Learning in unknown reward games: Application to sensor networks

Archie C. Chapman¹, David S. Leslie², Alex Rogers³ & Nicholas R. Jennings³

¹School of Electrical and Information Engineering, University of Sydney, NSW 2006, Australia
²School of Mathematics, University of Bristol, University Walk, Bristol, BS8 1TW, UK
³School of Electronics and Computer Science, University of Southampton, SO17 1BJ, UK
Email: archie.chapman@sydney.edu.au, david.leslie@bristol.ac.uk, {acr,nrj}@ecs.soton.ac.uk

This paper demonstrates a decentralised method for optimisation using game-theoretic multi-agent techniques, applied to a sensor network management problem. Our first major contribution is to show how the marginal contribution utility design is used to construct a unknown-reward potential game formulation of the problem. This formulation exploits the sparse structure of sensor network problems, and allows us to apply a bound to the price of anarchy of the Nash equilibria of the induced game. Furthermore, since the game is a potential game, solutions can be found using multiagent learning techniques. The techniques we derive use Q-learning to estimate an agent's rewards, while an action adaptation process responds to an agent's opponents' behaviour. However, there are many different algorithmic configurations that could be used to solve these games. Thus, our second major contribution is an extensive evaluation of several action adaptation processes. Specifically, we compare six algorithms across a variety of parameter settings to ascertain the quality of the solutions they produce, their speed of convergence, and their robustness to pre-specified parameter choices. Our results show that they each perform similarly across a wide range of parameters. There is, however, a significant effect from moving to a learning policy with sampling probabilities that go to zero too quickly for rewards to be accurately estimated.

Keywords: Potential games, learning in games, distributed optimisation

Received 15 November 2012; revised 28 February 2013; accepted 15 May 2013

1. INTRODUCTION

Designing and deploying a large, distributed system poses many engineering challenges. In particular, in many scenarios, centralised control is not possible, because limits on the system's communication and computational resources prevent a central authority from having sufficient knowledge of the environment and direct communication with all of the components of the system. In response to these constraints, researchers have focused on decentralised control mechanisms for such systems. This paper demonstrates a decentralised method for optimisation using game-theoretic multi-agent techniques, applied to a distributed sensor network management problem. doing so, it brings together several strands of work, on reinforcement learning, compact game representations, equilibrium efficiency analysis and learning in games.

Given this context, consider a wide—area surveillance problem given by an ad hoc wireless sensor network management problem, such as the one described in [1]. The problem is to maximise the efficiency of a sensor network deployed for wide—area surveillance, in which each sensor's daily available battery charge is constrained, so they can only be actively sensing for a limited time each day. For example, if this period is one third of the day, then the

agent has to make a decision on when it chooses to actively sense, and when it should sleep. In order to cover the entire field of observation, the sensors' observation ranges overlap, which means that the usefulness of each sensor's observations is coupled with that of its neighbours'. Hence, the first part of the problem is to coordinate sense/sleep cycles of the sensors so to maximise the expected number of events observed each day. However, these events occur at random, and, at the outset, the mean frequency of events is unknown to the sensors. Thus, the sensors have to search the joint action space in order to learn the mean frequencies of events occurring, while also coordinating their sense/sleep cycles to reduce the likelihood to redundant event observations. In this setting, resource constraints make centralised computation of a policy impossible, due to the extra energy costs of centrally collecting and returning a solution, and because the computation of a solution may require more energy or memory than any one node possesses.

In this paper, we use this example domain to demonstrate the efficacy of several game—theoretic learning algorithms we derive for problems with unknown noisy rewards. Specifically, we cast the sensor network management problem as a *potential game* [2], which is a class of noncooperative games that have gained prominence as a design template for decentralised control in the distributed optimisation and multi-agent systems research communities, and for which there are many distributed iterative algorithms that are guaranteed to converge.

In general, given a global reward function, a potential game is constructed by distributing the system's control variables among a set of agents (or players), and each agent's reward function is derived so that it is aligned with the system-wide goals. That is, an agent's reward increases only if the global reward increases (as in [3] or, more recently, [4]). If the agents' rewards are aligned with the global reward function, then the global reward function is a potential for the game, and the game is a potential game. This, in turn, implies that the (pure) Nash equilibria of the constructed game correspond to the local optima of the potential function and the global reward function. This is a very useful property of potential games, because a Nash equilibrium is an action profile that is robust to unilateral changes in agents' strategies; and as a consequence, the local optima of the potential function are stable in these games. Furthermore, under the common assumption that the global reward function is submodular in the agents' contributions (i.e. each agent has a decreasing marginal contribution), then the ratio of the worst-case Nash equilibrium to the optimum can be bounded. This ratio is known as the *price of anarchy*, and [5] show that in submodular marginal contribution games it is at most 1/2, or in other words, the value of the worst Nash equilibrium solution (local maximum) is within 1/2 of that of the global optimum. This is effectively a bound on the cost of distributing the control of the problem among multiple autonomous agents.

Given this framework for distributing an optimisation problem, we then specify several distributed algorithms for finding a Nash equilibrium. This problem is addressed by the literature on learning in games; the dynamics of learning processes in repeated games is a well investigated branch of game theory (see [6], for example). In particular, the results that are relevant to this work are the guaranteed convergence to Nash equilibrium in potential games of a variety of action adaptation processes, including finite-memory better reply processes [8], adaptive play [9], joint-strategy fictitious play [10], fading-memory regret monitoring [11], and generalised weakened fictitious play [7]; we also include in our investigation regret-matching [23], which converges to the set of correlated equilibria. Thus, a decentralised solution to an optimisation problem can be found by, first, constructing a potential game from the optimisation problem, and then using one of these algorithms to compute an equilibrium.

There is, however, one major shortcoming to these algorithms. As is standard in game theory, there is an assumption that the value of each configuration of variables, or the agents' rewards for different joint strategy profiles, is known from the outset. Although this is a sound assumption in some domains, in the sensor network domain that we are tackling, it is not realistic to assume that the rewards for different variable configurations can be pre–specified. This

is because the system's task is to learn about the phenomena under observation, but the rewards earned by the agents in the system are a function of the phenomena detected, so their rewards cannot be known before they are deployed.

In this paper, we take several algorithms that are known to converge in potential games, use Q-learning to estimate joint action rewards [12], and interleave their action selection with an appropriately defined learning policy that ensures that rewards are accurately learnt. In more detail, we cast the sensor network management problem as a potential game with unknown noisy rewards. These are games in which an agent's payoff for each outcome is assumed to be drawn from a distribution with bounded variance whose mean is consistent with a potential function. The adaptive processes we consider simultaneously perform recursive estimation of reward function means using Qlearning and adaptation to the strategies of others in the game. Our approach to these types of problems gives agents the ability to effectively learn their reward functions, while coordinating on a pure strategy Nash equilibrium.

Furthermore, since the number of joint actions in a game is exponential in the number of agents, estimating a reward for each joint action quickly becomes an intractable problem. For example, in the application described above, if a network of n sensors each have three periods in which to collect observations, the resulting game has 3^n joint actions. A typical technique for avoiding such computational difficulties is to find compact representations of the problem at hand, and in this vein, we consider a commonly used compact graphical representation of games known as graphical normal form [13, 14]. This representation uses a graph to summarise reward dependencies, and can be exponentially more compact than the standard normal form if the agents' interaction structure is sufficiently sparse, as is typically the case in problems with geographic spread, such as sensor networks. This allows us to use efficient learning policies for Q-learning, such that the learning problem facing the agents is significantly reduced.

We draw three main conclusions from our experiments:

- Our results show a surprising uniformity in the performance of the game—theoretic algorithms, with the execption of the regret-matching algorithm, which performs relatively poorly. Thus, any of the remaining algorithms seem to be suitable for use as distributed optimisation routines for problems formulated as potential games with unknown rewards.
- We also find that the algorithms are quite robust to choices in their free parameters, although the effect of overly-large amount of inertia or belief discounting can be detrimental to algorithms using those effects.
- There is, however, a significant effect from moving to a learning policy with sampling probabilities that go to zero too quickly for rewards to be accurately estimated, and we conjecture that this is a direct effect of relying incorrect reward estimates.

We also test the algorithm' sensitivity to changes in the sparsity of the problem, and we see a significant effect from increasing the averge number of neighbours that a node has, but this effect is small and reasonable given the local search nature of the algorithms investigated here.

1.1. Related work

This paper is an empirical accompaniment to [15], which provides theoretical convergence results for the finite memory algorithms evaluated here in games with unknown noisy rewards. As such, several results from [15] and other works are stated here without proof.

There are alternative frameworks which could be used to model the problem addressed in this paper. One is to adopt the distributed constraint optimisation (DCOP) framework by representing the problem with in a factor graph, in which each overlapping region of sensing is treated as a separate factor (e.g. [1]). Such problems can be solved using Max-Sum or a related algorithm. However, when using the graphical normal form game representation, we do not need to explicitly model each factor (as it is, our model has an agent's utility function implicitly composed of marginal contributions to factors). Moreover, in general, it may not be always possible to identify factors separately, as the agents might not be able identify the phenomena (and do not have localisation capacities) in order to associate one sensor's observation with another. In this case, the problem cannot be formulated as a factor graph, and if it cannot admit this representation, the Max-Sum algorithm cannot be applied. On the other hand, if two observations of any phenomena can be associated with a region, then a factor can be estimated. However, this may require additional computational resources.

Another point of difference is that our game-theoretic approaches are guaranteed to converge to a bounded approximate solution. In contrast, Max–Sum has no such convergence guarantees on loopy topologies, such as the factor–graphs typically encountered in sensor network problems. Nonetheless, we also include a Max–Sum implementation based on *Q*–learned rewards as a benchmark algorithm in Section 5.2.

The paper progresses as follows: In the next section we cover the necessary background on non-cooperative games, games with unknown rewards, the graphical normal form representation and potential games derived using the marginal contribution utility design. Section 3 covers the sensor network problem that we are addressing, and shows how to formulate it as a marginal contribution potential game. In Section 4, we describe the algorithms investigated in this paper, beginning with a multi-agent version of Q-learning using the ε -greedy policy, and then considering each of the action adaptation processes listed above, before combining the two. Our experimental evaluation is covered in Section 5, which begins with the design of the experiments and the algorithm configurations examined, considering both the quality of the solutions produced by the algorithms as well as their robustness to different choices of values for their free parameters, and then reports and analyses the results of our simulations. Section 6

summarises the contributions of this paper, and discusses how our results may be extended.

2. PRELIMINARIES

This section covers noncooperative games, defines games with unknown and noisy rewards and games in graphical normal form, and discusses potential games derived via marginal contribution utilities. Throughout, we use $\mathbb{P}(\cdot)$ to denote the probability of an event occurring.

2.1. Noncooperative games

We consider repeated play of a finite noncooperative game $\Gamma = \langle N, \{A_i, r_i\}_{i \in N} \rangle$, where $N = \{1, \dots, n\}$ is a set of agents, A_i is the set of actions of agent i, and $r_i : \times_{i \in N} A_i \to \mathbb{R}$ is i's reward function. Let $A = \times_{i \in N} A_i$ be the set of all joint actions (also called outcomes), and $\mathbf{a} \in A$ be a particular joint action. Agents can choose to play according to a distribution over pure actions, $\sigma_i \in \Delta(A_i)$, known as a *mixed strategy*, where each element $\sigma_i(a_i)$ is the probability i plays a_i . The rewards of the mixed extension of the game are given by the expected value of r_i under the joint mixed strategy $\sigma \in \times_{i \in N} \Delta(A_i)$ over outcomes:

$$r_i(\mathbf{\sigma}) = \sum_{\mathbf{a} \in A} \left(\prod_{j \in N} \sigma_j(a_j) \right) r_i(\mathbf{a}). \tag{1}$$

We use the notation $\mathbf{a} = (a_i, \mathbf{a}_{-i})$, where \mathbf{a}_{-i} is the joint action chosen by all agents other than i and, similarly, $\sigma = (\sigma_i, \sigma_{-i})$ where σ_{-i} is the joint independent lottery. In this paper, we are interested two solutions, namely *Nash* and *correlated equilibrium*.

DEFINITION 2.1. A joint strategy, $\sigma^* \in \times_{i \in N} \Delta(A_i)$, is a Nash equilibrium if it satisfies:

$$r_i(\sigma^*) - r_i(\sigma_i, \sigma_{-i}^*) \ge 0 \quad \forall \ \sigma_i \in \Delta(A_i), \ \forall \ i \in N.$$

If there is no player, i, that is indifferent between σ_i^* and another strategy (i.e. the inequality above is strict $\forall \sigma_i \neq \sigma_i^*$), then σ^* is a strict Nash equilibrium. All strict Nash equilibria are pure.

DEFINITION 2.2. A distribution $\psi \in \Delta(A)$ is a correlated equilibrium if it satisfies:

$$\sum_{\mathbf{a}\in A: a_{i}\neq k} \psi(\mathbf{a}) \left(r_{i}(\mathbf{a}) - r_{i}(k, \mathbf{a}_{-i}) \right) \geq 0 \quad \forall \ k \in A_{i}, \ \forall \ i \in N.$$

Note that every Nash equilibrium is a correlated equilibrium with ψ a product distribution. Approximate δ -Nash and correlated equilibria are defined by replacing the right hand side of the two expressions above with $\delta>0$. A final technical definition is *generic games*: Γ is generic if a small change to any single reward does not change the number or location of the Nash equilibria of Γ . A sufficient condition for Γ to be generic is that a player is never indifferent between its pure actions. An important implication is that all pure Nash equilibria in generic games are strict.

2.2. Games with unknown noisy rewards

We now introduce the model of rewards received in a repeated learning situation such as the one in our sensor network application. This models the realistic scenario that the observed rewards are noisy, and comprise of an expected value equal to the unknown underlying reward function $r_i(\mathbf{a})$ and a zero–mean random perturbation. We call this scenario *unknown noisy rewards*. This situation therefore requires the individuals to estimate their underlying reward functions, while also adapting their strategies in response to the actions of other agents.

DEFINITION 2.3. A game with unknown noisy rewards is a game in which, when the joint action $\mathbf{a} \in A$ is played, agent i receives the reward

$$R_i = r_i(\mathbf{a}) + e_i \tag{2}$$

where $r_i(\mathbf{a})$ is the true expected reward to agent i from joint action $\mathbf{a} \in A$, and e_i is a random variable with expected value 0 and bounded variance.

To avoid unnecessary over–complication in this article, we assume that each realisation of each e_i is independent of all other random variables. Note that a game with unknown noisy rewards is a generalisation of the bandit problem discussed by [12], and we shall use similar reinforcement learning strategies to estimate the values of $r_i(\cdot)$.

2.3. Games in graphical normal form representation

The game representation above is known as *standard normal form*. One drawback of this representation is that the joint action space A grows exponentially with the number of agents. Moreover, in games with noisy rewards, the estimation problem becomes impractical because the number of joint actions to sample is so large. However, in systems with an inherent structure, such as those with a natural spatial structure in which interaction only directly occurs between geographically close individuals, agents should only need to consider the actions of their neighbours. We now introduce a compact representation for games, which we will later show can be exploited to improve the agents' learning rates.

The representation is called *graphical normal form* (GNF), and it is used to represent games in which some agents' rewards are independent of others' strategies [13]. In this form, the nodes of a graph correspond to the set of agents, while edges connect an agent to the others with which it shares a reward dependency, called its neighbours. The neighbourhood of i is the smallest set v_i of players such that agent i's reward is entirely determined by a_i and $\{a_j: j \in v_i\}$. We say an undirected reward dependency exists between i and $j(\neq i)$ if either $j \in v_i$ or $i \in v_j$.

DEFINITION 2.4. A game in GNF comprises a set of agents located on the nodes of a graph. An agent is connected to those with which it shares an undirected reward dependency, which includes its set of neighbours $V_i \subseteq N$. Its reward function, $r_i(\mathbf{a}_{i,V_i})$, is then given by an array

indexed by tuples from the set $\times_{j \in \{i,v_i\}} |A_j|$. Games in GNF with unknown noisy rewards are defined similarly, with the difference being that when the joint action $\mathbf{a} \in A$ is played, agent i receives the reward

$$R_i = r_i(a_i, \mathbf{a}_{\mathbf{v}_i}) + e_i, \tag{3}$$

where $r_i(a_i, \mathbf{a}_{v_i})$ is the true expected reward to agent i for the joint action (a_i, \mathbf{a}_{v_i}) , and e_i is a random variable with zero mean and bounded variance.

Note that in GNF, $r_i(\mathbf{a})$ depends only on a_i and \mathbf{a}_{v_i} , where \mathbf{a}_{v_i} is the joint action of all the neighbours of i. Subsequently, we write r_i as a function of the joint actions of i and its neighbours; that is, $r_i(\mathbf{a}_{i,v_i})$. Also, note that GNF is *fully expressive*, as any game in standard normal form can be represented in GNF with a complete graph.

2.4. Potential games and marginal contribution utilities

We wish to use game—theoretic techniques to solve a distributed optimisation problem, and in this section we introduce a class of games that make this possible.

Assume that $\phi(\mathbf{a})$ is a function that represents the systems' global objectives. If an increase in a player's reward improves $\phi(\mathbf{a})$ by the same amount, then we say that is *aligned* with $\phi(\mathbf{a})$. This condition is formalised by the following class of games [2]:

DEFINITION 2.5. A function $\phi : A \to \mathbb{R}$ is a **potential** for a game if $\forall i \in N, \forall a_i, \tilde{a}_i \in A_i$, and $\forall \mathbf{a}_{-i} \in A_{-i}$:

$$\phi(a_i, \mathbf{a}_{-i}) - \phi(\tilde{a}_i, \mathbf{a}_{-i}) = r_i(a_i, \mathbf{a}_{-i}) - r_i(\tilde{a}_i, \mathbf{a}_{-i}), \tag{4}$$

and a game that admits such a function is a potential game.

We say utility functions are aligned with a global reward if the reward function is a potential for the game induced by the utility design.

LEMMA 2.1. All games that admit a potential $\phi(\mathbf{a})$ possess a pure Nash equilibrium; all local maxima of $\phi(\mathbf{a})$ are pure Nash equilibria.

The existence and characterisation of pure Nash equilibria are a consequence of Lemma 2.1 of [2].

The most straightforward way to align the agents' rewards with the global reward function is to set them all equal to this function, resulting in a *team game* (a.k.a. *common interest game*). A second, more sophisticated, way to do this is to set each agent's reward equal to the marginal contribution it makes to the global reward. This often reduces the coupling between different agents' reward functions when the agents' interactions are sparse, as is the case for games compactly represented in GNF, which facilitates significant reductions in the communication and computation requirements facing each agent.

In more detail, the sensor network management problem that we address here is an example of an allocation problem, which is characterised by a global reward that is a *set function*, $\phi: 2^{|N|} \to \mathbb{R}$. Computing an agent's marginal contribution to ϕ involves computing $\phi(\emptyset, \mathbf{a}_{-i})$, which is the

global reward, *sans* i, earned by the team of agents. Agent i's marginal contribution to $\phi(\mathbf{a})$ is then computed by taking the difference of the reward if i contributes its component a_i of \mathbf{a} and the reward if i adopts a *null* action \emptyset (i.e. as if i was removed from the system):

$$r_i(a_i, \mathbf{a}_{-i}) = \phi(a_i, \mathbf{a}_{-i}) - \phi(\mathbf{0}, \mathbf{a}_{-i}). \tag{5}$$

Any change in *i*'s reward due to a unilateral change in its action now results in a equal change in the global reward:

$$r_i(a_i, \mathbf{a}_{-i}) - r_i(\tilde{a}_i, \mathbf{a}_{-i})$$

$$= \phi(a_i, \mathbf{a}_{-i}) - \phi(\mathbf{0}, \mathbf{a}_{-i}) - (\phi(\tilde{a}_i, \mathbf{a}_{-i}) - \phi(\mathbf{0}, \mathbf{a}_{-i}))$$

$$= \phi(a_i, \mathbf{a}_{-i}) - \phi(\tilde{a}_i, \mathbf{a}_{-i}).$$

Now, although the computation and communication benefits of the marginal contribution utility design may not be apparent in the general case above, they will become clear once we introduce our sensor network model (in Section 3).

Furthermore, using marginal contributions to a set function allows us to apply a bound to the game's price of anarchy, which is the ratio of the worst Nash equilibrium to the global optimum [5]. In order to do this, however, we must assume some additional conditions on the problem. Specifically, if the potential function is a set function that is non–decreasing and submodular, then the price of anarchy of the locally optimal Nash equilibrium solutions can be bound.

DEFINITION 2.6. A set function $f: 2^{|Z|} \to \mathbb{R}$ in submodular if $f(X) + f(Y) \ge f(X \cup Y) + f(X \cap Y)$, $\forall X, Y \subseteq Z$, and is non-decreasing if $f(X) \le f(Y)$, $\forall X \subseteq Y \subseteq Z$.

A simple class of submodular functions are the *linear* functions: $f(X) = \sum_{i \in I} w^i$, for some set of weights $w^i \in \mathbb{R}$ (n.b. if all $w^i \geq 0$ then $f(\cdot)$ is also non-decreasing). Monotone submodular functions also arise in many combinatorial settings, such as coverage functions and cut functions in graphs and hypergraphs, and other setting where returns to resource allocations are naturally decreasing, such as entropy functions in estimation and monitoring problems and in many typical economic scenarios with decreasing marginal returns to effort or investment. In the next section we show that our sensor network problem is non–deceasing submodular.

The properties of non-deceasing submodularity lead to the following result.

THEOREM 2.1. In marginal contribution potential games with non–decreasing submodular task reward functions, the ratio of the worst–case Nash equilibrium to the optimum is bounded by 1/2.

This result is an application of the results of [16], as shown for marginal contribution potential games by [5].

3. SENSOR NETWORK MANAGEMENT MODEL

The problem we consider in this paper is that of maximising the efficiency of a sensor network deployed for wide area surveillance by coordinating the sense/sleep schedules of power constrained energy-harvesting sensor nodes. Specifically, acoustic sensors, which can be used to detect foot or vehicle traffic, are deployed in an urban setting to monitor nearby traffic. The sensors run on energy harvested from the environment; That is, they operate in an "energyneutral" mode, such that their energy use is equal to their generation [1, 17]. Since the signal-processing required to detect events is typically the most energy intensive activity, the sensors cannot be permanently powered. Rather, they must adopt a duty cycle and sensing schedule that maintains energy-neutral operation. For example, if the length of time that a sensor operates for is one third of the day, then it has to decide on which third of the day it senses, and in which periods it sleeps. The sensors are assumed to be placed randomly, so in order to cover the entire field of observation, they are dispersed densely enough to ensure that nearby sensors' observation ranges overlap. As such, the usefulness of each sensor's observations is coupled with that of other sensors covering a common section of road. An example of the simulation domain is given in Figure 1, which shows the sensors' locations and ranges and the underlying road network on which traffic flows.

This sensor network coverage optimisation problem can be divided into two parts. The first is to coordinate sense/sleep cycles of the sensors so as to maximise the expected number of events observed each day. However, these events occur at random, and, at the outset, the mean frequency of events is unknown to the sensors — below, we show that this makes the sensors' rewards unknown and noisy. The second part of the problem, then, is to estimate the payoffs for different configurations of sensor cycles (which are a function of the unknown mean frequencies of events in the different regions under surveillance and the sleep/sense cycles of the sensors). To do this, the sensors have to learn their payoffs while also coordinating their sense/sleep cycles to maximise the number of event observed. The large number of sensor nodes (there may be hundreds in the system) and the constraints on their computation and communication rules out a centralised optimisation method, so a distributed method must be used.

In more detail, at the system—wide level, during any particular day, a set X of traffic events occurs. The simulator generates several hundred potential event locations, and an event occurs at a particular location in a particular period of the day with a fixed probability, which varies across periods of the day, but is fixed from day to day. These correlations have their origins in the flow of traffic through the underlying network. We define the value of a sensor observing an event, $x \in X$ as:

$$V^{x}(\mathbf{a}) = 1 - \beta^{\#^{x}(\mathbf{a})}$$

where $\#^x(\mathbf{a})$ is the number of sensors that observe x (it is observed if it occurs within a sensor's sensing radius when the sensor is on), and $0 < \beta < 1$ is a sensor's falsenegative detection rate. The parameter β acts to differentiate between sensing cycle configurations that result in many redundant observations of the same event, by imposing diminishing contributions to the global reward for each additional observation of an event. In other words, $V^x(\cdot)$ is

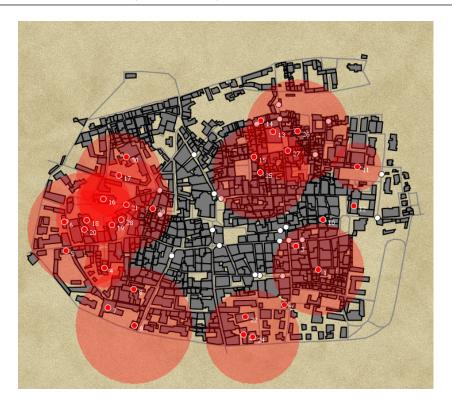


FIGURE 1. The ad hoc wireless sensor network with overlapping sensor regions and the underlying road network. Solid red numbered dots are sensors, opaque red circles indicate the observation regions of active sensors, and white dots represent the vehicles causing events.

submodular. Also note that V^x is 0 if x goes unobserved. The agents time–stamp the events that they sense, and at the end of each day, they compare the lists of time–stamped events to evaluate their action (their choice of sensing period) for that day. An agent's reward for observing an event is the difference in reward it earns for the system for observing or not observing the event; that is, its marginal contribution to the system's performance:

$$R_i^{x}(a_i, \mathbf{a}_{v_i}) = \beta^{\#^{x}(a_i, \mathbf{a}_{v_i}) - 1} - \beta^{\#^{x}(a_i, \mathbf{a}_{v_i})}$$
(6)

Then, each day, its total reward from a sensing cycle is the sum of rewards for all events it observes, $x \in X_i$:

$$R_i(a_i, \mathbf{a}_{\mathbf{v}_i}) = \sum_{x \in X_i} R_i^x(a_i, \mathbf{a}_{\mathbf{v}_i})$$
(7)

See that *i*'s reward depends on the actions of only those agents whose sensing ranges overlap with its own. In this way, only neighbouring agents' payoffs are coupled, and the optimisation problem can be viewed as a game in GNF. This utility derivation results in a marginal contribution potential game, with a potential given by the total system value for all events:

$$V(a) = \sum_{x \in X} 1 - \beta^{\#^{x}(\mathbf{a})}, \tag{8}$$

whose maxima correspond to the Nash equilibria of the associated game. By focusing on high-reward event observations, which are those that are observed by fewer sensors, an agent moves the system towards observing

more events in total. Furthermore, since submodularity is preserved under addition, Equation (8) is submodular, and the worst–case price of anarchy bound of $\frac{1}{2}$ applies.

Given the daily rewards described in Equation (7), an agent estimates the reward it receives in a given time period (e.g. third of the day) and given the actions of its neighbours. Importantly, an agent does not know what portion of its sensing areas overlap with its neighbours, so it cannot use the observations they make when it is asleep to update the Q-values from joint actions other than the one played, because the agent does not know which events it also would have seen. The agent then computes its expected number of event observations for each time period in the next day, and chooses to sense during a time period to using one of the algorithms discussed in the next section.

A simple example is given in Figure 2. In this, two of the three nodes, A and C, are sensing during a period when four events occur x_1 , x_2 , x_3 and x_4 . Note, this is one period during a day, and although not currently on duty, node B will be on and earning rewards during another period during the current day. Letting, for example, $\beta = 0.6$, then the rewards are as follows:

		Event	Marginal
Event	Observed by	reward	contribution
x_1	A	0.40	0.40
x_2	\sim	0	\sim
x_3	A,C	0.64	0.24
<i>x</i> ₄	A,C	0.64	0.24

In more detail, the marginal contribution of A to

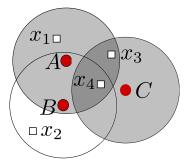


FIGURE 2. Three sensors A, B and C, given by solid red dots, with overlapping sensor regions as indicated by circles, and and four squares representing the events. The two shaded circles are the currently sensing nodes A and C.

sensing event x_1 is $0.6^0 - 0.6^1 = 0.40$, while its marginal contribution to x_3 is $0.6^1 - 0.6^2 = 0.24$; thus, the reward received by A for this cycle is the sum of its marginal contributions, $R_A = 0.40 + 0.24 + 0.24 = 0.88$. For node C it is $R_C = 0.24 + 0.24 = 0.48$, while for node B, we cannot say what its reward is, as we have not specified what events have occurred during its sensing cycle. Note that as more sensors observe an event, the marginal contribution of each decreases, as seen in the marginal contributions of A to x_1 in comparison to x_3 or x_4 . Furthermore, if B were also sensing during this period, the task reward of x_4 would be $1 - 0.6^3 = 0.784$, and the agents' marginal contributions would be $0.6^2 - 0.6^3 = 0.144$. This would lower both A and C's reward for sensing during this period. This demonstrates the submodularity of the reward function, which is justified because the additional information gathered by making another observation of an event is less than that for making the previous.

4. ALGORITHMS

In the previous section, we formulated the sensor network management problem as a potential game with unknown noisy rewards that is repeated over time. In general, the local iterative algorithms employed to solve this type of game operate as follows (see the pseudo code in Algorithm 1). At the start of each day, the agents observe the actions selected by their neighbours, a_{-i} (line 4), and use this information compute their received reward, R_i , as per Equation (2), or Equation (3) for games in GNF (line 5). In our application, this corresponds to calculating Equation (7). Based on this information, the individuals update their estimates of the reward functions (line 6) and update their internal states according to their action adaptation process (line 7). A learning parameter ε is set according to the learning policy in use (line 8), and a new action for the day is selected according to either the unperturbed process (lines 9-10) or randomly sampled in order to explore the joint action space (lines 11-12). For these games, we wish to have behaviour converge to an equilibrium, thereby providing a distributed method of computing (locally) optimal joint strategies with only noisy evaluations of the global reward function.

Algorithm 1 General iterative algorithmic framework

```
1: a_i^0 \leftarrow \text{Random choice from } A_i
 2: for each day do
 3:
         t \leftarrow t + 1
 4:
         Get other agents' last actions a_{-i}^{t-1}
         Compute received reward R_i^{t-1}
 5:
 6:
         Update reward estimate (using Q-learning)
 7:
         Update internal states (according to action adaptation process)
 8:
         Set \varepsilon_i^t (according to learning policy)
 9:
         if (1 - \varepsilon_i^t) > random then
10:
             Follow greedy action of adaptation process
11:
12:
             Randomly sample an action from A_i
13:
         Implement and communicate choice
```

The first step in defining a class of algorithms to do this is to specify a method to accurately learn the unknown reward functions. This is covered in the next subsection, making use of the Q-learning algorithm. After this, we introduce six action adaptation processes that, when interleaved with Q-learning, converge to an equilibrium. We break these into three broad classes of action adaptation process, namely, the finite memory algorithms (adaptive play and finite memory better replies, Section 4.2), infinite joint–strategy memory algorithms (joint–strategy fictitious play, weighted regret monitoring and regret matching, Section 4.3), and infinite independent–strategy memory algorithms, which are also known as the class of generalised weakened fictitious play processes, (Section 4.4). The final subsections of this section then addresses Q-learning variants of these six processes.

4.1. *Q*-learning unknown reward functions

In noisy environments, reinforcement learning is often used to estimate the mean value of a perturbed reward function [12]. In particular, if the agents update their estimates of the expected rewards for joint actions using Q-learning, and select actions using an appropriate ε -greedy learning policy, then the reward function estimates will converge to their true mean values. Here, we estimate rewards as functions of joint actions, while simultaneously reasoning about the action selection of the other agents. The Q-learning scheme we derive here will be used in conjunction with all six of the action adaptation processes considered in the remainder of this section.

In particular, we consider a multi-agent version of Q-learning for single-state problems, in which the agents select a joint action and each receives an individual reward. This algorithm operates by each individual recursively updating an estimate of its value of a joint action **a**. Specifically, after playing action a_i^t , observing actions \mathbf{a}_{-i}^t , and receiving reward R_i^t , each individual i updates estimates Q_i^t using the equation, $\forall \mathbf{a} \in A$:

$$Q_i^{t+1}(\mathbf{a}) = Q_i^t(\mathbf{a}) + \lambda(t)I\{\mathbf{a}^t = \mathbf{a}\} \left(R_i^t - Q_i^t(\mathbf{a})\right)$$
(9)

where the indicator $I\{\mathbf{a}^t = \mathbf{a}\}$ takes value 1 if $\mathbf{a}^t = \mathbf{a}$ and 0 otherwise, and $\lambda(t) \in (0,1)$ is a learning parameter. We

¹This is the JAL approach suggested (without analysis) in the context of fictitious play by [18].

use the following form of $\{\lambda(t)\}_{t\geq 1}$ in the remainder of the paper:

$$\lambda(t) = \left(C_{\lambda} + \#'(\mathbf{a})\right)^{-\rho_{\lambda}} \tag{10}$$

where $C_{\lambda} > 0$ is an arbitrary constant, $\rho_{\lambda} \in (1/2, 1]$ is a learning rate parameter, and $\#^{t}(\mathbf{a})$ is the number of times the joint action \mathbf{a} has been selected up to time t. Given this form of $\{\lambda(t)\}_{t\geq 1}$, if all $Q_{i}(\mathbf{a})$ are updated infinitely often (i.e. all joint actions are played infinitely often), then $Q_{i}^{t}(\mathbf{a}) \to \mathbb{E}[R_{i}^{t} | \mathbf{a}^{t} = \mathbf{a}]$ with probability 1.

The condition that all actions $\mathbf{a} \in A$ are played infinitely often can be met with probability 1 by using a randomised *learning policy*, in which the probability of playing each action is bounded below by a sequence that tends to zero sufficiently slowly as t becomes large. Furthermore, this learning policy can be chosen so that it is greedy in the limit, in that the probability with which it selects maximal reward actions tends to 1 as $t \to \infty$. Such policies are called *greedy in the limit with infinite exploration (GLIE)* [19].

In this paper, we use the commonly–applied GLIE policy ε –greedy. Under this policy, an agent selects a "greedy" action at time t with probability $(1 - \varepsilon_i(t))$ (although note that we have not yet defined what a greedy action should be in this context), and chooses an action at random with probability $\varepsilon_i(t)$.

However, in contrast to single–agent settings, in multi–player games, the choice of joint action is made by the independent choices of more than one agent. As such, for each Q–value to be updated infinitely often, the schedule $\{\varepsilon_i(t)\}_{t\to\infty}$ that each agent's sampling sequence follows must reflect the fact that the agents cannot explicitly coordinate to sample specific joint actions.

LEMMA 4.1. In a game with unknown noisy rewards, if agents select their actions using a learning policy in which, for all $i \in N$, $a_i \in A_i$ and $t \ge 1$,

$$\mathbb{P}(a_i^t = a_i) \ge \varepsilon_i(t), \quad with \quad \varepsilon_i(t) = c_{\varepsilon}t^{-1/|N|},$$

where $c_{\varepsilon} > 0$ is a positive constant, then

$$\lim_{t \to \infty} |Q_i^t(\mathbf{a}) - r_i(\mathbf{a})| = 0 \quad \forall i \in \mathbb{N}, \ \forall \mathbf{a} \in A,$$

with probability 1.2

Directly applying this reasoning may result in a practical learning procedure if |N| is sufficiently small. However, in games with many agents, visiting each joint action infinitely often is an impractical constraint; to achieve sufficiently high exploration rates through independent sampling would require the agents' ε_i sequences to decrease so slowly that in any practical sense the agents will never move into an exploitation phase. To address this limitation, we consider sparse games in GNF, in which each agent interacts directly with only a small number of other agents (as is the case in geographically dispersed sensor networks), such that the

number of reward values each individual estimates can be significantly reduced.

In more detail, for games in GNF, each agent needs to learn only its rewards over it and its neighbours' joint action spaces, given by: $A_{i,v_i} = A_i \times_{j \in v_i} A_j$. For large but sparse games, this is a much more feasible task than estimating the full reward function on A. Each individual i now updates its estimates Q_i^t for all $\mathbf{a}_{i,v_i} \in A_{i,v_i}$ using the equation,:

$$Q_i^{t+1}(\mathbf{a}_{i,\mathbf{v}_i}) = Q_i^t(\mathbf{a}_{i,\mathbf{v}_i}) + \lambda(t)I\{\mathbf{a}_{i,\mathbf{v}_i}^t = \mathbf{a}_{i,\mathbf{v}_i}\} \left(R_i^t - Q_i^t(\mathbf{a}_{i,\mathbf{v}_i})\right). \tag{11}$$

Now the sequence $\{\varepsilon(t)\}_{t\to\infty}$ can be altered to take advantage of the reduced size of each agent's joint action space. while still ensuring that each Q-value is updated infinitely often. Specifically, let i's $neighbourhood\ size$ be the number of neighbours of i plus 1 for i itself. Given this, let J_i be the size of the largest of the neighbourhoods of i or any j in V_i .

LEMMA 4.2. In a game with unknown noisy rewards, if agents select their actions using a policy in which, for all $i \in N$, $a_i \in A_i$ and $t \ge 1$,

$$\mathbb{P}(a_i^t = a_i) \ge \varepsilon_i(t), \quad with \quad \varepsilon_i(t) = c_{\varepsilon} t^{-1/J_i},$$

where $c_{\varepsilon} > 0$ is a positive constant, then $\forall i \in \mathbb{N}, \forall \mathbf{a}_{i,v_i} \in A_{i,v_i}$:

$$\lim_{t\to\infty}|Q_i^t(\mathbf{a}_{i,\mathbf{v}_i})-r_i(\mathbf{a}_{i,\mathbf{v}_i})|=0$$

with probability 1.

We now consider the first of three classes of action adaptation algorithms.

4.2. Finite memory processes

The common feature of all algorithms in the class of finite memory strategy adaptation processes, is that each agent possesses a memory of length m, recalling the previous m actions taken by its opponents, or neighbours v_i for games in GNF, to which the agent responds. We consider two specific subclasses, namely, better–reply processes with inertia and adaptive play processes.

First, we say that a finite memory algorithm has *inertia* if each agent has a strictly positive probability of repeating its current action, $\mathbb{P}(a_i^t = a_i^{t-1}) > 0$.

DEFINITION 4.1 (Better reply processes with inertia, BRI). An agent has a memory of length m and recalls the previous m actions taken by its opponents. At each time step, with probability ξ the agents plays the same action as in the previous time step, $a_i^t = a_i^{t-1}$, while with probability $1 - \xi$ the agent selects an action according to a distribution that puts positive probability on all actions that are better responses to its memory.

Any algorithm that selects from the set of better replies falls into this class of algorithms, including the better response dynamics [20] and the evolutionary–inspired process of [21]. For any potential game with known rewards

²The proofs of this claim and those in Section 4.2 are presented in the companion paper [15], which covers theoretical convergence results for finite memory algorithms in games with unknown noisy rewards.

 Γ , Theorem. 6.2 of [8] states that the better–reply processes with inertia converges almost surely in period-by-period behaviours to a pure Nash equilibrium in Γ .

The second class of algorithms we consider are the adaptive play processes.³

DEFINITION 4.2 (Adaptive play, AP). An agent has a finite memory of length m, and recalls the previous m actions taken by its opponents. At each time step, each agent samples $k \le m$ of the elements of its memory, and plays a best response to the actions in the sample.

For a generic potential game Γ , Theorem 1 of [9] provides a condition that almost surely guarantees that a pure Nash equilibrium is reached: If $k \leq m/(L_{\Gamma}+2)$, then for all time steps beyond $T=kL_{\Gamma}+m$, the probability that the joint memories of all agents consists entirely of one pure Nash equilibrium is strictly positive. Thus, adaptive play converges almost surely in period-by-period behaviours to a pure Nash equilibrium of Γ .

4.3. Infinite joint-strategy memory processes

We now consider three strategy adaptation process with infinite memory, namely, joint-strategy fictitious play, regret-based dynamics and regret matching.

Before defining these algorithms, we must introduce a few terms. Let:

$$z_{\mathbf{a}}^{t} = \frac{1}{t} \sum_{\tau=0}^{t-1} \mathbb{I}\{\mathbf{a}^{\tau} = \mathbf{a}\}\$$

be the empirical frequency of play of joint action \mathbf{a} at time t, and let \mathbf{z}^t be the vector of length |A| containing all of the components $\{z_{\mathbf{a}}^t\}_{\mathbf{a}\in A}$. Denote by $z_{\mathbf{a}_{-i}}^t$ the empirical frequency of a joint action by i's opponents, \mathbf{a}_{-i} , which can be calculated recursively by:

$$z_{\mathbf{a}_{-i}}^{t} = \frac{1}{t} \sum_{\tau=0}^{t-1} \mathbb{I}\{\mathbf{a}_{-i}^{\tau} = \mathbf{a}_{-i}\} = \left(1 - \frac{1}{t}\right) z_{\mathbf{a}_{-i}}^{t-1} + \frac{1}{t} \mathbb{I}\{\mathbf{a}_{-i}^{t-1} = \mathbf{a}_{-i}\}.$$
(12)

These values are interpreted as agent *i*'s *belief* over the joint actions of its opponents, and *i* stores these values in a vector \mathbf{z}_{-i}^t , of length $|\prod_{j \in -i} A_j|$. For a game in GNF, the same values are stored for only an agent's neighbours, \mathbf{v}_i , so that $\mathbf{z}_{\mathbf{v}_i}^t$ is of length $|\prod_{j \in \mathbf{v}_i} A_j|$. Building on this, a variation of the belief update in Equation (12) in which agents have exponentially *fading memory* can be implemented by:

$$z_{\mathbf{a}_{-i}}^{t} = (1 - \gamma)z_{\mathbf{a}_{-i}}^{t-1} + \gamma \mathbb{I}\{\mathbf{a}_{-i}^{t-1} = \mathbf{a}_{-i}\},\tag{13}$$

where $0 < \gamma < 1$ is a parameter controlling the weight placed on recent observations.

The first infinite memory algorithm considered is *joint*–strategy fictitious play, in which each agent recursively constructs a belief from its full set of observations, and plays

a best response to this belief [22, 6]. Specifically, in joint–strategy fictitious play the expected reward to i for an action $a_i \in A_i$ given belief \mathbf{z}_{-i}^t is:

$$r_i(a_i, \mathbf{z}_{-i}^t) = \sum_{\mathbf{a}'_{-i} \in A_{-i}} z_{\mathbf{a}'_{-i}}^t r_i(a_i, \mathbf{a}'_{-i}), \tag{14}$$

where each $i \in N$ updates its \mathbf{z}_{-i}^t according to Equation (12) or Equation (13). Although forms of joint strategy fictitious play using each belief update have been proven to converge to Nash equilibria in potential games [11], here we focus on one specific variant with fading memory and inertia.

DEFINITION 4.3 (Fading–memory joint–strategy fictitious play with inertia, JSFP). Each agent maintains a vector, \mathbf{z}_{-i}^t , containing its fading–memory beliefs over its opponents' joint actions. At each time step, \mathbf{z}_{-i}^t is updated by Equation (13), with γ held constant across all t. The agent then computes the expected reward for its actions by Equation (14). If $\max_{a_i \in A_i} r_i^t (a_i, \mathbf{z}_{-i}^t) = r_i (a_i^{t-1}, \mathbf{z}_{-i}^t)$, then the agent continues to play $a_i^t = a_i^{t-1}$; otherwise, with probability $1 - \xi$ the player chooses an action with uniform probability from the set of best responses to its beliefs, $b_i(\mathbf{z}_{-i}^t)$, and with probability ξ the agent continues to play $a_i^t = a_i^{t-1}$.

For a generic potential game Γ , by Theorem 3.1 of [11], fading memory JSFP with inertia converges almost surely to a pure Nash equilibrium, as long as $0 < \xi < 1$.

The second family of processes are known as $regret-based\ dynamics\ [10]$. These use measures of regret rather than expected utility to evaluate action choices. The regret for an action is constructed by first computing the average difference in reward between taking action a_i at every time step and following the actual sequence of actions employed, and then taking only positive values:

$$v_i^t(a_i) = \max\{r_i(a_i, \mathbf{z}_{-i}^t) - r_i(\mathbf{z}^t), 0\}, \tag{15}$$

where each $r_i(\cdot)$ is computed as in Equation (14). The first term on the righthand side is the expected reward for action a_i given i's current beliefs, while the second term is the average reward i has earned over the actual history of play. If \mathbf{z}_i^t is updated by Equation (12), then the difference is that used in regret matching; if it is by Equation (13), then we have the family of regret–based dynamics. Both regret algorithms then select a new action from the set of actions with (strictly) positive regret. Adding inertia to the agents' action choice then results in the following algorithm:

DEFINITION 4.4 (Fading–memory regret–based dynamics with inertia, FRD). Each agent maintains a vector, \mathbf{z}_{-i}^t , containing its fading–memory beliefs over its opponents' joint actions. At each time step, \mathbf{z}_{-i}^t is updated by Equation (13), with γ held constant across all t, and the agent computes the regret for its actions by Equation (15). Then, with probability ξ the agents plays the same action as in the previous time step, $a_i^t = a_i^{t-1}$, while with probability $1 - \xi$ the agent selects an action according to a distribution that puts positive probability on all actions with positive regret.

³Adaptive play algorithms are also called *fictitious play with finite memory* in [8].

Theorem 4.1 of [10] proves that this algorithm converges almost surely to a Nash equilibrium in generic potential games, assuming $0 < \xi < 1$.

The third algorithm in this class is *regret matching* [23], which we include for completeness (since it is commonly applied) and as a benchmark. Let:

$$\xi_i \ge (|A_i| - 1) \max_{\mathbf{a}, \mathbf{a}'} \{|r_i(\mathbf{a})| - r_i(\mathbf{a}')|\} \quad \forall \ i \in \mathbb{N}$$
 (16)

be an inertial constant. Under regret matching, action choice probabilities are calculated as follows.

DEFINITION 4.5 (Regret matching, RM). At each time step, \mathbf{z}^t is updated by Equation (12), and the agent computes the regret for its actions by Equation (15). Then, the agent chooses an action with probability:

$$\mathbb{P}(a_i^t = a_i') = \begin{cases} \frac{1}{\xi_i} v_{a_i^{t-1}, a_i'}(t), & \text{for all } a_i' \neq a_i^{t-1}; \\ 1 - \frac{1}{\xi_i} \sum_{a_i' \neq a_i^{t-1}} v_{a_i, a_i'}(t) & a_i' = a_i^{t-1}. \end{cases}$$

Note that the choice of ξ ensures that a_i^{t-1} is repeated with positive probability, and that any other action is chosen iff it has positive regret. If all agents playing a generic game use the procedure above, then sequence $\{\mathbf{z}^t\}_{t\geq 1}$ "approaches" the set of correlated equilibria (in the sense of [24]), such that as $t\to\infty$, $\mathbb{P}(\mathbf{z}^t\in\Psi)=1$; in other words, the distribution of the empirical history of play converges to the set of correlated equilibria [23].

4.4. Generalised weakened fictitious play processes

To begin, we describe the belief update used in classical fictitious play process [25], and also in the *joint action learner* of [18]. We then consider the broader class of generalised weakened fictitious play processes analysed by [7].

Let agent *i*'s *historical frequency* of playing a_i , be defined as:

$$\hat{\sigma}_{i,a_i}^t = \frac{1}{t} \sum_{\tau=0}^{t-1} I\{a_i^{\tau} = a_i\}. \tag{17}$$

We write $\hat{\sigma}^t = \{\sigma_{i,a_i}^t\}_{i \in N, a^i \in A^i}$ for the vector of these beliefs, and σ_{-i}^t for the collection of beliefs about all agents other than i. Note that, in contrast to the infinite joint–strategy memory algorithms, each agent's belief is stored independently.

Building on this, [7] define the class of generalised weakened fictitious play (GWFP) processes. These are processes that admit a more general belief–updating process and allow δ –best responses to be played by the agents. Let:

$$\hat{\sigma}_{i,a_i}^t = \alpha^t \sum_{\tau=0}^{t-1} I\{a_i^{\tau} = a_i\}.$$
 (18)

be the generalised belief update rule, with $\alpha^t \to 0$ as $t \to \infty$, subject to the conditions that $\sum_{t \ge 1} \alpha^t = \infty$ and $\sum_{t=1}^{\infty} (\alpha^t)^2 < \infty$.

DEFINITION 4.6 (Generalised weakened fictitious play, GWFP). At each time step, $\hat{\sigma}^t$ is updated by Equation (18), and the agent draws an action according to a lottery that is a δ -best response to its beliefs; that is, according to some mixed strategy $\sigma_i^t \in \{\sigma_i \in \Delta_i : \max_{\tilde{a}_i} r_i(\tilde{a}_i, \hat{\sigma}_{-i}) - r_i(\sigma_i, \hat{\sigma}_{-i}) \leq \delta\}$.

Note that classical fictitious play is the case where $\alpha^t = 1/t$. If $\delta \to 0$ as $t \to \infty$, then the limit set of a GWFP process consists of a connected set of Nash equilibria in potential games (i.e. non–strict but adjacent Nash equilibrium action profiles) and several smaller classes of games [7].

4.5. Interleaving *Q*-learning with action adaptation

Q-learning variants of the algorithms defined in the previous three subsections can be constructed using the ε -greedy learning policy, as follows:

DEFINITION 4.7 (Q-learning variants). At each timestep, each agent acts independently and:

- uniformly samples from A_i with probability $\varepsilon(t)$, or
- selects an action according to the standard algorithm definition with probability $1 \varepsilon(t)$ (e.g. Definition 4.2 for AP, etc.), with actions selected with respect to joint action reward estimates that are updated according to Equation (9) (or Equation (11) for GNF), where $\{\lambda(t)\}_{t\geq 1}$ follows Equation (10) with $C_{\lambda} > 0$ and $\rho_{\lambda} \in (1/2, 1]$.

The six algorithms listed in the preceding subsections are all covered by the results Section 4.1. That is, if the conditions of Lemmas 4.1 or 4.2 are met, then the reward estimates will be accurately learnt.

The Q-learning variants of AP and BRI are proven to converge to a strict Nash equilibrium under additional conditions on the $\varepsilon_i(t)$ schedules that ensure the processes are *strongly ergodic* [15]. Specifically, assuming the standard conditions are satisfied (i.e. on k for AP, and ξ for BRI), if, in addition, $\varepsilon_i(t) = ct^{-1/mJ_i}$, then in generic potential games with unknown noisy rewards, $\lim_{t\to\infty} \mathbb{P}(\mathbf{a}^t \text{ is a Nash equilibrium}) = 1$.

The Q-learning variant of RM converges to a correlated equilibrium in all generic games, a result that is implicit in [23] and made explicit in [15].

The convergence to Nash equilibrium of the *Q*-learning variants of JSFP, FRD and GWFP are not yet proven. However, if they do converge, we know that it must be to a Nash equilibrium, as no other joint actions are stable for these processes. Moreover, since the *Q*-learning results continue to apply to their reward estimates, it is reasonable to expect that their convergence in known-reward games without action selection perturbations to transfer to the unknown-reward case with vanishing perturbations.

5. EXPERIMENTAL EVALUATION

In this section we evaluate the algorithms derived earlier, in the sensor network problem above. We begin with the setup of the simulations and their connection to the model from Section 3, then compare the solution quality across the action adaptation processes (Section 5.2) and learning policies (Section 5.3), conduct a detailed examination of the parameter settings for each individual processes (Section 5.4), and finish with an investigation into the effects of varying the problem density on the algorithms' performance (Section 5.5). For reference, a list of the algorithms and their abbreviations is given in Table 1.

5.1. General features of the experiments

The algorithms are tested in the scenario described in Section 3. Specifically, 50 sensor nodes are randomly spread throughout a surveillance area, over which vehicle activity is simulated. Vehicles travel along roads, which means that the events detected by the sensors are located along roads, and these events occur in much greater numbers than the number of sensors. Each road has a fixed probability distribution over the number of vehicles travelling on it during a given third of the day; learning the mean frequencies of these trips is the goal of the sensor network.

We recorded the ratio of the value of the solution found by an algorithm at each time step to the scenario optimum — that is, the proportion of the optimum, $V(\mathbf{a}^t)/V(\mathbf{a}^*)$, where \mathbf{a}^* is optimal joint action for that scenario — so that we can aggregate our simulation results across scenarios with different payoff levels. For each algorithm configuration, we average this measure over 20 runs each of 50 different scenarios. For Sections 5.2 and 5.3, we also record the worst performance by each algorithm in any problem instance, in order to empirically validate the theoretical worst—case bound of 1/2.

We use the same Q-learning parameters for all algorithms and benchmarks throughout the experiments, with $\rho_{\lambda}=1$ and $c_{\lambda}=0$. However, we use several different learning policies, and several different action adaptation process parameter configurations.

We next describe our results, and the different learning policy and adaptive process parameters that we investigate. These are divided into three sections. The first compares the performance across the algorithms, including a comparison to a Max–Sum based Q–learning algorithm. The second section compares the performance of two of the algorithms across three different learning policies, which satisfy both, one or neither of the convergence conditions derived earlier, respectively. In the third section, we investigate the robustness of the algorithms to the settings of their free parameters. The final section shows how the problem density affects the performance of these algorithms.

5.2. Comparison across algorithm choice

In this section, we compare the performance across the algorithms in order to ascertain if there is any great benefit in using one over the other, or if any are worth ignoring in future deployments. We also compare these algorithms to a Max–sum instantiation, which selects an approximately optimal action at each time step, and which we treat as a upper bound on their expected performance; it is, however, an order of magnitude more computationally intensive to run than our game theoretic algorithms, as is discussed below.

5.2.1. Algorithm configurations

The specific parameters used by our algorithms are given in the legends on each of the plots, and are:

- BRI with inertia using memory length m = 3 and inertial constant $\xi = 0.3$;
- AP with memory length m = 5 and sample size k = 1;
- JSFP with discounting factor $\gamma = 0.2$ and inertial constant $\xi = 0.3$;
- FRD with discounting factor $\gamma = 0.2$ and inertial constant $\xi = 0.3$;
- GWFP with learning rate parameter $\rho_{\alpha} = 0.8$.

Apart from AP, these values were chosen as they represent the middle of regions of relatively equal performance across the algorithms' parameters (investigated in more detail in the next section), and as such can be considered reasonable *a priori* parameter choices. For RM, ξ is set to satisfy Equation (16), so it has no free parameters.

The learning policy schedules used for BRI and AP is a ε -greedy policy that satisfies all of the Nash equilibrium convergence criteria of [15], as listed in Section 4.5. The specific the sampling probabilities used were: $\varepsilon(t) = \frac{1}{8} t^{-1/j_{im}}$ for all $i \in N$.

For all versions of JSFP, FRD, RM and GWFP, the sampling probabilities used were $\varepsilon(t) = 1/8 \ t^{-1/J_i}$ for all $i \in N$, which is sufficient for the Q-learnt reward estimates to converge. Since the proofs for these algorithms' convergence is not based on strong ergodicity arguments, we do not require the "slower" decay of sampling probabilities used for BRI and AP.

5.2.2. Benchmark algorithm: Max-Sum Q-learning

As a benchmark for this first set of results, we compare the algorithms to an implementation of the Max-Sum algorithm that uses the Q-learnt reward estimates. This algorithm has been applied to numerous distributed optimisation settings, and most relevant here is the sensor network application of [1]. In more detail, this benchmark algorithm operates according to the pseudo code in Algorithm 1, with the approximate optimal joint action computed using the Max-Sum algorithm (line 6). Each agent then uses its component of this joint action as its greedy action with the εgreedy learning policy (line 10). Max-Sum operates by passing messages between variables (agents' schedules) and factors (reward functions for event observations), which, in our instantiation, are each of size $\prod_{i \in \{|v_i| \cap i\}} A_i$. (For reference, the problems in this section have an average $|v_i|$ of approximately 2.4, so a node has on average 2.4 neighbours). Max-Sum is optimal and efficient on problems with treestructured representations, and is known to perform well on

⁴The optimum **a*** was computed using simulated annealing, which is prohibitively slow for this setting.

Abbreviation	Algorithm description	Parameters
BRI	Better reply process with inertia	Finite memory length m ; inertial constant ξ
AP	Adaptive play	Finite memory length m ; sample length k
JSFP	Fading-memory joint-strategy fictitious play with inertia	Discounting factor γ ; inertial constant ξ
FRD	Fading-memory regret-based dynamics with inertia	Discounting factor γ ; inertial constant ξ
RM	Regret matching	No free parameters
GWFP	Generalised weakened fictitious play	Learning rate parameter ρ_{α}
MS-QL	Max–Sum <i>Q</i> –learner (benchmark)	No free parameters

TABLE 1. List of algorithm evaluated in Section 5.

loopy topologies. A discussion of the data–structures used in the operation of max-sum is beyond this paper (and can be found in [1]), but it is important to note that the rewards that are estimated by the agents' Q–learning processes are not the marginal rewards for observing an event (as in Equation (6)), but the *average* reward per observing agent. That is, an agent's reward for observing an event is:

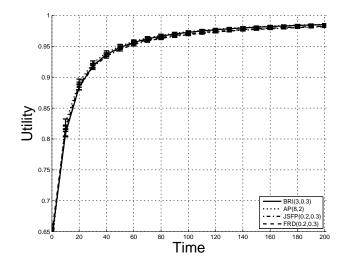
$$R_{i,ms}^{x}(a_{i},\mathbf{a}_{v_{i}}) = \frac{1 - \beta^{\#^{x}(a_{i},\mathbf{a}_{v_{i}})}}{\#^{x}(a_{i},\mathbf{a}_{v_{i}})}$$
(19)

with total reward $R_{i,ms}(a_i, \mathbf{a}_{\mathbf{v}_i}) = \sum_{x \in X_i} R_{i,ms}^x(a_i, \mathbf{a}_{\mathbf{v}_i})$. Thus, the agents' rewards sum to the global value:

$$\begin{split} \sum_{i=1}^{n} R_{i,ms}(a_i, \mathbf{a}_{v_i}) &= \sum_{i=1}^{n} \sum_{x \in X_i} \frac{1 - \beta^{\#^x(a_i, \mathbf{a}_{v_i})}}{\#^x(a_i, \mathbf{a}_{v_i})} \\ &= \sum_{x \in X} 1 - \beta^{\#^x(\mathbf{a})} = V(\mathbf{a}) \end{split}$$

which is approximately computed by the Max–Sum algorithm.^{5,6} In our implementation, the duration of a simulated run is partitioned into two portions:⁷ (i) During the first portion, the agents only randomly and independently sample, in order to learn something of their rewards for joint actions (i.e. ε is set to 1 and the Max–Sum algorithm is not used); (ii) In the second portion, at each time step, the Max–Sum algorithm is used to find the greedy action for the ε –greedy learning policy (line 6 of Algorithm 1), while ε values themselves decrease according $\varepsilon(t) = 1/8 t^{-1/J_i}$ for all $i \in N$ (line 8), which is the schedule used by JSFP, etc. The agents' employ Q–learning to update their reward estimates throughout the entirety of a run. This partition is done

⁷This technique is called ε–first in the multi–armed bandit literature.



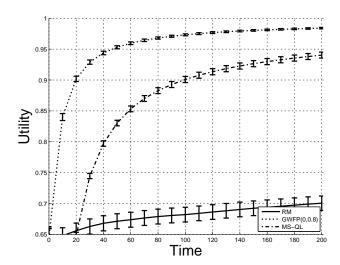


FIGURE 3. Results of the sensor network simulations, plotting the average ratio of the reward earned to the global optimum for each of the algorithms.

because the Max–Sum algorithm itself takes a considerable amount of time and computational resources to run (c.f. the greedy action computation of our game–theoretic algorithms at each time step), and there is very little benefit in using it on inaccurate reward estimates. We set the first portion equal to the first 20 time–steps of the run, and the second portion to the remainder.

⁵The key conceptual difference in the operation of the game–theoretic algorithms discussed in this paper and Max–Sum is that Max–Sum directly computes the optimal value and the argmax variable configuration, whereas the game–theoretic algorithms use the gradient of rewards (i.e. marginal rewards) to find a local optimum.

⁶Note that the problem representation to which Max–Sum is applied to here differs from that of [1]: In that work, each region that is sensed by different combinations of neighbouring agents is treated as a separate factor, and, as such, its rewards are estimated separately. To do this requires a pre–processing stage to identify the overlapping regions, which is not needed by our game–theoretic algorithms. As a consequence, all of the algorithms here estimate a reward function of the same size (i.e. the local joint action spaces under both representations are the same). However, this does adversely affect the operation Max–Sum algorithm, as size of the messages it passes are often considerably larger here compared to those in [1]; that is, the dimension of the message is always $|\mathbf{v}_i|+1$, rather than number of sensors that can see a particular segment, which is at most $|\mathbf{v}_i|+1$, and typically less.

Algorithm	Mean	2 S.E	Minimum
BRI	0.9798	(0.0013)	0.7573
AP	0.9813	(0.0013)	0.7595
JSFP	0.9818	(0.0012)	0.7571
FRD	0.9825	(0.0012)	0.7545
RM	0.7005	(0.0117)	0.1228
GWFP	0.9840	(0.0012)	0.7475
MS-QL	0.9403	(0.0048)	0.6791

TABLE 2. Summary of the algorithm configurations evaluated in Section 5.2 at T = 400.

5.2.3. Results and discussion

The results of our simulations comparing the algorithms are given in the plots in Figure 3, for 200 iterations, with two standard errors as shown, while Table 2 shows the results at 400 iterations (the slow upward trajectory being maintained for all algorithms). The results show an overall good performance by all of the algorithms except RM. This is not entirely unexpected, as RM converges only to the set of correlated equilibria, and not to the pure Nash equilibria of the game. Since the pure Nash equilibria are local optima of the objective function, it is possible (and likely) that RM, in not finding these points with a high probability, remains relatively far from the local, and therefore the global, optima of the problem.

The Max–Sum algorithm performs comparably, but not as well as, our game–theoretic algorithms. This is somewhat surprising, particularly given the results of [1], which show better performance than many algorithms similar to the ones we investigate; we had anticipated similar results here. Since the algorithm uses the -A schedule, the reward functions will be accurately learnt, so the problem must be with the Max–Sum algorithm itself. Our interpretation of these results is that our formulation of the factor–graph induces more loops than the representation of [1]. This, in turn, adversely affects the accuracy of the computation performed by the Max–Sum algorithm. In other words, we surmise that the representation used by our game–theoretic algorithms causes a poor approximation of the optimal variable configuration by the Max–Sum algorithm.

It should also be noted that the Max–Sum algorithm, on each iteration, takes considerably more time to compute the next joint action, and accordingly, more energy (processing time itself is not a concern in this domain, where a time–step is a day). Specifically, on the problems in this section, the processing time per node at each time–step is approximately 8 seconds, (not including the latency for passing messages on an ad–hoc wireless network). This compares to between 0.5 and 20 milliseconds for the game theoretic algorithms, which pass only one message per time–step.

Finally, the minimum values recorded for each algorithm follows the same ordering as the average values, and do not contradict, nor even approach, the lower 1/2 bound.

5.3. Comparison across learning policy choice

In this section, we compare the performance of the algorithms, BRI and AP, across three different learning

Algorithm	Mean	2 S.E	Minimum
BRI	0.9798	(0.0013)	0.7573
BRI–A	0.9852	(0.0011)	0.7563
BRI–B	0.9631	(0.0015)	0.7163
AP	0.9813	(0.0013)	0.7595
AP-A	0.9857	(0.0011)	0.7570
AP–B	0.9436	(0.0015)	0.7134

TABLE 3. Summary of the algorithm configurations evaluated in Section 5.3 at T = 400.

policies, in order to evaluate the algorithms' sensitivity to their theoretical convergence conditions. Our main aim here is to empirically test the necessity of the learning policy derived in [15], which provides sufficient conditions for the algorithms to converge, but may sample with a higher probability than is required in practice. This is done by comparing the sampling scheme derived in [15] to one that only satisfies Lemma 4.1 (Q-value convergence). A secondary aim is to test the effect of using a policy that does not satisfy Lemma 4.1 so does not guarantee that the rewards converge, to test if and how this affects the behaviour of the algorithms. In this section, we examine only BRI and AP, because they are the algorithms considered in [15] so their consideration allows us to look at the effect of all three learning policies at once, and because the effects of learning policy choice on JSFP, FRD, RM and GWFP are almost idential to their effect on BRI and AP.

5.3.1. Three learning policy configurations

The first learning policy schedules used is the standard ε greedy policy used in the section above (i.e. satisfying both
the *Q*-learning and strong ergodicity conditions), namely: $\varepsilon(t) = 1/8 t^{-1/J_{im}}$ for all $i \in N$.

Under the second policy, denoted by the suffix "-A", the ε schedule used is $\varepsilon(t) = 1/8t^{-1/J_i}$, which satisfies Lemma 4.1 (Q-value convergence), but does not satisfy our conditions guaranteeing BRI and APs convergence to Nash equilibrium.

The third policy, denoted by the suffix "-B", means that the algorithm uses a Boltzmann learning policy:

$$\mathbb{P}(a_i^t = a_i') = \frac{e^{Q_i^t(a_i', \mathbf{z}_{-i})/\eta(t)}}{\sum_{a_i \in A_i} e^{Q_i^t(a_i, \mathbf{z}_{-i})/\eta(t)}}$$

with the temperature parameter following $\eta(t) = 16(0.9^t)$. This learning policy does not satisfy Lemma 4.1, so the the Q-values are not guaranteed to converge. This particular policy is chosen as it is the one used by [18] in their work on learning in games with unknown rewards.

5.3.2. Results and discussion

The BRI and AP variants plotted on Figures 4 and 5, respectively, with Table 3 summarising the results at T=400. First, note that the algorithms using learning policies that follow the standard schedule and the -A schedule (satisfying the *Q*-learning conditions only) both significantly outperform the algorithms using the -B

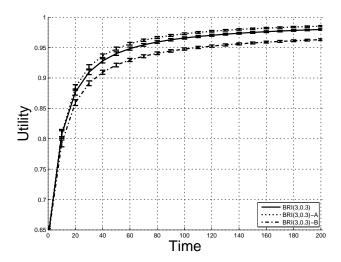


FIGURE 4. Results of the sensor network simulations for BRI, across the three learning policies.

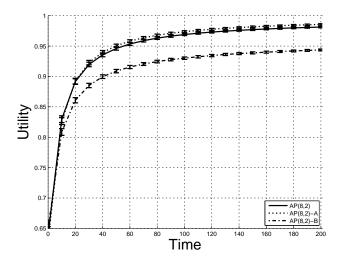


FIGURE 5. Results of the sensor network simulations for AP, across the three learning policies.

schedule. We propose that the reason for this is that the -B algorithms often become stuck in low-value configurations, because they do not sample new actions with a sufficient frequency to learn that this is the case. In contrast, the plots show that standard and -A schedules allow the algorithms to improve their average performance over time, as they can determine when they have been caught in "pseudo" local optima under incorrect reward estimates. This matches with the guaranteed convergence to Nash equilibrium for the standard schedule and the *Q*-learning convergence for learning policies following the -A schedule. This result is important as it highlights the need to guarantee that accurate estimates are made in games with rewards that are initially unknown.

Second, observe that, although statistically insignificant, the -A schedule algorithms perform better on average than

the those that satisfy all of the convergence conditions of [15]. We believe that the reason for this is that the standard schedule over–samples; that is, the sampling probabilities given by the standard schedule are much more than are required to ensure convergence to Nash equilibrium. In contrast, the -A schedule, under which the sampling probabilities go to zero much faster, but which are still sufficient to learn the full reward structure accurately, converges to Nash equilibrium much faster in practice (even though it is not guaranteed to do so in theory).

Finally, in these experiments, the minimum value recorded for each algorithm does not follow the same ordering as the average values, in contrast to the results in Section 5.2. This tempers our comment on over–sampling above, as it may indicate that for some problem instances, all of the convergence conditions enumerated in [15] are required for the algorithms to converge. Replications of the results on the same instances were, however, difficult to obtain, due to the stochastic nature of the algorithms (the problem instances were identical in their realisations), which curtailed any further analysis of this trait, and indicated a degree of sensitivity to initial conditions (given that the initial actions of the agents were chosen at random). As such, taking into account the insignificant benefit on average of using the -A schedule, it would appear that using the the standard learning policy, with its guaranteed convergence, is the prudent choice for a system designer.

5.4. Algorithm robustness to parameter choice

In this section, we show that a system designer can be confident that the algorithms' performance is not reliant on tuning the free parameters to the specific problem at hand. Sensitivity to the value of free parameters is a concern that is particularly prominent in wireless sensor networks and other problems where it may be difficult to adjust settings once the system is deployed. For the finite memory algorithms, we also investigate the effects of violating their theoretical convergence conditions.

5.4.1. Algorithm configurations

Having identified that the BRI, AP, JSFP, FRD and GWFP algorithms all perform relatively similarly for some specific parameter choices, we now test each algorithm's sensitivity to variations in their free parameters. Due to its lack of free parameters and its poor performance, we do not consider RM in this section.

In more detail, we run each of the algorithms across a range of parameter configurations, in order to test whether their performance is subject to any major variations as a result of different choices of parameters. Specifically, we consider:

- BRI with inertia using memory lengths from m = 1 to 12 and inertial constants from $\xi = 0.1$ to 0.9 (using the standard learning policy schedule);
- AP with memory length and sample size configurations ranging from (m,k) = (5,1) to (25,5), (using the standard learning policy schedule);

			ξ		
m	0.1	0.3	0.5	0.7	0.9
1	0.9836	0.9835	0.9805	0.9683	0.9039
	(0.0011)	(0.0012)	(0.0013)	(0.0014)	(0.0018)
3	0.9803	0.9798	0.9730	0.9658	0.8936
	(0.0013)	(0.0013)	(0.0014)	(0.0014)	(0.0019)
6	0.9762	0.9742	0.9689	0.9619	0.8875
	(0.0013)	(0.0013)	(0.0014)	(0.0015)	(0.0021)
9	0.9701	0.9736	0.9673	0.9583	0.8846
	(0.0014)	(0.0014)	(0.0014)	(0.0016)	(0.0023)
12	0.9567	0.9517	0.9504	0.9467	0.8698
	(0.0017)	(0.0017)	(0.0017)	(0.0019)	(0.0028)

TABLE 4. BRI (m,ξ) across memory lengths m and inertial constants ξ .

		(m, k)		
5, 1	10, 2	15, 3	20, 4	25, 5
0.9813	0.9802	0.9795	0.9768	0.9736
(0.0013)	(0.0013)	(0.0014)	(0.0015)	(0.0017)

TABLE 5. AP(m,k) across memory length m and sample size k combinations.

- JSFP with discounting factors ranging from $\gamma = 0.1$ to 0.4 and inertial constants from $\xi = 0.1$ to 0.9;
- FRD with discounting factors ranging from γ = 0.1 to 0.4 and inertial constants from ξ = 0.1 to 0.9;
- GWFP with learning rate parameters ranging from $\rho_{\alpha} = 0.5$ to 1.0 (nb. The *joint action learner* of [18] corresponds to $\rho_{\alpha} = 1.0$).

We now consider each of the processes in turn.

5.4.2. Results and discussion

Table 4 show that, for each memory length, BRI performs well for all inertial constants ≤ 0.5 , however, beyond this point value its performance drops off significantly. It has an overall peak performance at the shortest memory length, but this is in part a result of the sampling probability schedule of the standard learning policy keeping the sampling rate high for large values of t when the memory is long. Nonetheless, the effect of increasing m is less dramatic than that of moving ξ beyond 0.7.

Table 5 shows significant degradation in performance as the memory length of AP increases. This is in the main part due to the effect that increasing *m* has on the sampling schedule: AP with a memory of 25 is much more likely to randomly sample than AP with memory of 5, and this drives the process to be out–of–equilibrium much more often, and hence have a lower average solution quality.

Taking the results for BRI and AP together, in this domain it appears using longer memories is of no benefit for these algorithms. On one hand, longer memories lead to less thrashing when good solutions are found; that is, once a good solution is found, the algorithm has a greater chance of staying near it irrespective of the level of random play. On the other hand, if the algorithm moves away from a good solution, it can take more time to move back to a good configuration, as the longer memory length implies a greater

			ξ		
γ	0.1	0.3	0.5	0.7	0.9
0.1	0.9826	0.9824	0.9796	0.9689	0.9222
	(0.0011)	(0.0012)	(0.0012)	(0.0014)	(0.0019)
0.2	0.9825	0.9818	0.9787	0.9682	0.9213
	(0.0011)	(0.0012)	(0.0012)	(0.0015)	(0.0020)
0.3	0.9822	0.9811	0.9784	0.9646	0.9201
	(0.0011)	(0.0012)	(0.0012)	(0.0016)	(0.0021)
0.4	0.9817	0.9803	0.9763	0.9629	0.9179
	(0.0012)	(0.0012)	(0.0013)	(0.0017)	(0.0024)

TABLE 6. JSFP(γ , ξ) across discount factors γ and inertial constants ξ .

			ξ		
γ	0.1	0.3	0.5	0.7	0.9
0.1	0.9832	0.9831	0.9802	0.9696	0.9229
	(0.0011)	(0.0011)	(0.0012)	(0.0014)	(0.0018)
0.2	0.9826	0.9825	0.9792	0.9687	0.9208
	(0.0011)	(0.0012)	(0.0013)	(0.0014)	(0.0018)
0.3	0.9818	0.9811	0.9778	0.9638	0.9184
	(0.0011)	(0.0012)	(0.0013)	(0.0015)	(0.0022)
0.4	0.9810	0.9798	0.9756	0.9614	0.9149
	(0.0012)	(0.0012)	(0.0013)	(0.0018)	(0.0025)

TABLE 7. FRD(γ , ξ) across discount factors γ and inertial constants ξ .

inertia on the agents' actions. However, our results from the earlier section show that they actually perform better when the -A schedule is used. Under the -A schedule, sampling probabilities are independent of memory length, but the convergence of the algorithms is no longer theoretically guaranteed. This represents a trade—off between assured asymptotic convergence and finite time quality — if average performance is the key factor, rather than guaranteed worst—case bounds, then a case can be made for using the -A schedule with these algorithms.

Tables 6 and 7 show that JSFP and FRD perform similarly across the ranges of both of their parameters. Specifically, both have performances that are very stable across $\gamma=0.1$ to 0.4 and $\xi=0.1$ to 0.5. This indicates very robust performance across a large space of possible parameter settings. Additionally, both processes see a statistically significant drop–off in the quality of the solutions that they produce as ξ goes to 0.7 and beyond. We propose that the reason for this is that since the processes are evaluated over a finite duration, large values of ξ mean that the algorithms have failed to converge relatively more often than for smaller values of ξ , and this impacts on the average solution quality over a finite duration.

Finally, GWFP is remarkably consistent across the range of ρ_α considered, as shown in Table 8, with effectively no

		ρ	α		
0.5	0.6	0.7	0.8	0.9	1.0
0.9575	0.9569	0.9574	0.9576	0.9578	0.9579
(0.0070)	(0.0070)	(0.0070)	(0.0071)	(0.0070)	(0.0070)

TABLE 8. GWFP(ρ_{α}) across learning rate ρ_{α} .

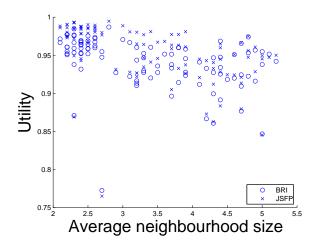


FIGURE 6. Utility at T = 200 versus problem density, as measured by the average neighbourhood size in a problem instance.

variation between different parameter choices.

Considering the JSFP, FRD (over their well-performing parameter range) and GWFP processes together reveals a great similarity in the quality of the solutions that they return. This indicates that there are several options available over the choice of process used, so if there is some other domain–specific benefits to using one of these over the others, then it can be happily applied to the problem at hand.

5.5. Comparison across problem density

In this section we test the algorithms on problems with more and less dense interaction structure. Note that it is not the case that overall larger networks are required to show how the algorithms perform at scale; these algorithms are designed to scale as long as the sparsity of the problem is fixed. That is, in a real deployment, the computational bottleneck will not be in the total number of agents (n) but in the size of the largest neighbourhood $(\max(J_i))$. However, since we simulate the network on a single machine, attempting to run large problems (large n) for any size of $\max(J_i)$, without the significant parallel processing capacity that an n-node sensor network possess, leads only to long processing times, and doesn't contribute any insights to our analysis. Thus, we examine problems with varying levels of density in the agents' interactions.

5.5.1. Problem generation

The problems in this section are constructed by varying the sensing radius of each node, thereby inducing more or fewer numbers of overlapping regions with nearby sensors. The average number of neighbours, $|\overline{\mathbf{v}}| = 1/n\sum |\mathbf{v}_i|$, was recorded for each problem instance, and we generated scenarios with a range of $|\overline{\mathbf{v}}|$ between 2 and 5.5. For each instance, BRI and JSFP both were run 50 times, and the average proportion of the optimum was recorded for each.

Average $ v_i $	Computation time (ms)
< 2.5	0.5
2.5-3.5	1.8
3.5-4.5	5.4
> 4.5	13.4

TABLE 9. Summary of computation time vs problem density (average neighbourhood size).

5.5.2. Results and discussion

Each point on Figure 6 represents the average result at T=200 for one problem instance solved using one of BRI or JSFP (circles and crosses, respectively). Standard errors for each point are similar to those in Figure 3, so are omitted. The results show a small but statistically significant downward linear trend in the quality of solutions produced by the algorithms, and also a significant increase their variability.

These results are expected, as the more dense the problem becomes, the greater the chance that the problem contains many local optima. Recall that the game–theoretic algorithms are in some sense local search algorithms, in that their convergent points are (a subset of) the set of local optima. Thus, in problems with many local optima, it is not surprising that sometimes the algorithms converge to points that are a substantial distance from the global optimum, and increasingly so with more local optima. Nonetheless, the slide in average performance in denser problems is quite reasonable; however, the increase in variability at higher |v| may not suit some applications.

The average computation time per time–step per node was also recorded (across both BRI and JSFP), and is displayed for four bins in Table 9. It shows an fairly rapid increase, which is evience of the exponential growth in the agents' local joint action spaces as the number of neighbours increases, but on average, computations are still completed within 20ms for the densest problems.

6. CONCLUSION

In this article, we have demonstrated how a distributed optimisation problem can be formulated as a marginal contribution potential game, and then shown that *Q*-learning versions of the finite-memory better reply processes, adaptive play, joint-strategy fictitious play, fading-memory regret monitoring and generalised weakened fictitious play processes all provide effective methods for solving them. These processes have the additional advantage over their alternatives of having bounds on their worst-case convergence points, if the optimisation problem is transformed into a game using marginal contribution payoffs. We have also identified a number of tradeoffs in the choice of process and/or free parameters for these processes, which give the designer of a large distributed system flexibility in their choice of algorithm.

The most pressing outstanding question from this work is ascertaining if the infinite memory algorithms can be proven to converge to Nash equilibrium in potential games with noisy rewards. Some work towards this goal indicates that it is difficult to show for generalised weakened fictitious play processes, and may be easier for the joint-strategy memory algorithms joint-strategy fictitious play and fading-memory regret monitoring.

Another direction for future work is the use of similar methods of constructing marginal contribution games in the setting of sequential decision–making problems such as Markov decision processes or stochastic games, which may then be solved by local iterative algorithms like the ones considered in this article. However, an additional complication in these settings is the observability of the problem, which may require significant new adaptations to the existing distributed algorithms.

ACKNOWLEDGEMENT

This work was supported by the ALADDIN (Autonomous Learning Agents for Decentralised Data and Information Systems) project and was jointly funded by a BAE Systems and EPSRC (Engineering and Physical Sciences Research Council) strategic partnership (EP/C548051/1). Archie Chapman was supported by University of Sydney Business School Post-doctoral Fellowship. The authors thank W. T. Luke Teacy for supplying an instantiation of the Max—Sum algorithm.

REFERENCES

- [1] Farinelli, A., Rogers, A., and Jennings, N. R. (2008) Maximising sensor network efficiency through agent–based coordination of sense/sleep schedules. Workshop on Energy in Wireless Sensor Networks in conjunction with DCOSS 2008, Marina Del Rey, CA, USA, June, pp. IV–43–IV–56.
- [2] Monderer, D. and Shapley, L. S. (1996) Potential games. Games and Economic Behavior, 14, 124–143.
- [3] Groves, T. (1973) Incentives in Teams. *Econometrica*, 41, 617–631.
- [4] Wolpert, D. H. and Tumer, K. (2002) Collective intelligence, data routing and Braess' paradox. *Journal of Artificial Intelligence Research*, **16**, 359–387.
- [5] Marden, J. R. and Wierman, A. (2008) Distributed welfare games. Proceedings of the 47th IEEE Conference on Decision and Control (CDC-08), Cancun, Mexico, 9-11 December.
- [6] Fudenberg, D. and Levine, D. K. (1998) *The Theory of Learning in Games*. MIT Press, Cambridge, MA.
- [7] Leslie, D. S. and Collins, E. J. (2006) Generalised weakened fictitious play. *Games and Economic Behavior*, **56**, 285–298.
- [8] Young, H. P. (2004) Strategic Learning and its Limits. Oxford University Press.
- [9] Young, H. P. (1993) The evolution of conventions. *Econometrica*, **61**, 57–84.
- [10] Marden, J. R., Arslan, G., and Shamma, J. S. (2007) Regret based dynamics: Convergence in weakly acyclic games. Proceedings of the 6th International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS '07), New York, NY, USA, 14–18 May, pp. 42:1–42:8. ACM.
- [11] Marden, J. R., Arslan, G., and Shamma, J. (2009) Joint strategy fictitious play with inertia for potential games. *IEEE Transactions on Automatic Control*, **54**, 208–220.

- [12] Sutton, R. S. and Barto, A. G. (1998) *Reinforcement Learning*. MIT Press, Cambridge, MA.
- [13] Kearns, M., Littman, M., and Singh, S. (2001) Graphical models for game theory. *Proceedings of the 17th on Uncertainty in Artificial Intelligence (UAI '01)*, Seattle, WA, USA, 2–5 August, pp. 253–260. Morgan Kaufmann.
- [14] Gottlob, G., Greco, G., and Scarcello, F. (2005) Pure Nash equilibria: Hard and easy games. *Journal of Artificial Intelligence Research*, **24**, 357–406.
- [15] Chapman, A. C., Leslie, D. S., Rogers, A., and Jennings, N. R. (2013) Convergent learning algorithms for unknown reward games. SIAM Journal on Control and Optimization, in press.
- [16] Vetta, A. (2002) Nash Equilibria in Competitive Societies, with Applications to Facility Location, Traffic Routing and Auctions. *IEEE Symposium on the Foundations of Computer Science (FOCS'02)*, Vancouver, Canada, 16–19 November, pp. 416–425.
- [17] Kansal, A., Hsu, J., Zahedi, S., and Srivastava, M. B. (2007) Power management in energy harvesting sensor networks. ACM Transactions on Embedded Computing Systems, 6, 32:1–32:38.
- [18] Claus, C. and Boutilier, C. (1998) The dynamics of reinforcement learning in cooperative multiagent systems. In Proceedings of the 15th AAAI National Conference on Artificial Intelligence, Madison, WI, USA, 26–30 July, pp. 746–752. AAAI Press.
- [19] Singh, S. P., Jaakkola, T., Littman, M. L., and C. Szepesvári (2000) Convergence results for single–step on–policy reinforcement–learning algorithms. *Machine Learning*, 38, 287–308.
- [20] Friedman, J. W. and Mezzetti, C. (2001) Learning in games by random sampling. *Journal of Economic Theory*, **98**, 55–84.
- [21] Kandori, M., Mailath, G. J., and Rob, R. (1993) Learning, mutation, and long run equilibria in games. *Econometrica*, **61**, 29–56.
- [22] Fudenberg, D. and Kreps, D. M. (1993) Learning mixed equilibria. *Games and Economic Behavior*, **5**, 320–367.
- [23] Hart, S. and Mas-Colell, A. (2000) A simple adaptive procedure leading to correlated equilibrium. *Econometrica*, 68, 1127–1150.
- [24] Blackwell, D. (1956) An analogue of the minimax theorem for vector payoffs. *Pacific Journal of Mathematics*, **6**, 1–8.
- [25] Brown, G. W. (1951) Iterative solution of games by fictitious play. In Koopmans, T. C. (ed.), Activity Analysis of Production and Allocation, pp. 374–376. John Wiley & Sons, Inc., New York.