sigma}^i, \hat{\sigma}^{-i}) \geq r^i(\sigma^i, \hat{\sigma}^{-i}) \qquad \text{for any } i \text{ and } \sigma^i \in \Delta^i \tag{2}$$

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

2.2. Distributed optimisation by potential games

It is possible to cast distributed optimisation problems as potential games [1, 11], thus the task of finding an optimal solution for a distributed optimisation problem can be seen as the search for a Nash equilibrium in a game. An optimisation problem can be solved distributively if it can be divided into $\mathcal{D}$ coupled or independent sub-problems with the following property [4]:

$$r(s) - r(\tilde{s}) > 0 \Leftrightarrow r^i(s^i) - r^i(\tilde{s}^i) > 0, \; i \in \mathcal{I}, \forall s, \tilde{s} \tag{3}$$

where s and $\tilde{s}$ are any sets of actions by the agents , r represents the global reward function or global utility, and r^i, $i \in \mathcal{I}$, represent players' reward or utility. Equation (3) implies that a joint action s, should have similarly positive or negative impact on the global and the local task. Thus a solution s should increase or decrease both the local and global utility when it is compared with another joint action $\tilde{s}$.

There is a direct analogy between (3) and ordinal potential games. Ordinal potential games are games where the reward function has a potential function with the following property [26]

$$r^i(s^i, s^{-i}) - r^i(\tilde{s}^i, \tilde{s}^{-i}) > 0 \Leftrightarrow \phi(s^i, s^{-i}) - \phi(\tilde{s}^i, \tilde{s}^{-i}) > 0, \forall s = (s^i, s^{-i}), \; \forall \tilde{s} = (\tilde{s}^i, \tilde{s}^{-i}) \tag{4}$$

where ϕ is a potential function and the above equality stands for every player i. Exact potential games (or potential games thereafter), is a subclass of ordinal potential games which can be used to solve distributed optimisation problems where the difference in the global reward between two joint actions is the same as the difference in the potential function [26]:

$$r^i(s^i, s^{-i}) - r^i(\tilde{s}^i, \tilde{s}^{-i}) = \phi(s^i, s^{-i}) - \phi(\tilde{s}^i, \tilde{s}^{-i}), \; \forall s = (s^i, s^{-i}), \; \forall \tilde{s} = (\tilde{s}^i, \tilde{s}^{-i}) \tag{5}$$

where similarly to ordinal potential games ϕ is a potential function and the above equality stands for every player i. An advantage of potential games is that they have at least one pure Nash equilibrium, hence there is at least one joint action s where no player can increase their reward, i.e. their potential function, through a unilateral change of action.

It is often feasible to choose appropriate forms of agents' utility functions and also the global utility in order to enable the existence and use of a potential function of the system. In this paper we assume that all players/robots share the same global reward. There are cases where, because of communication or other physical constraints, only reward functions that are shared among groups of robots can be used. But even in these cases it is possible to create reward functions that can act as potentials. Wonderful life utility is such a utility function. It was introduced in [41] and applied in [1] to formulate distributed optimisation tasks as potential games. Player i's utility, when wonderful life utility is used, can be defined as the difference between the global utility r_g and the utility of the system when a reference action is used as Player i's action. More formally when Player i chooses an action s^i one can define

$$r^i(s^i) = r_g(s^i, s^{-i}) - r_g(s^i_0, s^{-i}), \forall s^i,\ \forall s^{-i} \tag{6}$$

where s^i_0 denotes a reference action for player i. A reference action is introduced because in strategic games there is no action to represents the case when the player chooses to take no action.

3. Rational agent cooperation

3.1. Agent Definitions

By analogy to previous definitions [22, 42, 6, 7] of AgentSpeak-like architectures, we define our agents as a tuple:

$$\mathcal{R} = \{\mathcal{F}, B, L, \Pi, A\} \tag{7}$$

where:

- $\mathcal{F} = \{p_1, p_2, \ldots, p_{n_p}\}$ is the set of all predicates.
- $B \subset \mathcal{F}$ is the total set of belief predicates. The current belief base at time t is defined as $B_t \subset B$. Beliefs that are added, deleted or modified can be either called *internal* or *external* depending on whether they are generated from an internal action, in which case are referred to as "mental notes", or from an external input, in which case they are called "percepts".
- $L = \{l_1, l_2, \ldots\ l_{n_l}\}$ is a set of logic-based implication rules in terms of predicates.
- $\Pi = \{\pi_1, \pi_2, \ldots, \pi_{n_\pi}\}$ is the set of executable plans or *plans library*.

  Current applicable plans at time t are part of the subset applicable plan $\Pi_t \subset \Pi$ or "desire set".
- $A = \{a_1, a_2, \ldots, a_{n_a}\} \subset \mathcal{F} \setminus B$ is a set of all available actions. Actions can be either *internal*, when they modify the belief base or data in memory, or *external*, when they are linked to external functions that operate in the environment.

Some AgentSpeak like languages, including the limited instruction set agent [17], can be fully defined and implemented by listing the following items:

- *Initial Beliefs*.
  The initial beliefs and goals $B_0 \subset F$ are a set of literals when the agent reasoning cycle is first run.
- *Initial Actions*.
  Initial actions $A_0 \subset A$ are a set of actions that are executed before any reasoning cycle is run is first run. The actions are generally goals that activate initialisations of hardware and software and wake up procedures for perception processes repeated run during each reasoning cycle.

The following three operations are repeated for each reasoning cycle.

- *Maintenance of Percepts*. This means generation of perception predicates for B_t and data objects such as the world model.
- *Logic rules*.
  A set of logic based implication rules L, which describe reasoning to improve the agent's current knowledge about the world.
- *Executable plans*.
  A set of *executable plans* or *plan library* Π. Each plan π_j is described in the form:

$$p_j : c_j \leftarrow a_1, a_2, \ldots, a_{n_j} \tag{8}$$

  where $p_j \in B$ is a *triggering predicate*, which prompts the plan to be retrieved from the plan library whenever it appears in the current belief base, $c_j \in B$ is a logic formula of a *context*, which helps the agent to check the condition of the world, described by the current belief set B_t, before applying a particular plan sequence $a_1, a_2, \ldots, a_{n_j} \in A$ with a list of actions. Each a_j can be one of (1) predicate of an external action with arguments of names of data objects, (2) internal (mental note) with a preceding + or - sign to indicate whether the predicate needs to be added or taken away from the belief set B_t (3) conditional set of items from (1)-(2).

The reasoning cycle of LISA used in this paper consists of the following steps (Fig. 1):

1. *Belief base update*.
   The agent updates the belief base by retrieving information about the world through perception and communication. The job is done by two functions, called *Belief Update Function (BUF)* and *Belief Review Function (BRF)*. The BUF takes care of adding and removing beliefs from the belief base; the BRF updates the set of current events E_t, which can be trigger predicates in plans, by looking at the changes in the belief base.
2. *Application of logic rules*.
   The logic rules in L are applied in a round-robin fashion (restarting at the beginning of the list) until there are no new predicates generated for B_t. This means that rules need to be verified not to lead to infinite loops at the design stage of the agent program.
3. *Trigger Event Selection*.
   For every reasoning cycle a function called *Intention Set Function* selects the current event set E_t:

$$S_t : \wp(B_t) \rightarrow \wp(E_t) \tag{9}$$

   where $\wp(\cdot)$ is the *power operator* and represents the set of all possible subset of a particular set. We will call the current trigger set $S_t(B) = T_t$ and the associated plans the *Intention Set*.
4. *Plan Selection*.
   All the triggered plans in T_t are checked for their context to form the *Applicable Plans* set Π_t, denoted by π_t.
5. *Plan Executions*.
   All plans in π_t are started to be executed concurrently by going through the plan items $a_1, a_2, \ldots, a_{n_j}$ one-by-one sequentially.

3.2. Cooperation cycle

The robots review their actions at a much slower rate than their reasoning cycle operates. During a time period of cooperation cycle each of the agents observes the other's actions, learns from it, defines a reward function relevant to the environmental circumstances, performs optimisation to select its own action or continues the action undergoing. In most practical situations the currently executed actions is one of the options for selection. The agent theory described in the previous subsection can be used to make the cooperative game-based process well coordinated among the agents using logic rules, perception cycles and executable plans allocated.

Slower than the reasoning cycle, is the asynchronous *cooperation cycle*. Its length in time can vary depending on the agent's and circumstances. It is asynchronous across the agent set as availability of communication is not guaranteed between all the agents at all times. Each agent executes the following steps during its operational cycle, which is an integer multiple $N_o \geq 3$ of its reasoning cycle. Note the updating of a simultaneous localisation and mapping-based world model W. The following steps are carried out during the reasoning cycles in the operational cycle.

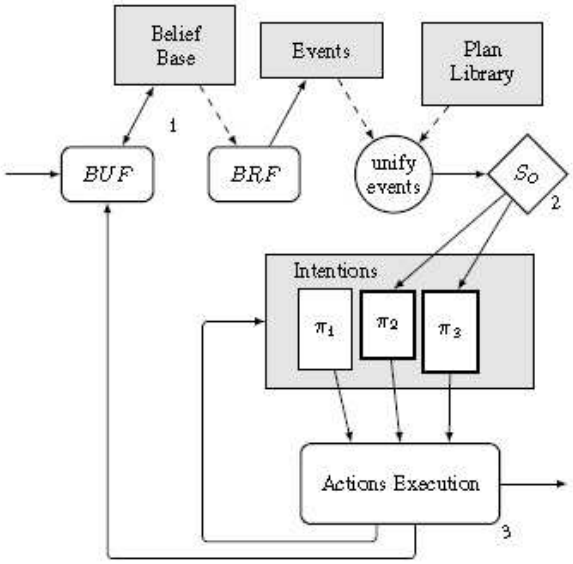

Figure 1. Reasoning cycle of LISA: the plans run in a multi-threaded way, avoiding the need of S_E and S_I

1. Agent i performs estimation of other agent's actions set s^{-i} by analysing changes in W in time. The plan associated with this is

   $$estim\_agents : cooperate \leftarrow estim\_movements(W, Am), infer\_activities(Am, Sni), opt(R, Sni, Act), trigger(Act).$$

   where Am is available movement description data for the rest of the agents in the team and Sni is a notation for s^{-i}. Each item in $s \in s^{-i}$ can be a pair describing the situation and the action the agent is taking. The reward function $r^i(s^i, s^{-i})$ is represented by a data object R and $opt(R, Sni, Act)$ is an optimisation procedure to obtain the best action Act.
2. $trigger(Act)$ adds a predicate $pAct$ to B_{t+1} to trigger the action plan associated with Act:

   $$pAct : \tilde{}active(pAct)\ \&\ \tilde{}\ goal(reached) \leftarrow perform(pAct).$$

   Here the predicate $goal(reached)$ can be established by logic inference in L during reasoning cycles involving predicates generated by perception.
3. Finally there is a mission completion plan to be automatically executed when the mission is completed.

   $$goal(reached) : mission\_started \leftarrow return\_to\_base(B), -misson\_started.$$

   where $return\_to\_base(B)$ is a repeated action of returning with collision avoidance to a base described by data object B.

The above schema is generally applicable, can be particularised for its actions and can be added to make and agent cooperative by fictitious play.

3.3. Example

Consider the following case of a UAV team as an example. Assume that one of the UAVs in the team finds itself with limited battery power and it can choose between two actions. One is that it quickly lands as this action has small control cost and can therefore be performed safely. The second action is to pick up a parcel first, which has high control cost and can result in unsafe landing later. In addition, consider the case where the action of landing will affect negatively the performance of other UAVs and the team's mission will fail, while picking up the parcel would result in a successful performance of the desired joint task.

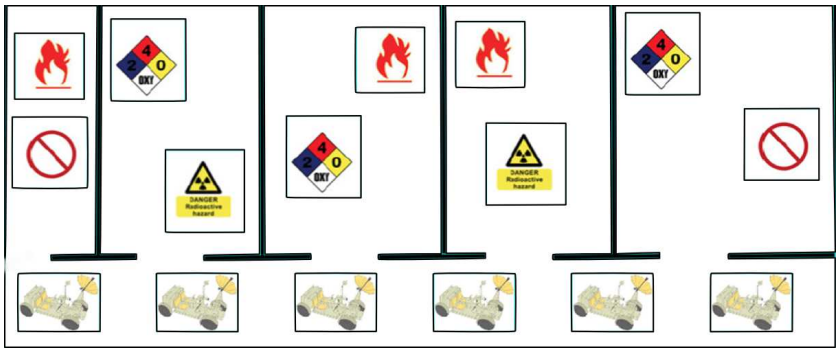

Figure 2. Vehicles should patrol five areas of a warehouse. Each area has two objects of different attribute and therefore different value. The vehicles, according to their capabilities, should choose a joint action that will minimise their common cost function.

The following case study, which is studied in simulation, shows how the proposed game theoretic decision making mechanism works. We consider a team of robots who are to coordinate their team in order to identify possible threats in a warehouse, where we assume that there are $\mathcal{N}$ rooms with some hazardous items. The materials have different attributes of the following categories: flammable, chemical, radioactive and security sensitive. In each room there can be items that belong up to two different categories but there are constraints. For instance, not allowed to have in the same room flammable, radioactive and chemical materials together . Each of the $\mathcal{I}$ robots of the team is equipped with sensors which can sense the different attributes and capabilities. Figure 2 depicts an example of this scenario. The set of rooms correspond to the set of possible placement actions of each robot and is denoted by C. In each of these delivery states there are control actions of the robots described by dynamical models. As an example, we can consider the task of moving towards the region of interest. Each robot i should choose one of the available rooms n, based on the cost of the total control costs of moving to room n, but also the estimated decisions of the other robots and the suitability of robot i to examine room n for its hazardous materials.

In order to create the performance model of the proposed decision making process we need to define a utility function. Utility functions have been used as metrics of the robots coordination efficacy in applications, such as [29, 47, 38, 8, 39]. Since the players of the game will maximise their expected reward, the utility function can be seen as the negative of a cost function. Therefore we can include in the utility any control costs of the robots and environmental elements of the robots' tasks and the constraints that might arise from specific tasks. The function of the control cost should take into account the spatial characteristics of the problem like the cost to the robot to move towards a specific position. The environmental part of the cost function takes into account costs that arise from the nature of the coordination problem and can also include the quality of the sensors of a robot, the aptness of the robots to perform specific tasks, etc. The differences between the robots can also be expressed in terms of their endurance in a specific environment, their efficiency to accomplish a specific task and the presence of the correct sensor to identify a specific threat. Fuzzy variables can be used to quantify robots' efficacy. Two fuzzy variables that can be used are sensors' quality and battery life. The values of battery life are short, fair, long and values of sensors' quality are low, medium and high. If a robot is not equipped with a specific sensor then its efficiency to detect an event is zero. Each robot should then coordinate with the other robots in order to efficiently choose a region to patrol.

The robots should coordinate and choose a region which they will patrol based on the possible threats that should be detected in each area, their sensors' specifications and the actions of the other robots. Each robot can choose only one region to patrol, but many robots can choose the same area.

Each material in the warehouse has different significance and therefore each room has a different value depending on the objects that are stored there which can be incorporated in $E_i(\cdot)$. In a game theoretic terminology we can define a potential game Γ with $\mathcal{I}$ players which have $\mathcal{N}$ available actions. The utility function that is associated with room n can be defined as:

$$r_n(s) = \sum_{\forall i, s^i = n} r_i^m(s^i) + \sum_{\forall s^i = n, n \in \mathcal{N}} r_i^e(s^i, s^{-i}).$$

where $r_i^m(s^i)$, is a function that depends on the initial position of the robot and the room n, $n \in \mathcal{N}$, and represent the

robot's cost to move towards a specific room and $r_i^e(s^i, s^{-i})$ is the environmental part of the cost.

Logic constraints such as the fact that a robot should not visit an area which it has not the appropriate sensors to patrol or that robots with the same attributes (sensors) should not choose the same area can be included in (3.3) using penalty terms in $r_i^e(s^i, s^{-i})$, i.e if a robot i chooses to select an area which is not suitable of its sensors or then $r_i^e(s^i, s^{-i})$ should have a negative or zero value. Similarly if two robots i and j with the same sensors choose the same area then $r_i^e(s^i, s^{-i})$ and $r_j^e(s^j, s^{-j})$ should also have negative or zero values. The global utility that the robots will share is defined by:

$$r_g(s) = \sum_{n=1}^{n=\mathcal{N}} r_n(s) \tag{10}$$

4. The learning process

In this section we present a combination of fictitious play and extended Kalman filters as the algorithm that we will use in the learning block of the proposed decision making module. We briefly present the classic fictitious play algorithm and how it can be combined with extended Kalman filters in a decision making algorithm.

4.1. Fictitious play

Fictitious play [9], is a widely used learning technique in game theory. In fictitious play each player chooses his action according to the best response of his beliefs about his opponent's strategy.

Initially each player has some prior beliefs about the strategy that his opponent uses to choose an action. These beliefs are expressed by a weighting function $\kappa_t^{i\to j}(s^j)$, denoting the weight of the estimated belief function of Player i that Player j will play action s^j at the t_{th} iteration. The players, after each iteration, update their beliefs about their opponents' strategy and play again the best response to their beliefs. More formally, at the beginning of a game players maintain some arbitrary non-negative initial weight functions $\kappa_0^{i\to j}(s^j)$, $i = 1, \ldots, \mathcal{I}$, $j \in \{1, \ldots, \mathcal{I}\}\backslash\{i\}$ that are updated using the formula [15]:

$$\kappa_t^{i\to j}(s^j) = \kappa_{t-1}^{i\to j}(s^j) + \Im_{s_t^j = s^j} \tag{11}$$

for each j, where $\Im_{s_t^j = s^j} = \left\{ \begin{array}{ll} 1 & \text{if } s_t^j = s^j \\ 0 & \text{otherwise.} \end{array} \right\}$.

Equation (11) suggests that all the observed actions have the same impact. Therefore players assume that their opponents choose their actions using a fixed mixed strategy. It is natural then to use a multinomial distribution to approximate an opponent's mixed strategy. The parameters of the multinomial distribution can be estimated using the maximum likelihood method. The mixed strategy of opponent j is then estimated from the following formula:

$$\sigma_t^{i\to j}(s^j) = \frac{\kappa_t^{i\to j}(s^j)}{\sum_{s' \in S^j} \kappa_t^{i\to j}(s')}. \tag{12}$$

4.2. Fictitious play as a state space model

A more realistic assumption than using (11) is to presume that players are intelligent and change their strategies according to the other players' actions. We follow [36] and will represent the fictitious play process as a state-space model. According to the state space model each player has a propensity $Q_t^i(s^i)$ to play each of their available actions $s^i \in S^i$, and then to form a strategy based on these propensities. Finally players can choose an action based on their strategy and the best response decision rule. Because players have no information about the evolution of their opponents' propensities, we model propensities using a Gaussian autoregressive prior on all propensities [36]. Thus we set $Q_0 \sim N(0, I)$, where I is the identity matrix. Under the assumption that the changes in propensities are small from one iteration of the game to another, the following recursive process can be used to update the values of Q_t according to the value of Q_{t-1} as follows:

$$Q(s_t) = Q(s_{t-1}) + \eta_t \tag{13}$$

where $\eta_t \sim N(0, \chi^2 I)$.

Similarly to the sigmoid function that is widely used in the neural network literature [5], to relate the weights and the observation layer of a neural network, we assume that propensities are connected with players' actions by the following Boltzman formula for every $s^j \in S^j$

$$\Im_{s_t^j = s^j} = \frac{e^{(Q^j(s^j)/\tau)}}{\sum_{\tilde{s} \in S^j} e^{(Q_t(\tilde{s})/\tau)}}. \tag{14}$$

4.3. Kalman filters and Extended Kalman filters

Our objective is to estimate Player i's opponent propensity and thus to estimate the marginal probability $p(Q_t, s_{1:t})$. This objective can be represented as a Hidden Markov Model (HMM). HMMs are used to predict the value of an unobserved variable c_t, the hidden state, using the observations of another variable $z_{1:t}$. There are two main assumptions in the HMM representation. The former one is that the probability of being at any state c_t at time t depends only at the state of time $t-1$, c_{t-1}. The latter one is that an observation at time t depends only on the current state c_t. One of the most common methods to estimate $p(c_{1:t}, z_{1:t})$ is Kalman filters and its variations. Kalman filter [19] is based on two assumptions, the first is that the state variable is Gaussian. The second is that the observations are the result of a linear combination of the state variable. Hence Kalman filters can be used in cases which are represented as the following state space model:

$$\begin{aligned} c_t &= Ac_{t-1} + \xi_{t-1} \text{ hidden layer} \\ y_t &= Bc_t + \zeta_t \text{ observations} \end{aligned} \tag{15}$$

where ξ_t and ζ_t follow a zero mean normal distribution with covariance matrices $\Xi = q_t I$ and $Z = r_t I$ respectively, and A, B are linear transformation matrices. When the distribution of the state variable c_t is Gaussian then $p(c_t|y_{1:t})$ is also a Gaussian distribution, since y_t is a linear combination of c_t. Therefore it is enough to estimate its mean and variance to fully characterise $p(c_t|y_{1:t})$.

Nevertheless in the state space model we want to implement, the relation between Player i's opponent propensity and his actions, which is not linear (14). Thus a more general form of state space model should be used as:

$$\begin{aligned} c_t &= f(c_{t-1}) + \xi_t \\ y_t &= h(x_t) + \zeta_t \end{aligned} \tag{16}$$

where $f(\cdot)$ and $h(\cdot)$, are non-linear functions, ξ_t and ζ_t are the hidden and observation state noise respectively, with zero mean and covariance matrices $\Xi = q_t I$ and $Z = r_t I$ respectively. The distribution of $p(c_t|y_{1:t})$ is not a Gaussian distribution because $f(\cdot)$ and $h(\cdot)$ are non-linear functions. Extended Kalman filter (EKF) provides a simple method to overcome this shortcoming by using a first order Taylor expansion to approximate the distributions of the sate space model in (16). In particular if we let $c_t = m_{t-1} + \epsilon$, where m_t denotes the mean of c_t and $\epsilon \sim N(0, P)$, we can rewrite (16) as:

$$\begin{aligned} c_t &= f(m_{t-1} + \epsilon) + w_{t-1} = f(m_{t-1}) + F_c(m_{t-1})\epsilon + \xi_{t-1} \\ y_t &= h(m_t + \epsilon) + \zeta_t = h(m_t) + H_c(m_t)\epsilon + \zeta_t \end{aligned} \tag{17}$$

where $F_c(m_{t-1})$ and $H_c(m_t)$ is the Jacobian matrix of f and h evaluated at m_{t-1} and m_t, respectively. If we use the transformations of (17) then $p(c_t|y_{1:t})$ is a Gaussian distribution.

Since $p(c_t|y_{1:t})$ is a Gaussian distribution, to fully characterise it, we need to evaluate its mean and its variance. The EKF process [18, 16] estimates this mean and variance in the prediction and the update steps respectively. In the prediction step, at any iteration t, the distribution of the state variable is estimated based on all the observations until time $t-1$, $p(c_t|y_{1:t-1})$. The distribution of $p(c_t|y_{1:t-1})$ is Gaussian and we will denote its mean and variance as m_t^- and P_t^- respectively. During the update step the estimation of the prediction step is corrected in the light of the new observation at time t, so we estimate $p(c_t|y_{1:t})$. This is also a Gaussian distribution and we will denote its mean and variance as m_t and P_t respectively.

The estimates of the state c, $p(c_t|y_{1:t-1})$ and $p(c_t|y_{1:t})$, are evaluated based on the prediction and update steps of the EKF process [18, 16] respectively as follows:

Prediction Step

$$\begin{aligned} \bar{c}_t &= f(\hat{c}_{t-1}) \\ P_t^- &= F(m_{t-1})P_{t-1}F(m_{t-1}) + W_{t-1} \end{aligned}$$

where $\bar{c}_t$ and $\hat{c}_{t-1}$ are the estimates of c for $p(c_t|y_{1:t-1})$ and $p(c_{t-}|y_{1:t-1})$ respectively, and the j, j' element of $F(m_t)$ is defined as

$$[F(m_t^-)]_{j,j'} = \frac{\partial f(c_j, r)}{\partial c_{j'}}|_{c=m_t^-, q=0}$$

Update Step

$$\begin{aligned} v_t &= z_t - h(\bar{c}_t) \\ S_t &= H(\bar{c}_t)P_t^- H^T(\bar{c}_t) + Z \\ K_t &= P_t^- H^T(\bar{c}_t)S_t^{-1} \\ \hat{c}_t &= \bar{c}_t + K_t v_t \\ P_t &= P_t^- - K_t S_t K_t^T \end{aligned}$$

where z_t is the observation vector and the $(j, j)'$ element of $H(c_t)$ is defined as:

$$[H(m_t^-)]_{j,j'} = \frac{\partial h(c_j, r)}{\partial c_{j'}}|_{c=m_t^-, r=0}$$

4.4. Fictitious play and EKF

For the rest of this paper we will only consider inference over a single opponent mixed strategy in fictitious play. Separate estimates will be formed identically and independently for each opponent. We therefore consider only one opponent, and we will drop all dependence on player i, and write s_t, σ_t and Q_t for Player i's opponent's action, strategy and propensity respectively. Moreover for any vector x, $x[j]$ will denote the j_{th} element of the vector and for any matrix y, $y[i, j]$ will denote the $(i, j)_{th}$ element of the matrix.

We can use the following state space model to describe the fictitious play process:

$$\begin{aligned} Q_t &= Q_{t-1} + \xi_{t-1} \\ \Im_{s_t^j = s^j} &= h(Q_t) + \zeta_t \end{aligned} \tag{18}$$

where $\xi_{t-1} \sim N(0, \Xi)$, is the noise of the state process and $\zeta_t \sim N(0, Z)$ is is the error of the observation state with zero mean and covariance matrix Z, which occurs because we approximate a discrete process like best response (1), using a continuous function $h(\cdot)$, where $h_{s(k)}(Q) = \frac{exp(Q_t[s(k)]/\tau)}{\sum_{\tilde{s}\in S} exp(Q_t[\tilde{s}]/\tau)}$, where τ is a temperature parameter. By their definition Ξ and Z can take arbitrary values, their impact on the proposed algorithm will be discussed in Section 6.

Hence we can combine the EKF with fictitious play as follows.

At time $t-1$ Player i has observed action s_{t-1} and based on the update step of the EKF process has an estimate of its opponent's propensity, $\hat{Q}_{t-1}$ with the variance P_{t-1}. Then at time t it uses EKF prediction step to estimate his opponent's propensity. The estimate and its variance are:

$$\begin{aligned} \bar{Q}_t &= \hat{Q}_{t-1} \\ P_t^- &= P_{t-1} + \Xi \end{aligned} \tag{19}$$

Player i then evaluates his opponents strategies using his estimations as:

$$\sigma_t(s_t = k) = \frac{exp(\bar{Q}(s_t = k)/\tau)}{\sum_{\tilde{s}\in S} exp(\bar{Q}(\tilde{s})/\tau)}. \tag{20}$$

Player i then uses the estimate of his opponents' strategy (20) and best responses (1), to choose an action. After observing his opponents' action s_t, Player i corrects his estimate about his opponent's propensity using the update equations of the EKF process. The update equations are:

$$\begin{aligned} v_t &= \Im_{s_t^j = s^j} - h(\bar{Q}_t) \\ S_t &= H(\bar{Q}_t)P_t^- H^T(\bar{Q}_t) + Z \\ K_t &= P_t^- H^T(\bar{Q}_t)S_t^{-1} \\ \hat{Q}_t &= \bar{Q}_t + K_t v_t \\ P_t &= P_t^- - K_t S_t K_t^T \end{aligned}$$

1. At time t agent i maintains estimates of his opponent's propensities up to time $t-1$, $\hat{Q}^j_{t-1}$, with covariance P^j_{t-1} of his distribution.
2. Agent i predicts his opponents' propensities $\bar{Q}^j_{tk},\ j \in \{1,...,\mathcal{I}\}\backslash\{i\},\ k \in S^j$ using (19).
3. Based on the propensities in 2 each agent updates his beliefs about his opponents' strategies using (20).
4. Agent i chooses an action based on the beliefs in 3 and applies best response decision rule.
5. The agent i observes his opponents' action $s^j_{tk},\ j \in \{1,...,\mathcal{I}\}\backslash\{i\}$.
6. The agent update its estimates of all of its opponents' propensities using extended Kalman Filtering to obtain $\hat{Q}^j_t,\ j \in \{1,...,\mathcal{I}\}\backslash\{i\}$.

Table 3. EKF based fictitious play algorithm.

The Jacobian matrix $H(\bar{Q}_t)$ is defined as

$$[H(\bar{Q}_t)]_{j,j'} = \begin{cases} \frac{\sum_{j\neq j'}\exp(\bar{Q}_t[j])\exp(\bar{Q}_t)}{(\sum_j \exp(\bar{Q}_t[j]))^2} & \text{if } j = j' \\ -\frac{\exp(\bar{Q}_t[j])\exp(\bar{Q}_t[j'])}{(\sum_j \exp(\bar{Q}_t[j]))^2} & \text{if } j \neq \text{j'} \end{cases}.$$

Table 3 summarises the fictitious play algorithm when EKF is used to predict opponents' strategies.

4.5. EKF fictitious play as part of the reasoning cycle of the agent

As described in Section 3.2 robots actions and reasoning cycles are not synchronised. After each action of the robot team the utility function can be updated and a new reasoning cycle which will include the EKF fictitious play learning algorithm, can be started. In particular steps 1 and 2 of the the reasoning cycle of the agent, (Section 3.1), can be updated using the proposed learning process. The update of the propensities can be seen as *Belief base update*, since when agents update their propensities, they update their beliefs about other agents actions for a given state of the environment. Then the action selection of EKF fictitious play is the *Trigger Event Selection*. Note here that instead of Best Response other alternatives can be used with EKF fictitious play algorithm as part of the *Trigger Event Selection* such as Smooth Best Response [15].

5. The Main Results

In this section we present our convergence results for games with at least one pure Nash equilibrium. The results are for the EKF fictitious play algorithm of Table 3, when the covariance matrices Ξ and Z are defined as $\Xi = (\tilde{\xi} + \epsilon)I$ and $Z = (1/t)I$ respectively, where $\tilde{\xi}$ is a constant, ϵ is an arbitrarily small Gaussian random variable, $\epsilon \sim N(0, \Psi)$, t is the t_{th} iteration of fictitious play, and I is the identity matrix.

The EKF fictitious play algorithm has the following properties:

Proposition 1. *If at iteration t of the EKF fictitious play algorithm, action k is played from Player i's opponent, then the estimate of his opponent propensity to play action k increases, $\hat{Q}_{t-1}[k] < \hat{Q}_t[k]$. Moreover if we denote $\Delta[i]$ as $\Delta[i] = \hat{Q}_t[i] - \hat{Q}_{t-1}[i]$, then $\Delta[k] > \Delta[j] \forall\, j \in S^j$, where S^j is the action space of the j_{th} opponent of Player i. Therefore since $\sum_{k\in S^j}\sigma(s^j = k) = 1$, σ^j_t will be also increased.*

Proof. The proof of Proposition 1 is on Appendix Appendix A. □

Proposition 1 implies that players i, when they use EKF fictitious play, will learn their opponent's strategies and eventually they will choose the action that will maximise their reward based on their estimation. Nevertheless there are cases where players may change their action simultaneously and become trapped in a cycle instead of converging

to a pure Nash equilibrium. As an example, we consider the symmetric game that is depicted in Table 2. This is a simple coordination game with two pure Nash equilibria of the joint actions (U, L) and (D, R). In the case were the two players start from joint action (U, R) or (D, L) and they always change their action simultaneously, they will never reach one of the two pure Nash equilibria of the game.

Proposition 2. *When the players of a game Γ use an EKF fictitious play process to choose their actions, then with probability 1 they will not change their action simultaneously infinitely often.*

Proof. The proof of Proposition 2 is on Appendix Appendix B. □

Based on Proposition 1 and 2 we can infer the following propositions and theorem.

Proposition 3. *(a) If s is a pure Nash equilibrium of a game Γ, and s is played at time t in the process of EKF fictitious play, s will be played at all subsequent times. That is, strict Nash equilibria are absorbing for the process of EKF fictitious play.*

(b) Any pure strategy steady state of EKF fictitious play must be a Nash equilibrium.

Proof. Consider the case where players beliefs $\hat{\sigma}_t$, are such that their optimal choices correspond to a strict Nash equilibrium $\hat{s}$. In EKF fictitious play process players' beliefs are formed identically and independently for each opponent based on equation (20). By Proposition 1 we know that players' estimations about their opponents' propensities and therefore their strategies will increase for the actions that are included in $\hat{s}$. Thus the best response to their beliefs $\hat{\sigma}_{t+1}$ will be again $\hat{s}$ and since $\hat{s}$ is a Nash equilibrium they will not deviate from it. Conversely, if a player remains at a pure strategy profile, then eventually the assessments will become concentrated at that profile, due to Proposition 1 and so if the profile is not a Nash equilibrium, one of the players would eventually want to deviate. □

Proposition 4. *Under EKF fictitious play, if the beliefs over each player's choices converge, the strategy profile corresponding to the product of these distributions is a Nash equilibrium.*

Proof. Suppose that the beliefs of the players at time t, σ_t, converges to some profile $\hat{\sigma}$. If $\hat{\sigma}$ were not a Nash equilibrium, some players would eventually want to deviate and the beliefs would also deviate since based on Proposition 1 players eventually learn their opponents actions. □

Based on the propositions (1-4) we can show that EKF fictitious play converges to the the Nash equilibrium of games with a better reply path. A game with a better reply path can be represented as a graph in which the edges are the join actions of the game s and there is a vertex that connects s with s' iff only one player i can increase its payoff by changing its action [43]. Potential games have a better reply path [43].

Theorem 1. *The EKF fictitious play process converges to the Nash equilibrium in games with a better reply path.*

Proof. If the initial beliefs of the players are such that their initial joint action s_0 is a Nash equilibrium, from Proposition 3 and equation (20), we know that they will play the joint action which is a Nash equilibrium for the rest of the game.

Moreover, the initial beliefs of the players are such that their initial joint action s_0 is not a Nash equilibrium based on Proposition 1 and Proposition 2 after a finite number of iterations because the game has a better reply path, then the only player that can improve his pay-off by changing his actions will choose a new action which will result in a new joint action s. If this action is not the a Nash equilibrium then again after finite number of iterations the player who can improve his pay-off will change action and a new joint action s' will be played. Thus after the search of the vertices of a finite graph, and therefore after a finite number of iterations, players will choose a joint action which is a Nash equilibrium. Eventually after a finite number of time steps, T, the process will end up in a pure Nash equilibrium. The maximum number of iterations that is needed is the cardinality of the joint action set multiplied with the total number of iterations that is needed in order not to have simultaneous changes, $\binom{n}{2}(t_1 + t_2 + t_3 + \ldots + T)$ □

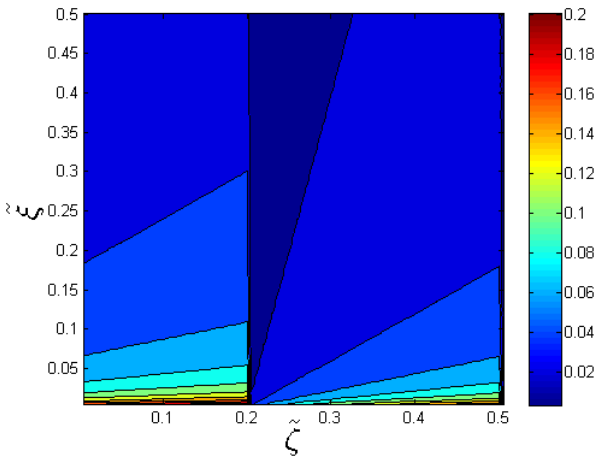

Figure 3. Contour plot of the combined mean square error for the pure and mixed strategy case. The range of ξ and ζ is [0.005, 0.5] . The last value of x-axis corresponds to $\tilde{\zeta} = 1/t$ the

6. Parameters selection

The covariance matrix of the state space error $\Xi = \tilde{\xi}I$ and the measurement error $Z = \tilde{\zeta}I$ are two parameters that we should define at the beginning of the EKF fictitious play algorithm. Our aim is to find values, or range of values, of $\tilde{\xi}$ and $\tilde{\zeta}$ that can efficiently track opponents' strategy when it smoothly or abruptly change, instead of choosing them heuristically for each opponent when we use the EKF algorithm. Nevertheless it is possible that for some games the results of the EKF algorithm will be improved for other combinations of $\tilde{\xi}$ and $\tilde{\zeta}$ than the ones that we propose in this section.

We examine the impact of EKF fictitious play algorithm parameters on its performance, by measuring the mean square error (MSE) between EKF estimation and the real opponent strategy, in the following two tracking scenarios. In the first scenario a single opponent chooses its actions using a mixed strategy which changes smoothly and has a sinusoidal form over the iterations of the tracking scenario. In particular for $t = 1, 2, \ldots, 5000$ iterations of the game: $\sigma_t(1) = \frac{cos\frac{2\pi t}{n}+1}{2} = 1 - \sigma_t(2)$, where $n = 100$. In the second toy example Player i's opponent change its strategy abruptly and chooses action 1 with probability $\sigma_t^2(1) = 1$ during the first and the last 1250 iterations' of the game and for the rest iterations of the game $\sigma_t^2(1) = 0$. The probability of the second action is calculated as: $\sigma_t^2(2) = 1 - \sigma_t^2(1)$.

Figure 3 depicts the mean square error for both tracking scenarios for $0.005 \leq \tilde{\xi} \leq 0.5$ and $0.005 \leq \tilde{\zeta} \leq 0.5$. Additionally we examine the performance of EKF fictitious play for $\tilde{\zeta} = 1/t$ which is depicted as the last element of the x-axis of Figure 3. There is a wide range of combinations of $\tilde{\xi}$ and $\tilde{\zeta}$ that minimise the MSE where the mean for both examples. The MSE is minimised for a wider range of $\tilde{\xi}$ when $\tilde{\zeta} = 1/t$ than the all the cases where a single value of $\tilde{\zeta}$ was used. Moreover the minimum value of the MSE was observed when $\tilde{\zeta} = \frac{1}{t}$ and $\tilde{\xi} = 0.1$. Figures 4 and 5 depict the opponent's strategy-tracking by EKF fictitious play in both examples for these parameters.

7. Simulations

This section contains the results of two simulation scenarios, on which the performance of the proposed learning algorithm was tested. Based on the results of the previous section, we used the following values for Ξ and Z in our simulations, $\Xi_t = (\tilde{\xi} + \epsilon)I$ and $Z_t = \tilde{\zeta}I$, where I is the identity matrix, $\tilde{\xi} = 0.1$, $\tilde{\zeta} = 1/t$ and ϵ is an arbitrarily small Gaussian random number. For both simulation scenarios we report the average results for 100 replications of each learning instance. In each learning instance the agents were negotiating for 50 iterations.

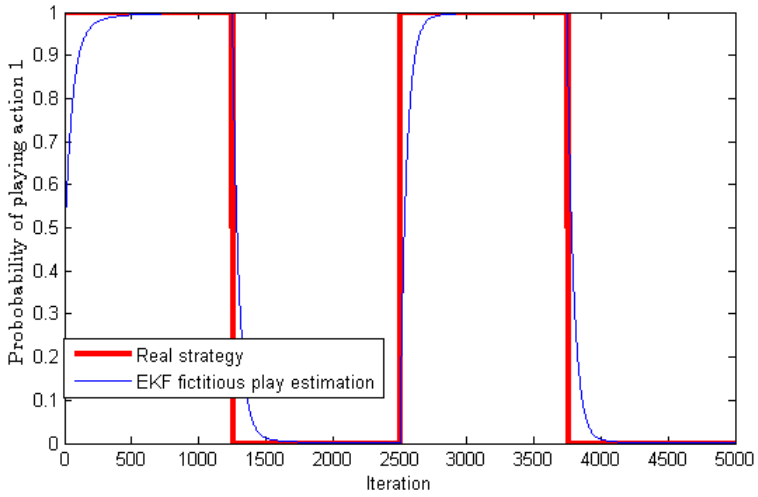

Figure 4. Tracking of opponent strategy at a pure strategy environment, where the pre-specified strategy of the opponent and its prediction are depicted as the red and blue line respectively.

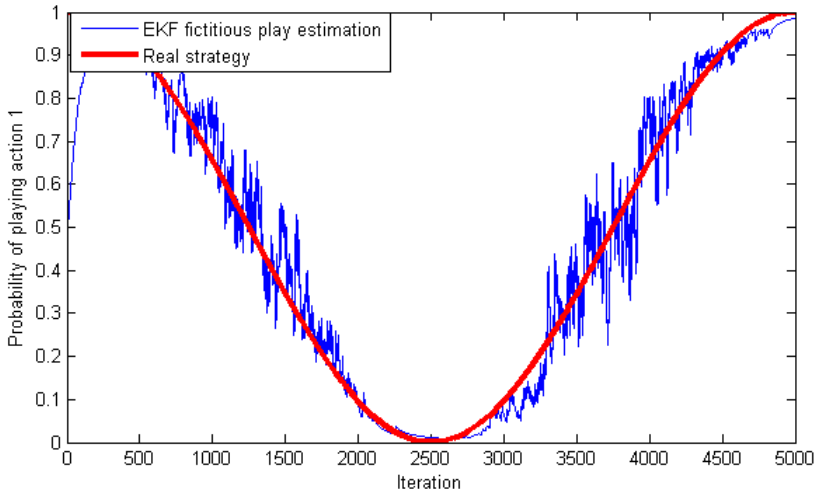

Figure 5. Tracking of opponent strategy at a mixed strategy environment, where the pre-specified strategy of the opponent and its prediction are depicted as the red and blue line respectively.

	Weak	Fair	Strong
Weak	1,1	0,0	0,0
Fair	0,0	1,1	0,0
Strong	0,0	0,0	1,1

Table 4. Players' rewards of a symmetric game.

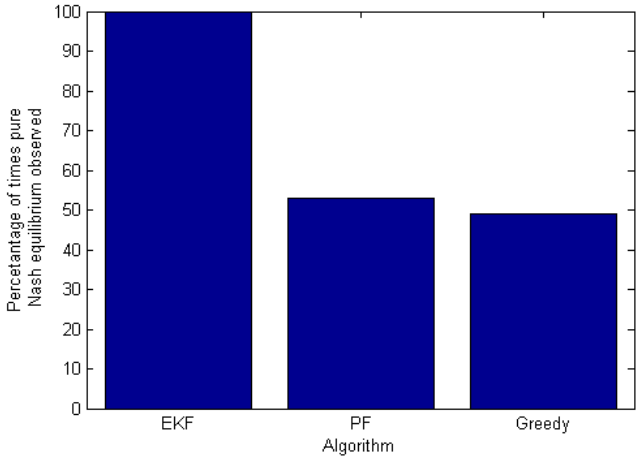

Figure 6. Percentage of the times that each algorithm converged to one of the two optimum joint actions for the game of Table 2.

7.1. Results in symmetric games

We initially tested the performance of the proposed algorithm in symmetric games as they are presented in Tables 2 and 4. Even though these are games that involve only two players, they can be ensued from realistic applications. The game of Table 2 can be seen as a collision avoidance game. Consider the case where two UAVs, the players of the game, should fly in opposite directions. They will gain some reward if they accomplish their task and fly in different altitudes. Therefore players should coordinate and choose one of the two joint actions that maximise their reward, UL and DR, who represent the choices of different altitudes from the UAVs.

Another example of how a symmetric game can be ensued from a realistic application comes from the area of material handling robots. Consider the case where two robots should coordinate in order to move some objects to a desired destination. The robots can either push or pull the objects depending on the direction of the destination of the object. Moreover, each robot can apply different forces to the objects. The amount of the force that a robot applies to an object can be represented as a fuzzy variable that can take the values: Weak, Fair and Strong. This scenario can lead to the game that is described in Table 4.

Since in this paper we focus on the decision making module of a robot's controller, we will compare the proposed learning algorithm of EKF fictitious play with the particle filter alternative [36]. We will use it as a baseline criterion for the greedy choice of the players, i.e the action that maximises the reward of a player independently of what the others are doing. In particular, for the games that are presented in Tables 2 and 4, this is the random action since the rewards for all the actions are the same.

Figures 6 and 7 depict the percentage of the replications where the algorithms converged to one of the optimal solutions for the games depicted in Tables 2 and 4, respectively. Because of the symmetry in the utility function, the particle-filter-based fictitious play behaves similarly to a random action. Thus when the initial beliefs were not supporting one of the pure Nash equilibria of the games, it failed to converge to a joint action with reward. EKF fictitious play on the other hand, always converged to a joint action that maximises players' rewards. Figure 8 shows the average utility that players obtain in each iteration of the game when the initial joint actions have zero reward. As we can observe, after the maximum of 35 iterations, the utility that players receive is 1 and therefore it converges to

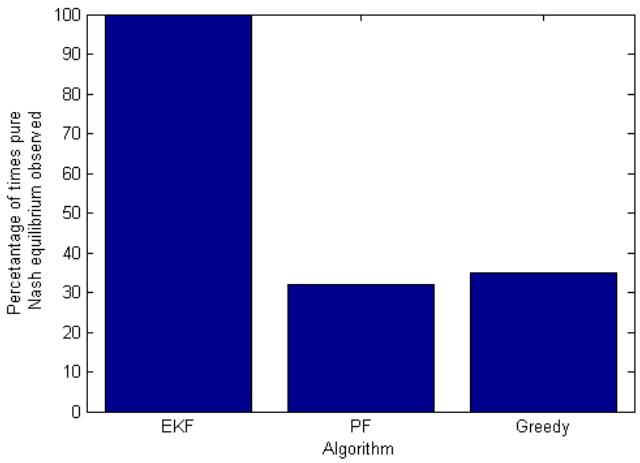

Figure 7. Percentage of the times that each algorithm converged to one of the three optimum joint actions for the game of Table4.

		Battery capacity		
		Short	Fair	Long
Sensors' quality	Low	0.2	0.3	0.5
	Medium	0.3	0.5	07
	High	0.5	0.7	0.9

Table 5. Efficiency that a robot efficiently patrol the area for a specific threat.

one of the pure Nash equilibria.

7.2. Results on a robot team's warehouse patrolling task

We also examined the performance of EKF fictitious play in the task allocation scenario we described in Section 2. A robot team, which consists of $\mathcal{I}$ robots, should examine an area A that is divided in $\mathcal{N}$ regions with hazardous objects. We present simulation results for various combinations of number of regions and sizes of the robots' team. In particular, we examined the performance of the EKF learning algorithm when the team consists of $\mathcal{I}$ robots, $\mathcal{I} \in \{5, 10, 15, 20, 25, 30, 35, 40\}$, who patrolling one of the $\mathcal{N}$ regions, $\mathcal{N} \in \{5, 10, 20\}$.

Each region contains two objects with different attributes. Each robot i then can be equipped with up to three of the following sensors: fire detector, chemical detector, Geiger counter and vision system. Therefore robots with a fire detector should patrol areas with flammable objects, robots with Geiger should patrol areas with radioactive material etc. Moreover each robot can move towards a region n, with a velocity $\mathcal{V}_{in}$. We assume that the velocity of a robot i, towards a region n can be either slow, medium or fast. Therefore the time that a robot needs to reach a region n, T_{in}, depends both on the distance between the robot and the region and the velocity that the robot will choose to move towards n as well. The global reward that is shared among the robots is defined as:

$$\begin{aligned} r_g(s) = \quad & -\sum_{n=1}^{\mathcal{N}} \sum_{i:s^i=n} c_1 T_{in} + c_2 \mathcal{V}_{in} \\ & + \sum_{n=1}^{\mathcal{N}} \sum_{k \in K_n} V_{nk} (1 - \prod_{i:s^i=n} (1 - \mathcal{E}_{ink})) \end{aligned} \tag{21}$$

where $c_1 = 1$, $c_2 = 1/6$ and $\mathcal{E}_{ink}$ is a metric of robot i's efficiency to patrol a region n based on its battery capacity and its sensor quality. It can also be seen as the probability that robot i has to detect a threat k in region n. We define $\mathcal{E}_{ink}$ as a function of robot i's battery capacity and its sensors' quality as it is presented in Table 5. If a robot has no suitable sensor to patrol a specific area $\mathcal{E}_{ink} = 0$. Note that the first term in the summation represents the control cost the robots have when they choose a specific region to allocate and the second term corresponds to the environmental cost.

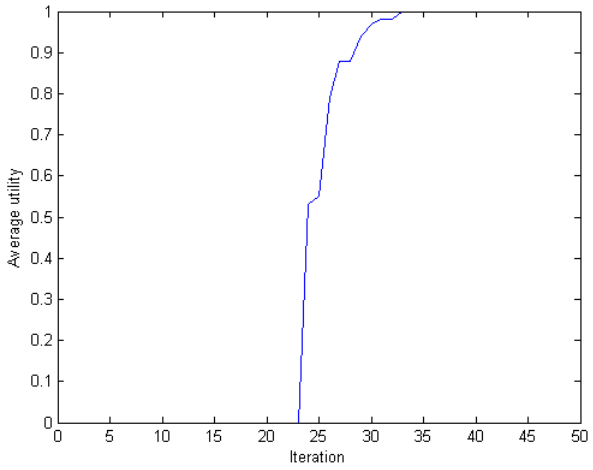

Figure 8. Average reward that players obtain when their initial action is either UR orDL for the game of Table 2 .

Even though this article considers distributed optimisation tasks, a centralised alternative, genetic algorithm (GA), will be also used. Since the centralised optimisation algorithms are expected to perform equally good or better than distributed optimisation algorithms, the performance of GA can be seen as a benchmark solution. Thus a central decision maker will choose in advance the area that each robot will patrol and the robot completes the pre-optimised allocation task. In our comparison we used the genetic algorithm function of Matlab's optimisation toolbox. Since genetic algorithms are stochastic processes, in each repetition of the same replication of the task allocation problem it will converge to a different solution. Thus we used two instances of the genetic algorithm, in the former one we used a single repetition of the genetic algorithm per replication of our problem, in the latter one for each replication of the task allocation problem we chose the allocation of the maximum utility among 50 runs of the genetic algorithm.

In order to be able to average across the 100 replications, we used the following scores for the global rewards players obtained for each learning algorithm they used: $\mathcal{F} = 100 \times \frac{r_g(s)}{r_{max}}$, where r_{max} is the reward that players would have obtained if all all areas had been patrolled with 100% efficiency.

Figures 9, 10 and 11 depict the score players obtain in the final iteration of the algorithm as a function of the number of robots, $\mathcal{I}$, for the tested algorithms when there are 5,10 and 20 areas of interest respectively.

In all cases when players chose the greedy actions, the outcome was very poor, which also indicates the need of the coordination controller that we propose when distributed solutions are considered. When the case with 5 areas is considered, Figure 9, all algorithms performed similarly, when $\mathcal{I} \leq 30$. When $\mathcal{I} \geq 35$ the best performance was observed when the proposed algorithm was used. When the searching areas were increased to 10, Figure 10, the EKF algorithm and the variants of the genetic algorithm have similar performance independently of the number of robots. But its performance is better than the particle filters algorithm when $\mathcal{I} \geq 20$. Finally, when we consider the case with 20 search areas, the EKF fictitious play performs significantly better than both variants of the genetic algorithms we used. But its score is 3% smaller than the one of the particle filter alternative when we consider the case where 40 robots had to coordinate.

For each agent we also tested the average computational cost of the 50 iterations, in seconds, for each of the algorithms we tested. For the genetic algorithms, since they are centralised algorithms, the cost per robot is the cost the algorithm needed to terminate. In order to have comparable results with the centralised cases, for the two distributed algorithms it is assumed that robots are able to communicate. Therefore, before they take an action there is a "negotiation" period among the robots. During this period the robots exchange messages indicating the areas, which they intend to patrol. Then, when the final iteration of the negotiation period is reached, they execute the last joint action. Under this process, the difference in the computational time of each robot is the total computational time of each algorithm. The experiments were performed on a PC with Intel i7 processor at 2.5 GHz and 8 GB memory.

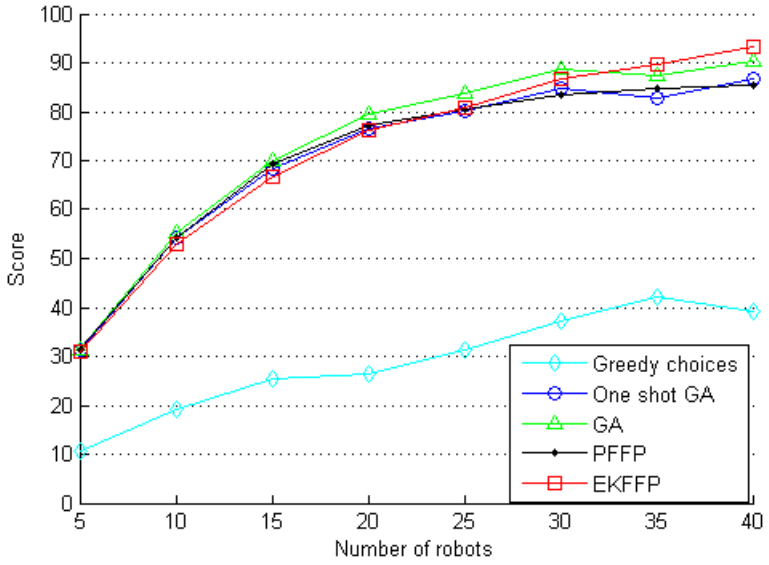

Figure 9. Robot team task allocation outcome for the case where $N = 5$. The x-axis represent the number of robots that was used in the simulation scenario and the y-axis the corresponding average score of the 100 replications of the allocation task.

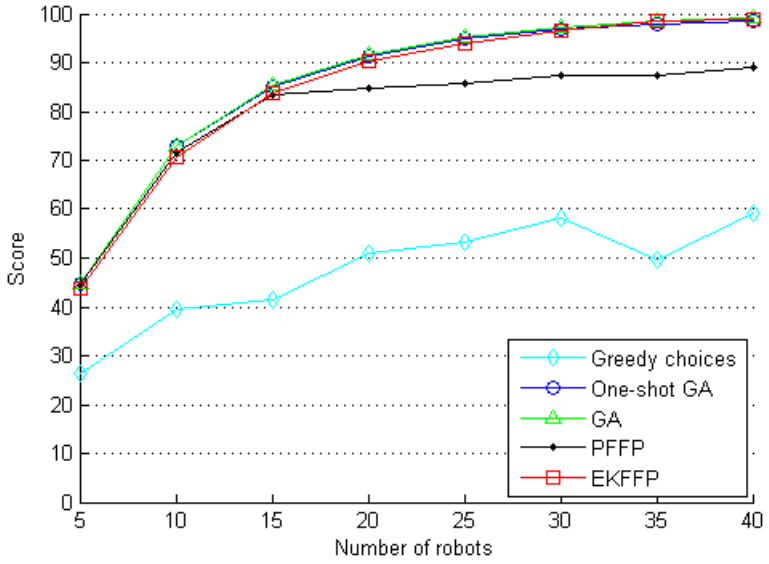

Figure 10. Robot team task allocation outcome for the case where $N = 10$. The x-axis represent the number of robots that was used in the simulation scenario and the y-axis the corresponding average score of the 100 replications of the allocation task.

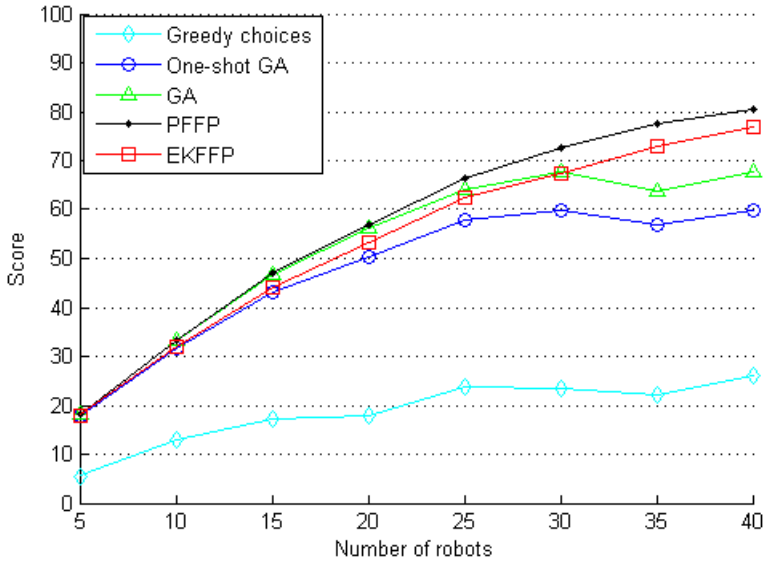

Figure 11. Robot team task allocation outcome for the case where $N = 20$. The x-axis represent the number of robots that was used in the simulation scenario and the y-axis the corresponding average score of the 100 replications of the allocation task.

	One-shot Genetic algorithm	Genetic algorithm	Particle filter fictitious play	EKF fictitious play
5	1.097	56.9	0.185	0.159
10	2.081	105.9	0.399	0.165
15	2.255	110.6	0.616	0.246
20	2.358	117.5	0.886	0.326
25	2.366	121.5	1.089	0.428
30	2.698	159.9	1.374	0.503
35	2.728	132.1	1.558	0.589
40	2.839	136.7	1.917	0.660

Table 6. Average computational time that each robot was needed per replication, 5 areas

Even though the performance of the proposed algorithm is not always improving the results of the existing algorithms we tested, it has the smallest computational cost among them, as it is depicted in Tables 6,7 and 8. This is important in robotics applications, where restrictions, such as battery life, should be taken into account.

7.3. Transportation of a heavy object through corridors.

In this section we describe how the proposed decision making module can be implemented as a coordination mechanism between two robots, which carry a heavy item through a set of possible unmapped corridors. Based on a game-theoretic formulation, after the robots sense that an object or wall block their way, they have to coordinate in order to choose the action that will allow them to continue their mission even when they are not aware of the environment's structure. We assume that the two robots can move in corridors using the moves that are depicted in Figure 13. In order to take into account obstacles and varying widths of corridors, we allowed two possible ways to turn left or right. In particular, Action 1 represents the move forward, Action 2 represents the move forward for a distance z meters and right z meters, Action 3 is a z meter diagonal right move of 45 degrees slope, Action 4 represents the move forward for z meters and left z meters and Action 5 is a z meter diagonal left move of 45 degrees slope. This also introduces the need of a coordination mechanism because there can be cases, such as the set of corridors which are depicted in Figure 12, where both moves can be used by the robots to change direction.

	One-shot Genetic algorithm	Genetic algorithm	Particle filter fictitious play	EKF fictitious play
5	2.056	106.9	0.628	0.674
10	4.353	190.5	1.165	0.705
15	4.318	199.85	1.881	1.105
20	3.296	240.25	2.579	1.704
25	4.313	246.75	3.208	2.119
30	5.092	248.5	3.762	3.143
35	5.275	255.5	4.445	3.899
40	6.023	296.0	5.271	4.487

Table 7. Average computational time that each robot was needed per replication, 10 areas

	One-shot Genetic algorithm	Genetic algorithm	Particle filter fictitious play	EKF fictitious play
5	3.455	136.9	0.826	0.706
10	12.023	556.9	1.906	1.105
15	20.141	749.2	2.932	1.704
20	20.681	870.8	3.276	2.119
25	22.014	875.1	4.955	2.229
30	22.211	909.8	6.138	3.143
35	26.146	911.6	6.754	3.899
40	27.276	944.5	8.399	4.487

Table 8. Average computational time that each robot was needed per replication, 20 areas

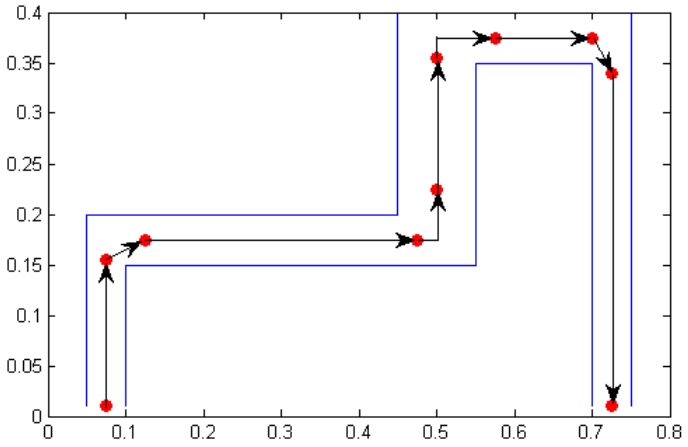

Figure 12. The configuration the corridor used in our simulation in a MATLAB^{TM} figure and illustration of robot moves.

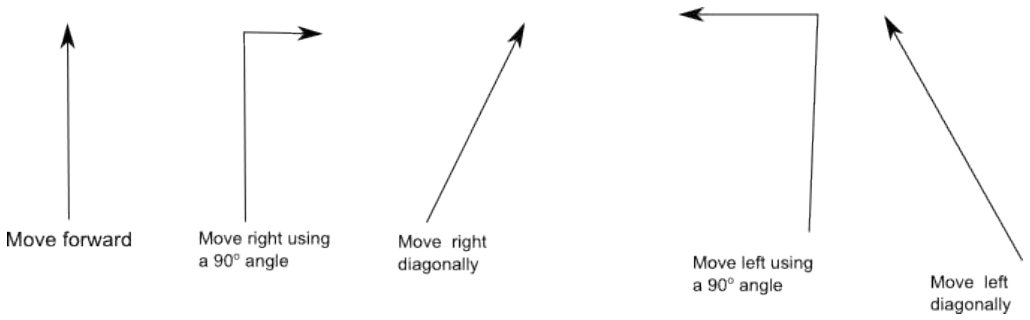

Figure 13. Available moves to each robot

Robots in this scenario have to make decisions in various areas of Figure 12. Each of these areas is assumed to be a checkpoint, at which when the robots arrive, they use the EKF algorithm in order to choose the joint action that will lead them to the next checkpoint. The checkpoints can either be predefined before the task starts, or they can be dynamically generated based on set time intervals or the distance that the robots have covered. In this game, when the robots fail to coordinate, they receive zero utility, and when they coordinate their gain is a constant $\mathfrak{c}$ if the joint action is feasible and zero otherwise. Thus, in each checkpoint the robots play a game with rewards defined as:

$$r(s) = \begin{cases} \mathfrak{c} & \text{if the joint action } s \text{ is feasible and } s^1 = s^2 \\ 0 & \text{otherwise} \end{cases} \tag{22}$$

A joint action s is considered feasible when it is executed and the task can proceed, i.e. during the execution of the action neither the robots nor the hit a wall.

We present the results over 100 replications of the above coordination task. In this task, because of the symmetry of the game, when the robots use the particle filter variant of fictitious play, they fail to coordinate in the majority of times. In particular, only in 34% of the replications of the coordination tasks were successfully completed when the particle filters algorithm was used. On the other hand, when the proposed algorithm was used, the task was completed in all replications.

Figure 14 displays a sample trajectory of the robots for the dynamically generated checkpoints. In this simulation scenario we assume that 0.1 distance units corresponds to 10 meters. The checkpoints are generated when robots move 5 meters from the previous checkpoint, i.e. $z = 5$. This can be seen also as a new checkpoint every 2 seconds when the robots are moving with speed of 2.5m/s.

Table 9 shows the joint actions who can produce some reward (successful joint action) for each check point and the average number of iterations the EKF fictitious play algorithm needed to converge to a solution.

7.4. Ad-hoc sensor network

The simulation scenario of this section is based on a coordination task of a power constrained sensor network, where sensors can be either in a sensing or sleeping mode [14]. When the sensors are in a sensing mode, they can observe the events that occur in their range. During their sleeping mode the sensors harvest the energy they need in order to be able function, when they are in the sensing mode. The sensors then should coordinate and choose their sensing/sleeping schedule in order to maximise the coverage of the events. This optimisation task can be cast as a potential game. In particular we consider the case where $\mathcal{I}$ sensors are deployed in an area where E events occur. If an event e, $e \in E$, is observed from the sensors, then it will produce some utility V_e. Each of the sensors $i = 1, \ldots, \mathcal{I}$ should choose an action $s^i = j$, from one of the $j = 1, \ldots, J$ time intervals which they can be in sensing mode. Each sensor i, when it is in sensing mode, can observe an event e, if it is in its sensing range, with probability $p_{ie} = \frac{1}{d_{ie}}$, where d_{ie} is the distance between the sensor i and the event e. We assume that the probability that each sensor observes an event is independent from the other sensors' probabilities. If we denote by i_{in} the sensors that are in sense mode when the event e occurs and e is in their sensing range, then we can write the probability of an event e to be observed from the sensors, i_{in} as:

$$1 - \prod_{i \in i_{in}} (1 - p_{ie}) \tag{23}$$

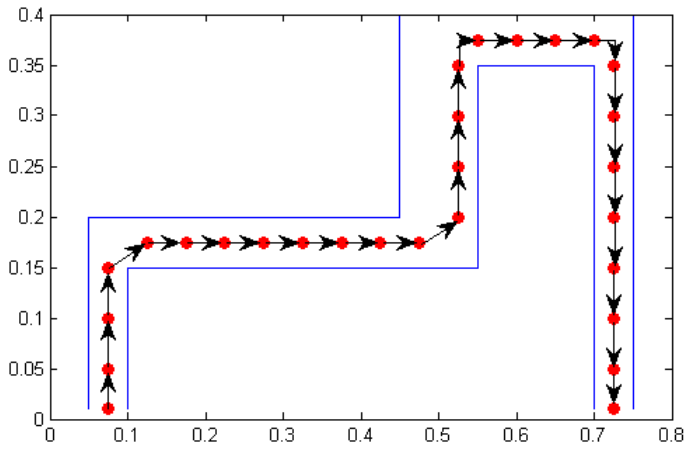

Figure 14. Trajectory of the robots through the checkpoints.

	Successful joint actions	Iterations
Checkpoint 1	(Action 1,Action 1)	1
Checkpoint 2	(Action 1,Action 1)	1
Checkpoint 3	(Action 1,Action 1)	1
Checkpoint 4	(Action 2,Action 2),(Action 3,Action 3)	25.6
Checkpoint 5	(Action 1,Action 1)	1
Checkpoint 6	(Action 1,Action 1)	1
Checkpoint 7	(Action 1,Action 1)	1
Checkpoint 8	(Action 1,Action 1)	1
Checkpoint 9	(Action 1,Action 1)	1
Checkpoint 10	(Action 1,Action 1)	1
Checkpoint 11	(Action 1,Action 1)	1
Checkpoint 12	(Action 4,Action 4),(Action 5,Action 5)	29.8
Checkpoint 13	(Action 1,Action 1)	1
Checkpoint 14	(Action 1,Action 1)	1
Checkpoint 15	(Action 1,Action 1)	1
Checkpoint 16	(Action 1,Action 1)	1
Checkpoint 17	(Action 2,Action 2),(Action 3,Action 3)	22.6
Checkpoint 18	(Action 1,Action 1)	1
Checkpoint 19	(Action 1,Action 1)	1
Checkpoint 20	(Action 1,Action 1)	1
Checkpoint 21	(Action 2,Action 2),(Action 3,Action 3)	27.9
Checkpoint 22	(Action 1,Action 1)	1
Checkpoint 23	(Action 1,Action 1)	1
Checkpoint 24	(Action 1,Action 1)	1
Checkpoint 25	(Action 1,Action 1)	1
Checkpoint 26	(Action 1,Action 1)	1
Checkpoint 27	(Action 1,Action 1)	1

Table 9. Average number of iterations that was needed by EKF fictitious play algorithm in order for the robots to reach consensus.

	EKF fictitious play	Particle filter fictitious play 500 particles	Particle filter 1000 particles
Average time	0.33	5.37	8.43

Table 10. Average computational time for each agent in seconds.

The expected utility which is produced from the event e, is the product of its utility V_e and the probability that it is observed by the sensors i_{in} that are in sensing mode when the event e occurs and e is in their sensing range. More formally, we can express the utility that is produced from an event e as:

$$r_e(s) = V_e(1 - \prod_{i \in i_{in}} (1 - p_{ie})) \tag{24}$$

The global utility is then the sum of the utilities that all events, $e \in E$, produce

$$r_{global}(s) = \sum_e r_e(s). \tag{25}$$

Each sensor, after each iteration of the game, receives some utility, which is based on the sensors and the events that are inside its communication and sensing range respectively. For a sensor i we denote $\bar{e}$ the events that are in its sensing range and $\tilde{s}^{-i}$ the joint action of the sensors that are inside its communication range. The utility that sensor i will receive if its sense mode is j will be

$$r_i(s^i = j, \tilde{s}^{-i}) = \sum_{\bar{e}} r_{\bar{e}}(s^i = j, \tilde{s}^{-i}). \tag{26}$$

In this simulation scenario 40 sensors are placed on a unite square, each have to choose one time interval of the day when they will be in sensing mode and use the rest of the time intervals to harvest energy. We consider cases where sensors had to choose their sensing mode among 4 available time intervals. Sensors are able to communicate with other sensors that are at most 0.6 distance units away, and can only observe events that are at most 0.3 distance units away. The case where 20 events are taking place is considered. Those events were uniformly distributed in space and time, so an event could evenly appear in any point of the unit square area and it could occur at any time with the same probability. The duration of each event was uniformly chosen between $(0 - 6]$ hours and each event had a value $V_e \in (0 - 1]$.

The results of EKF fictitious play are compared with the ones of the particle filter based algorithm. The number of particles in particle filter fictitious play algorithm, play an important role in the computational time of the algorithm. For that reason comparisons were made with three variants of the particle filter algorithm with 500-1000 particles.

The results presented in Figure 15 and Table 10 are the averaged reward of 100 instances of the above described scenario. Figure 15 depicts the the average global reward. The global reward in each instance was normalised using the maximum reward, which is achieved when all the sensors are in sense mode. As it is depicted in Figure 15 the average performance of the particle filter algorithm is not affected by the number of particles in this particular example. But the EKF fictitious play performed better than the particle filters alternatives. In addition, as it is shown in Table 10, the computational cost of the two particle filter's variants is greater than the EKF fictitious play algorithm.

8. Conclusions

A novel variant of fictitious play has been presented. This variant is based on extended Kalman filters and can be used as a decision making module of a robot's controller. The relation between the proposed learning process and the BDI framework was also considered. The main theoretical result has been, that the learning algorithm associated with the cooperative game converges to the Nash equilibrium of potential games. The application of this methodology has been illustrated on four examples. The filtering methods used for learning have also been compared for their performance and computational costs. The methods presented can potentially find widespread applications in the robotics industry. Possible improvements could be obtained by the adoption of more sophisticated reward/cost functions, without altering the essential features of the methodology.

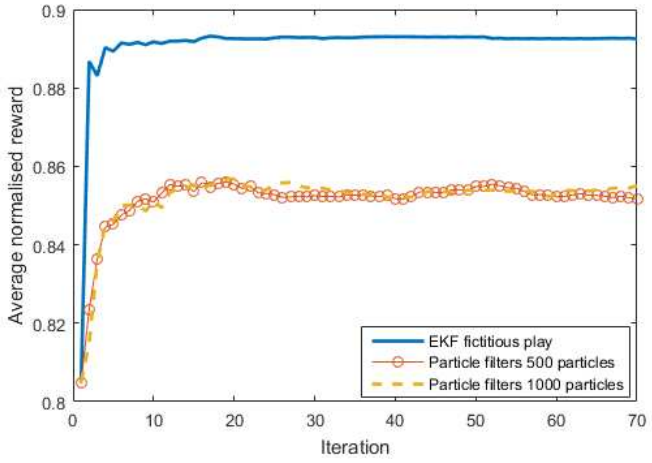

Figure 15. Average reward as a function of time of the three learning algorithms. The x-axis represents the Iteration and the y-axis the average normalised reward.

Appendix A. Proof of Proposition 1

Appendix A.1. Case where players have two available actions

In this Section the proof of Proposition 1 is provided for the case where players have two available actions.

Proof. In the case where players have only 2 available actions Player i's predictions about his opponent's propensity are:

$$\bar{Q}_t = \begin{pmatrix} \hat{Q}_{t-1}[1] \\ \hat{Q}_{t-1}[2] \end{pmatrix} \tag{A.1}$$

$$P_t^- = \begin{pmatrix} P_{t-1}^-[1,1] & P_{t-1}^-[1,2] \\ P_{t-1}^-[2,1] & P_{t-1}^-[2,2] \end{pmatrix} + qI \tag{A.2}$$

without loss of generality we can assume that its opponent in iteration t chooses action 2. Then the update step will be :

$$v_t = z_t - h(\bar{Q}_t) \tag{A.3}$$

since Player i's opponent played action 2 and $h_{s(k)}(Q) = \frac{exp(Q_t[s(k)]/\tau)}{\sum_{\tilde{s}\in S} exp(Q_t[\tilde{s}]/\tau)}$ we can write v_t and $H_t(\bar{Q}_t)$ as:

$$\begin{aligned} v_t &= \begin{pmatrix} 0 \\ 1 \end{pmatrix} - \begin{pmatrix} \sigma_{t-1}(1) \\ 1-\sigma_{t-1}(1) \end{pmatrix} \\ &= \begin{pmatrix} -\sigma_{t-1}(1) \\ \sigma_{t-1}(1) \end{pmatrix} \end{aligned}$$

$$H_t(\bar{Q}_t) = \begin{pmatrix} a_t & -a_t \\ -a_t & a_t \end{pmatrix} \tag{A.4}$$

where a_t is defined $a_t = \sigma_{t-1}(1)\sigma_{t-1}(2)$. The estimation of $S_t = H(\bar{Q}_t)P_t^- H^T(\bar{Q}_t) + Z$ will be:

$$S_t = a^2 \begin{pmatrix} b & -b \\ -b & b \end{pmatrix} + Z \tag{A.5}$$

where $b = P_t^-[1,1] + P_t^-[2,2] - 2P_t^-[1,2]$. The Kalaman gain, $K_t = P_t^- H^T(\bar{Q}_t) S_t^{-1}$ can be written as

$$K_t = \frac{1}{2rb + r^2} \begin{pmatrix} P_t^-[1,1] & k \\ k & P_t^-[2,2] \end{pmatrix} \begin{pmatrix} a_t & -a_t \\ -a_t & a_t \end{pmatrix} \begin{pmatrix} b+r & b \\ b & b+r \end{pmatrix} \tag{A.6}$$

up to a multiplicative constant we can write

$$K_1 \sim \begin{pmatrix} c & -c \\ -d & d \end{pmatrix} \tag{A.7}$$

where $c = P_t^-[1,1] - P_t^-[1,2]$ and $d = P_t^-[2,2] - P_t^-[1,2]$. The EKF estimate of the opponent's propensity is:

$$\hat{Q}_t = \begin{pmatrix} \hat{Q}_t[1] \\ \hat{Q}_t[2] \end{pmatrix} = \begin{pmatrix} \bar{Q}_t[1] - 2\sigma(1)\frac{a(b-k)}{4a^2(b-k)+(r+\epsilon)} \\ \bar{Q}_t[2] + 2\sigma(1)\frac{a(b-k)}{4a^2(b-k)+(r+\epsilon)} \end{pmatrix} \tag{A.8}$$

Based on the above we observe that $\hat{Q}_t(1) < \hat{Q}_{t-1}(1)$ and $\hat{Q}_t(2) > \hat{Q}_{t-1}(2)$. □

Appendix A.2. Case where players have more than two available actions

In this Section the proof of Proposition 1 is provided for the case where players have more than two available actions.

Proof. In the case where there are more than 2 available actions, when $1/t << 1$, then the covariance matrix $S_t \simeq H(\bar{Q}_t)P_t^- H^T(\bar{Q}_t)$ and then $K_t \simeq H(\bar{Q}_t)$. From the definition of $H(\bar{Q}_t)$ we know that it is invertible, with positive diagonal elements and negative off diagonal elements. In particular we can write :

$$H(\bar{Q}_t)[i,i] = \sum_{j \in S/i} \sigma_{t-1}(s^i)\sigma_{t-1}(s^j)$$

$$H(\bar{Q}_t)[i,j] = -\sigma_{t-1}(s^i)\sigma_{t-1}(s^j).$$

Suppose that action s^j is played from Player i's opponent then the update of $\hat{Q}_t[j]$ is the following:

$$\hat{Q}_t[j] = \bar{Q}_t + H(\bar{Q}_t)[j,\cdot]y.$$

The only positive element of y is $y[j]$. The multiplication of $H(\bar{Q}_t)[j,\cdot]$ and y is the sum of $\mathbb{I}$ positive coefficients and therefore the value of $\hat{Q}_t[j]$ will be increased. In order to complete the proof we should show that $\Delta[j] > \Delta[i] \forall i \in S/i$, where $\Delta[\tilde{i}] = \hat{Q}_t[\tilde{i}] - \hat{Q}_{t-1}[\tilde{i}], \tilde{i} \in S$. For simplicity of notation for the rest of the proof we will write $H[i,j]$ instead of $H(\bar{Q}_t)[i,j]$ and $\sigma(i)$ instead of $\sigma_{t-1}(s^i)$.

$$\begin{aligned} \Delta[i] = \; & H[i,\cdot]y \\ = \; & H[i,1]\sigma(1) + H[i,2]\sigma(2) + \ldots + \\ & H[i,i-1]\sigma(i-1) - H[i,i]\sigma(i) + \\ & H[i,i+1]\sigma[i+1] + \ldots + \\ & H[i,j-1]\sigma(j-1) - H[i,j](1-\sigma(j)) + \\ & H[i,j+1]\sigma(j+1) + \ldots + H[i,\mathcal{I}]\sigma(\mathcal{I}) \end{aligned} \tag{A.9}$$

$$\begin{aligned} \Delta[j] = \; & H[j,\cdot]y \\ = \; & H[j,1]\sigma(1) + H[j,2]\sigma(2) + \ldots + \\ & H[j,j](1-\sigma(j)) + \ldots + H[i,\mathcal{I}]\sigma(\mathcal{I}) \end{aligned} \tag{A.10}$$

$$\begin{aligned} \Delta[i] - \Delta[j] = \; & \sigma(1)(H[i,1] - H[j,1]) + \ldots + \\ & \sigma(i-1)(H[i,i-1] - H[j,i-1]) - \\ & \sigma(i)(H[i,i] + H[j,i]) + \\ & \sigma(i+1)(H[i,i+1] - H[i,i+1]) + \ldots + \\ & \sigma(j-1)(H[i,j-1] - H[j,j-1]) - \\ & (1-\sigma(j))(H[[i,j] + H[j,j]]) + \\ & \sigma(j+1)(H[i,j+1] - H[j,j+1]) + \ldots + \\ & \sigma(\mathcal{I})(H[i,\mathcal{I}] - H[j,\mathcal{I}]) \end{aligned} \tag{A.11}$$

If we substitute $H[\cdot,\cdot]$ with its equivalent then we can write $\Delta[i]-\Delta[j]$ as:

$$\begin{aligned}\Delta[i]-\Delta[j]= \quad & (\sigma(1))^2(\sigma(i)-\sigma(j))+\ldots+\\ & (\sigma(i-1))^2(\sigma(i)-\sigma(j))-\\ & (\sigma(i))^2((\textstyle\sum_{\tilde{j}\in S/i}\sigma(\tilde{j}))+\sigma(j))+\\ & (\sigma(i+1))^2(\sigma(i)-\sigma(j))+\ldots+\\ & (\sigma(j-1))^2(\sigma(i)-\sigma(j))-\\ & (1-\sigma(j))\sigma(j)((\textstyle\sum_{\tilde{j}\in S/j}\sigma(\tilde{j}))+\sigma(i))+\\ & (\sigma(j+1))^2(\sigma(i)-\sigma(j))+\ldots+\\ & (\sigma(\mathcal{I}))^2(\sigma(i)-\sigma(j))\end{aligned} \tag{A.12}$$

solving the inequality $\Delta[i]-\Delta[j]<0$ we obtain:

$$\begin{aligned}&\Delta[i]-\Delta[j]<0\Leftrightarrow\\ &(\sigma(i)-\sigma(j))((\textstyle\sum_{\tilde{j}\in S/\{i,j\}}(\sigma(\tilde{j})^2)))<\\ &((\sigma(i))^2((\textstyle\sum_{\tilde{j}\in S/i}\sigma(\tilde{j}))+\sigma(j))+\\ &(1-\sigma(j))(\sigma(j))((\textstyle\sum_{\tilde{j}\in S/j}\sigma(\tilde{j}))+\sigma(i)))\end{aligned} \tag{A.13}$$

In the case where $\sigma(i)<\sigma(j)$ the inequality is satisfied always because the left hand side of the inequality is always negative and the right hand side is always positive. In the case where $\sigma(i)>\sigma(j)$ the inequality (A.13) can be proven by contradiction that (A.13) is satisfied $\forall i, i\neq j$. Therefore we assume that:

$$\begin{aligned}&\Delta[i]-\Delta[j]\geq 0\Leftrightarrow\\ &(\sigma(i)-\sigma(j))((\textstyle\sum_{\tilde{j}\in S/\{i,j\}}(\sigma(\tilde{j})^2)))\geq\\ &((\sigma(i))^2+(1-\sigma(j))(\sigma(j)))((\textstyle\sum_{\tilde{j}\in S/i}\sigma(\tilde{j}))+\sigma(j))\end{aligned} \tag{A.14}$$

Since $((\sum_{\tilde{j}\in S/\{i,j\}}(\sigma(\tilde{j})^2)))<((\sum_{\tilde{j}\in S/i}\sigma(\tilde{j}))+\sigma(j))$ in order to complete the proof, we only need to show that:

$$\begin{aligned}\sigma(i)-\sigma(j)\geq \quad & (\sigma(i))^2+(1-\sigma(j))\sigma(j)\\ \sigma(i)-(\sigma(i))^2\geq \quad & 2\sigma(j)-(\sigma(j))^2\\ \sigma(i)(1-\sigma(i))\geq \quad & \sigma(j)(2-\sigma(j))\end{aligned} \tag{A.15}$$

Inequality (A.15) will be satisfied if the following inequality holds:

$$(1-\sigma(i))\geq \quad \tfrac{\sigma(j)}{\sigma(i)}(2-\sigma(j)) \tag{A.16}$$

Inequality (A.16) will be satisfied if

$$\begin{aligned}(1-\sigma(i))\geq \quad & (2-\sigma(j))\\ \sigma(j)-\sigma(i)\geq 1 \quad & \end{aligned} \tag{A.17}$$

since $\sigma(i)>\sigma(j)$, (A.17), is false $\forall i$ and thus by contradiction $\Delta[i]-\Delta[j]<0$, which completes the proof. □

Appendix B. Proof of Proposition 2

In this Section the proof of Proposition 2 is provided.

Proof. Similarly to the proof of Proposition 1, we consider only one opponent and a game with more than one pure Nash equilibria. We assume that a joint action $s=(s^1(j'),s^2(j))$ is played which is not a Nash equilibrium. Since players use EKF fictitious play because of Proposition 1 t,hey will eventually change their action to $\tilde{s}=(s^1(\tilde{j}'],s^2(\tilde{j}))$ which will be the best response to action $s^2(j)$ and $s^1(j')$ for players 1 and 2 respectively. But if this change is simultaneous then their is no guarantee that the resulting joint action $\tilde{s}$ will increase the expected reward of the players. We will show that with high probability the two players will not change their actions simultaneously infinitely often.

Without loss of generality we assume that at time t Player 2 changes its action from $s^2(j)$ to $s^2(\tilde{j})$. We want to show that the probability that Player 1 will change its action to $s^1(\tilde{j}')$ with probability less than 1. Player 1 will change its action if it thinks that Player 2 will play action $s^2(\tilde{j})$ with high probability such that its utility will be maximised

when it plays action $s^1(\tilde{j}')$. Therefore Player 1 will change its action to $s^1(\tilde{j}')$ if $\sigma^2(s^2(j)) > \lambda$, where $\sigma^2(\cdot)$ is the estimation of its Player 1 about Player 2's strategy. Thus we want to show that:

$$\begin{aligned}
&p(\sigma^2 > \lambda) < 1 \Leftrightarrow \\
&p(\tfrac{\exp(\hat{Q}_{t-1}[j])}{\sum_{j''\in S^2}\exp(\hat{Q}_{t-1}[j''])} > \lambda) < 1 \Leftrightarrow \\
&p(\hat{Q}_{t-1}[j] > \ln\tfrac{\lambda}{1-\lambda} + \textstyle\sum_{j''\in S^2/j}\exp(\hat{Q}_{t-1}[j''])) < 1 \Leftrightarrow \\
&p(\hat{Q}_{t-2}[j] + (K_{t-1}y_{t-1})[j] > \dots \\
&\quad \ln\tfrac{\lambda}{1-\lambda} + \textstyle\sum_{j''\in S^2/j}\exp(\hat{Q}_{t-1}[j''])) < 1
\end{aligned} \tag{B.1}$$

we can expand $K_{t-1}y_{t-1}[j]$ as follows

$$\begin{aligned}
(K_{t-1}y_{t-1})[j] = &\quad ((P_{t-2} + (\tilde{\xi} + \epsilon)I)HS^{-1}y_{t-1})[j] \\
= &\quad (P_{t-2}HS^{-1}y_{t-1})[j] + ((\tilde{\xi} + \epsilon)HS^{-1}y_{t-1})[j]
\end{aligned} \tag{B.2}$$

If we substitute $K_{t-1}y_{t-1}[j]$ in (B.1) with its equivalent, then we obtain the following inequality:

$$\begin{aligned}
&p(\hat{Q}_{t-2}[j] + (P_{t-2}HS^{-1}y_{t-1})[j] + ((\tilde{\xi} + \epsilon)HS^{-1}y_{t-1})[j] > \dots \\
&\quad \ln\tfrac{\lambda}{1-\lambda} + \textstyle\sum_{j''\in S^2/j}\exp(\hat{Q}_{t-1}[j''])) < 1 \Leftrightarrow \\
&p(((\tilde{\xi} + \epsilon)HS^{-1}y_{t-1})[j] > -\hat{Q}_{t-2}[j] - (P_{t-2}HS^{-1}y_{t-1})[j] \dots \\
&\quad \ln\tfrac{\lambda}{1-\lambda} + \textstyle\sum_{j''\in S^2/j}\exp(\hat{Q}_{t-1}[j''])) < 1 \Leftrightarrow \\
&p(\epsilon > C) < 1
\end{aligned} \tag{B.3}$$

where C is defined as:

$$\begin{aligned}
C = &\quad \tfrac{\ln\frac{\lambda}{1-\lambda}}{(HS^{-1}y_{t-1})[j]} + \\
&\quad \tfrac{\sum_{j''\in S^2/j}\exp(\hat{Q}_{t-1}[j''])}{(HS^{-1}y_{t-1})[j]} - \\
&\quad \tfrac{-\hat{Q}_{t-2}[j] - (P_{t-2}HS^{-1}y_{t-1})[j]}{(HS^{-1}y_{t-1})[j]} - \\
&\quad \tfrac{q(HS^{-1}y_{t-1})[j]}{(HS^{-1}y_{t-1})[j]}
\end{aligned} \tag{B.4}$$

Inequality (B.3) is always satisfied since ϵ is a Gaussian random variable. To conclude the proof we define χ_t as the event that both players change their action at time t simultaneously, and assume that the two players have changed their actions simultaneously at the iterations $t_1, t_2, \dots, t_T$, then the probability that they will also change their action simultaneously at time t_{T+1}, $P(\chi_{t_1}, \chi_{t_2}, \dots, \chi_{t_T}, \chi_{t_T+1})$ is almost zero for large but finite T. □

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

[2] Ayken, T., Imura, J.-i., 2012. Asynchronous distributed optimization of smart grid. In: SICE Annual Conference (SICE), 2012 Proceedings of. IEEE, pp. 2098–2102.

[3] Bauso, D., Giarre, L., Pesenti, R., 2006. Mechanism design for optimal consensus problems. In: Decision and Control, 2006 45th IEEE Conference on. pp. 3381–3386.

[4] Bertsekas, D. P., 1982. Distributed dynamic programming. Automatic Control, IEEE Transactions on 27 (3), 610–616.

[5] Bishop, C. M., 1995. Neural Networks for Pattern Recognition. Oxford University Press.

[6] Bordini, R. H., Hübner, J. F., 2006. Bdi agent programming in agentspeak using jason. In: Computational logic in multi-agent systems. Springer, pp. 143–164.

[7] Bordini, R. H., Hübner, J. F., et al., 2004. Jason: A java-based agentspeak interpreter used with saci for multi-agent distribution over the net. On-line at http://jason. sourceforge. net.

[8] Botelho, S., Alami, R., 1999. M+: a scheme for multi-robot cooperation through negotiated task allocation and achievement. In: Robotics and Automation, 1999. Proceedings. 1999 IEEE International Conference on. Vol. 2. pp. 1234–1239 vol.2.

[9] Brown, G. W., 1951. Iterative solutions of games by fictitious play. In: Koopmans, T. C. (Ed.), Activity Analysis of Production and Allocation. Wiley, pp. 374–376.

[10] Chapman, A. C., Rogers, A., Jennings, N. R., Leslie, D. S., 2011. A unifying framework for iterative approximate best-response algorithms for distributed constraint optimization problems. Knowledge Engineering Review 26 (4), 411–444.

[11] Chapman, A. C., Rogers, A., Jennings, N. R., Leslie, D. S., 2011. A unifying framework for iterative approximate best-response algorithms for distributed constraint optimization problems. Knowledge Engineering Review 26 (4), 411–444.

[12] Daskalakis, C., Goldberg, P. W., Papadimitriou, C. H., 2006. The complexity of computing a nash equilibrium. In: Proceedings of the thirty-eighth annual ACM symposium on Theory of computing. pp. 71–78.

[13] Evans, J., Krishnamurthy, B., 1998. Helpmate, the trackless robotic courier: A perspective on the development of a commercial autonomous mobile robot. In: Autonomous Robotic Systems. Vol. 236. Springer London, pp. 182–210.

[14] Farinelli, A., Rogers, A., Jennings, N. R., June 2008. Maximising sensor network efficiency through agent-based coordination of sense/sleep schedules. In: Workshop on Energy in Wireless Sensor Networks in conjuction with DCOSS 2008. pp. 43–56.
[15] Fudenberg, D., Levine, D., 1998. The theory of Learning in Games. The MIT Press.
[16] Grewal, M., Andrews, A., 2011. Kalman filtering: theory and practice using MATLAB. Wiley-IEEE press.
[17] Izzo, P., Qu, H., Veres, S. M., 2016. Reducing complexity of autonomous control agents for verifiability. Submitted in European Control Conference (ECC) 2016.
[18] Jazwinski, A., 1970. Stochastic processes and filtering theory. Vol. 63. Academic press.
[19] Kalman, R., et al., 1960. A new approach to linear filtering and prediction problems. Journal of basic Engineering 82 (1), 35–45.
[20] Kho, J., Rogers, A., Jennings, N. R., 2009. Decentralized control of adaptive sampling in wireless sensor networks. ACM Trans. Sen. Netw. 5 (3), 1–35.
[21] Kho, J., Rogers, A., Jennings, N. R., 2009. Decentralized control of adaptive sampling in wireless sensor networks. ACM Transactions on Sensor Networks (TOSN) 5 (3), 19.
[22] Lincoln, N. K., Veres, S. M., 2013. Natural language programming of complex robotic bdi agents. Journal of Intelligent and Robotic Systems 71 (2), 211–230.
[23] Madhavan, R., Fregene, K., Parker, L., 2004. Distributed cooperative outdoor multirobot localization and mapping. Autonomous Robots 17 (1), 23–39.
[24] Makarenko, A., Durrant-Whyte, H., 2004. Decentralized data fusion and control in active sensor networks. In: Pro. 7th International Conference on Information Fusion. Vol. 1. pp. 479–486.
[25] Miyasawa, K., 1961. On the convergence of learning process in a 2x2 non-zero-person game.
[26] Monderer, D., Shapley, L., 1996. Potential games. Games and Economic Behavior 14, 124–143.
[27] Nachbar, J., 1990. Evolutionary' selection dynamics in games: Convergence and limit properties. International Journal of Game Theory 19, 59–89.
[28] Nash, J., 1950. Equilibrium points in n-person games. In: Proceedings of the National Academy of Science, USA. Vol. 36. pp. 48–49.
[29] Parker, L., 1998. Alliance: an architecture for fault tolerant multirobot cooperation. Robotics and Automation, IEEE Transactions on 14 (2), 220–240.
[30] Raffard, R. L., Tomlin, C. J., Boyd, S. P., 2004. Distributed optimization for cooperative agents: Application to formation flight. In: Decision and Control, 2004. CDC. 43rd IEEE Conference on. Vol. 3. IEEE, pp. 2453–2459.
[31] Robinson, J., 1951. An iterative method of solving a game. Annals of Mathematics 54, 296–301.
[32] Semsar-Kazerooni, E., Khorasani, K., 2008. Optimal consensus algorithms for cooperative team of agents subject to partial information. Automatica 44 (11), 2766 – 2777.
[33] Semsar-Kazerooni, E., Khorasani, K., 2009. Multi-agent team cooperation: A game theory approach. Automatica 45 (10), 2205 – 2213.
[34] Simmons, R., Apfelbaum, D., Burgard, W., Fox, D., Moors, M., Thrun, S., Younes, H., 2000. Coordination for multi-robot exploration and mapping. In: AAAI/IAAI. pp. 852–858.
[35] Simmons, R., Apfelbaum, D., Burgard, W., Fox, D., Moors, M., Thrun, S., Younes, H., 2000. Coordination for multi-robot exploration and mapping. In: AAAI/IAAI. pp. 852–858.
[36] Smyrnakis, M., Leslie, D. S., 2010. Dynamic Opponent Modelling in Fictitious Play. The Computer Journal 53, 1344–1359.
[37] Stranjak, A., Dutta, P. S., Ebden, M., Rogers, A., Vytelingum, P., May 2008. A multi-agent simulation system for prediction and scheduling of aero engine overhaul. In: AAMAS '08: Proceedings of the 7th international joint conference on Autonomous agents and multiagent systems. pp. 81–88.
[38] Timofeev, A., Kolushev, F., Bogdanov, A., 1999. Hybrid algorithms of multi-agent control of mobile robots. In: Neural Networks, 1999. IJCNN '99. International Joint Conference on. Vol. 6. pp. 4115–4118.
[39] Tsalatsanis, A., Yalcin, A., Valavanis, K., 2009. Optimized task allocation in cooperative robot teams. In: Control and Automation, 2009. MED '09. 17th Mediterranean Conference on. pp. 270–275.
[40] Voice, T., Vytelingum, P., Ramchurn, S. D., Rogers, A., Jennings, N. R., 2011. Decentralised control of micro-storage in the smart grid. In: AAAI. pp. 1421–1427.
[41] Wolpert, D., Tumer, K., 2004. .A survey of collectives. Springer, Ch. Collectives and the Design of Complex Systems., pp. 1–42.
[42] Wooldridge, M., 2009. An introduction to multiagent systems. John Wiley & Sons.
[43] Young, H. P., 2005. Strategic Learning and Its Limits. Oxford University Press.
[44] Zecchin, A. C., Simpson, A. R., Maier, H. R., Leonard, M., Roberts, A. J., Berrisford, M. J., 2006. Application of two ant colony optimisation algorithms to water distribution system optimisation. Math. Comput. Model. 44 (5), 451–468.
[45] Zhang, P., Sadler, C. M., Lyon, S. A., Martonosi, M., 2004. Hardware design experiences in zebranet. In: Proc. SenSys'04. ACM, pp. 227–238.
[46] Zhang, Y., Schervish, M., Acar, E., 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). pp. 1525 – 1532.
[47] Zlot, R., Stentz, A., Dias, M. B., Thayer, S., 2002. Multi-robot exploration controlled by a market economy. In: Robotics and Automation, 2002. Proceedings. ICRA'02. IEEE International Conference on. Vol. 3.