This is a repository copy of *Multi-Agent Credit Assignment in Stochastic Resource Management Games*.

White Rose Research Online URL for this paper:
https://eprints.whiterose.ac.uk/121359/

Version: Accepted Version

Multi-Agent Credit Assignment in Stochastic Resource Management Games

PATRICK MANNION[1,2], SAM DEVLIN[3], JIM DUGGAN[2], ENDA HOWLEY[2]

[1]*Department of Computer Science & Applied Physics, Galway-Mayo Institute of Technology, Galway, Ireland*
E-mail: patrick.mannion@gmit.ie

[2]*Discipline of Information Technology, National University of Ireland Galway, Galway, Ireland*
E-mail: jim.duggan@nuigalway.ie, enda.howley@nuigalway.ie

[3]*Department of Computer Science, University of York, York, UK*
E-mail: sam.devlin@york.ac.uk

Abstract

Multi-Agent Systems (MAS) are a form of distributed intelligence, where multiple autonomous agents act in a common environment. Numerous complex, real world systems have been successfully optimised using Multi-Agent Reinforcement Learning (MARL) in conjunction with the MAS framework. In MARL agents learn by maximising a scalar reward signal from the environment, and thus the design of the reward function directly affects the policies learned. In this work, we address the issue of appropriate multi-agent credit assignment in stochastic resource management games. We propose two new Stochastic Games to serve as testbeds for MARL research into resource management problems: the Tragic Commons Domain and the Shepherd Problem Domain. Our empirical work evaluates the performance of two commonly used reward shaping techniques: Potential-Based Reward Shaping and difference rewards. Experimental results demonstrate that systems using appropriate reward shaping techniques for multi-agent credit assignment can achieve near optimal performance in stochastic resource management games, outperforming systems learning using unshaped local or global evaluations. We also present the first empirical investigations into the effect of expressing the same heuristic knowledge in state- or action-based formats, therefore developing insights into the design of multi-agent potential functions that will inform future work.

1 Introduction

In a Multi-Agent System (MAS), multiple agents act autonomously in a common environment. Agents in a MAS may be cooperative, competitive, or may exhibit elements of both behaviours. Agents in a cooperative MAS are designed to work together to achieve a system-level goal (Wooldridge, 2001), whereas agents in a competitive MAS are self-interested and may come into conflict with each other when trying to achieve their own individual goals. Numerous complex, real world systems have been successfully optimised using the MAS framework, including traffic signal control (Mannion et al., 2016a), air traffic control (Tumer and Agogino, 2007), electricity generator scheduling (Mannion et al., 2016c), network security (Malialis et al., 2015), water resource management (Mason et al., 2016), and data routing in networks (Wolpert and Tumer, 2002), to name a few examples.

Reinforcement Learning (RL) has proven to be successful in developing suitable joint policies for cooperative MAS in all of the problem domains mentioned above. RL agents learn by

maximising a scalar reward signal from the environment, and thus the design of the reward function directly affects the policies learned. The issue of credit assignment in Multi-Agent Reinforcement Learning (MARL) is an area of active research with numerous open questions. Reward shaping has been investigated as a mechanism to guide exploration in both single- and multi-agent RL problems, with promising results. Potential-Based Reward Shaping (PBRS) is a form of reward shaping that provides theoretical guarantees while guiding agents using heuristic knowledge about a problem. Difference rewards are another commonly used form of reward shaping that allow an individual agent's contribution to the system utility to be evaluated, in order to provide a more informative reward signal. Both of these reward shaping methodologies have proven to be effective in addressing the multi-agent credit assignment problem (see e.g. Devlin et al. (2011a, 2014); Mannion et al. (2016b); Tumer and Agogino (2007); Wolpert and Tumer (2002)).

In this paper we explore the issue of credit assignment in stochastic resource management games. To facilitate this investigation, we introduce two new Stochastic Games to serve as testbeds for MARL research into resource management problems: the Tragic Commons Domain (TCD) and the Shepherd Problem Domain (SPD).

The Tragic Commons Domain is a Stochastic Game which is inspired by both N-player dilemmas and resource dilemmas from the field of Game Theory. When multiple self-interested agents share a common resource, each agent seeks to maximise its own gain without consideration for the welfare of other agents in the system. In the case of a scarce resource, over-exploitation occurs when all agents follow a greedy strategy. This can have disastrous consequences, in some cases damaging the resource to the detriment of all agents in the system. This scenario is referred to as the Tragedy of the Commons. We introduce the TCD as a means to study resource dilemmas using the MAS paradigm, and investigate techniques to avoid over-exploitation of the resource, thus maximising the social welfare of the system.

By contrast, our study in the SPD deals with maximising the social welfare of a system when congestion in a group of common resources is unavoidable. The SPD models a system of interconnected pastures, where each shepherd (agent) begins in a certain pasture and must decide in which pasture it will graze its herd of animals. This is a congestion problem, as the number of herds that must be accommodated is far greater than the total capacity of the available pastures. Therefore, the goal is to minimise the effect of congestion on the system. Insights gained while studying congestion problems of this type may be useful when tackling congestion that occurs naturally in many real world systems, e.g. in road, air or sea transportation applications.

We evaluate the performance of various credit assignment techniques in these empirical studies. Our results demonstrate that systems using appropriate reward shaping techniques for multi-agent credit assignment can achieve near optimal performance in stochastic resource management games, outperforming systems learning using unshaped local or global evaluations. Additionally, this work contains the first empirical investigations into the effect of expressing the same heuristic knowledge in state- or action-based formats, therefore developing insights into the design of multi-agent potential functions that will inform future work.

In the next section of this paper, we discuss the necessary terminology and relevant literature. This is followed by our first empirical study in the Tragic Commons Domain. We then present the results of our second empirical study in the Shepherd Problem Domain. Finally, we conclude this paper with a discussion of our findings and possible future extensions to this work.

2   Background & Related Work

2.1   Reinforcement Learning

Reinforcement Learning (RL) is a powerful Machine Learning paradigm, in which autonomous agents have the capability to learn through experience. An RL agent learns in an unknown environment, usually without any prior knowledge of how to behave. The agent receives a scalar reward signal r based on the outcomes of previously selected actions, which can be either negative

or positive. Markov Decision Processes (MDPs) are considered the de facto standard when formalising problems involving a single agent learning sequential decision making (Wiering and van Otterlo, 2012). A MDP consists of a reward function R, set of states S, set of actions A, and a transition function T (Puterman, 1994), i.e. a tuple $< S, A, T, R >$. When in any state $s \in S$, selecting an action $a \in A$ will result in the environment entering a new state $s' \in S$ with probability $T(s, a, s') \in (0, 1)$, and give a reward $r = R(s, a, s')$.

An agent's behaviour in its environment is determined by its policy π. A policy is a mapping from states to actions that determines which action is chosen by the agent for a given state. The goal of any MDP is to find the best policy (one which gives the highest expected sum of discounted rewards) (Wiering and van Otterlo, 2012). The optimal policy for a MDP is denoted π^*. Designing an appropriate reward function for the environment is important, as an RL agent will attempt to maximise the return from this function, which will determine the policy learned.

RL can be classified into two paradigms: model-based (e.g. Dyna, Rmax) and model-free (e.g. Q-Learning, SARSA). In the case of model-based approaches, agents attempt to learn the transition function T, which can then be used when making action selections. By contrast, in model-free approaches knowledge of T is not a requirement. Model-free learners instead sample the underlying MDP directly in order to gain knowledge about the unknown model, in the form of value function estimates (Q values). These estimates represent the expected reward for each state action pair, which aid the agent in deciding which action is most desirable to select when in a certain state. The agent must strike a balance between exploiting known good actions and exploring the consequences of new actions in order to maximise the reward received during its lifetime. Two algorithms that are commonly used to manage the exploration exploitation trade-off are ϵ-greedy and softmax (Wiering and van Otterlo, 2012).

Q-Learning (Watkins, 1989) is one of the most commonly used RL algorithms. It a model-free learning algorithm that has been shown to converge to the optimum action-values with probability 1, so long as all actions in all states are sampled infinitely often and the action-values are represented discretely (Watkins and Dayan, 1992). In Q-Learning, the Q values are updated according to the equation below:

$$Q(s, a) \leftarrow Q(s, a) + \alpha[r + \gamma \max_{a'} Q(s', a') - Q(s, a)] \tag{1}$$

where $\alpha \in [0, 1]$ is the learning rate and $\gamma \in [0, 1]$ is the discount factor.

2.2 Multi-Agent Reinforcement Learning

The single-agent MDP framework becomes inadequate when we consider multiple autonomous learners acting in the same environment. Instead, the more general Stochastic Game (SG) may be used in the case of a MAS (Buşoniu et al., 2010). A SG is defined as a tuple $< S, A_{1...N}, T, R_{1...N} >$, where N is the number of agents, S is the set of states, A_i is the set of actions for agent i (and A is the joint action set), T is the transition function, and R_i is the reward function for agent i.

The SG looks very similar to the MDP framework, apart from the addition of multiple agents. In fact, for the case of $N = 1$ a SG then becomes a MDP. The next environment state and the rewards received by each agent depend on the joint action of all of the agents in the SG. Note also that each agent may receive a different reward for a state transition, as each agent has its own separate reward function. In a SG, the agents may all have the same goal (collaborative SG), totally opposing goals (competitive SG), or there may be elements of collaboration and competition between agents (mixed SG).

One of two different approaches is typically used when RL is applied to MAS: multiple individual learners or joint action learners. In the former case multiple agents are deployed into a common environment, each using a single-agent RL algorithm, whereas joint action learners use multi-agent specific algorithms which take account of the presence of other agents. When multiple

self-interested agents learn and act together in the same environment, it is generally not possible for all agents to receive the maximum possible reward. Therefore, MAS are typically designed to converge to a Nash Equilibrium (Shoham et al., 2007). While it is possible for multiple individual learners to converge to a point of equilibrium, there is no theoretical guarantee that the agents will converge to a Pareto optimal joint policy.

2.3   Reward Shaping

RL agents typically learn how to act in their environment guided by the reward signal alone. Reward shaping provides a mechanism to guide an agent's exploration of its environment, via the addition of a shaping signal to the reward signal naturally received from the environment. The goal of this approach is to increase learning speed and/or improve the final policy learned. Generally, the reward function is modified as follows:

$$R' = R + F \tag{2}$$

where R is the original reward function, F is the additional shaping reward, and R' is the modified reward signal given to the agent.

Empirical evidence has shown that reward shaping can be a powerful tool to improve the learning speed of RL agents; however, it can have unintended consequences. A classic example of reward shaping gone wrong is reported by Randløv and Alstrøm (1998). The authors designed an RL agent capable of learning to cycle a bicycle towards a goal, and used reward shaping to speed up the learning process. However, they encountered the issue of positive reward cycles due to a poorly designed shaping function. The agent discovered that it could accumulate a greater reward by cycling in circles continuously to collect the shaping reward encouraging it to stay balanced, than it could by reaching the goal state. As we discussed earlier, an RL agent will attempt to maximise its long-term reward, so the policy learned depends directly on the reward function. Thus, shaping rewards in an arbitrary fashion can modify the optimal policy and cause unintended behaviour.

Ng et al. (1999) proposed Potential-Based Reward Shaping (PBRS) to deal with these shortcomings. When implementing PBRS, each possible system state has a certain potential, which allows the system designer to express a preference for an agent to reach certain system states. For example, states closer to the goal state of a problem domain could be assigned higher potentials than those that are further away. Ng et al. (1999) defined the additional shaping reward F for an agent receiving PBRS as shown in Eqn. 3 below:

$$F(s, s') = \gamma \Phi(s') - \Phi(s) \tag{3}$$

where $\Phi(s)$ is the potential function which returns the potential for a state s, and γ is the same discount factor used when updating value function estimates. PBRS has been proven not to alter the optimal policy of a single agent acting in infinite-horizon and finite-horizon MDPs (Ng et al., 1999), and thus does not suffer from the problems of arbitrary reward shaping approaches outlined above. In single agent RL, even with a poorly designed potential function, the worst case is that an agent may learn more slowly than without shaping but the final policy is unaffected.

However, the form of PBRS proposed by Ng et al. (1999) can only express a designer's preference for an agent to be in a certain state, and therefore cannot make use of domain knowledge that recommends actions. Wiewiora et al. (2003) proposed an extension to PBRS called Potential-Based Advice, that includes actions as well as states in the potential function. The authors propose two methods of Potential-Based Advice: Look-Ahead Advice and Look-Back Advice. The former method defines the additional reward received F as follows:

$$F(s, a, s', a') = \gamma \Phi(s', a') - \Phi(s, a) \tag{4}$$

Wiewiora et al. (2003) provided a proof of policy invariance for Look-Ahead Advice for single agent learning scenarios. No corresponding proof has been provided for Look-Back Advice, although empirical results suggest that this method also does not alter the optimal policy in single agent learning scenarios. To maintain policy invariance when using Look-Ahead Advice, the agent must choose the action that maximises the sum of both Q-value and potential:

$$\pi(s) = argmax_a(Q(s,a) + \Phi(s,a)) \tag{5}$$

where $\pi(s)$ is the agent's policy in state s (the action that will be chosen by the agent in state s). We will use Wieiwora's Look-Ahead Advice to test action-based heuristics in the experimental sections of this paper. From this point forth, we will use the abbreviations sPBRS and aPBRS to refer to PBRS approaches with state-based and action-based potential functions respectively.

In MARL, work by Devlin and Kudenko (2011) proved that PBRS does not alter the set of Nash equilibria of a SG. Furthermore, Devlin and Kudenko (2012) also proved that the potential function can be changed dynamically during learning, while still preserving the guarantees of policy invariance and consistent Nash equilibria. PBRS does not alter the set of Nash equilibria of a MAS, but it can affect the joint policy learned. It has been empirically demonstrated that agents guided by a well-designed potential function can learn at an increased rate and converge to better joint policies, when compared to agents learning without PBRS (Devlin et al., 2011a). However, with an unsuitable potential function, agents learning with PBRS can converge to worse joint policies than those learning without PBRS.

2.4 Multi-Agent Credit Assignment Structures

Here we introduce the MARL credit assignment structures that we will evaluate in the experimental sections of this paper. Two typical reward functions for MARL exist: local rewards unique to each agent and global rewards representative of the group's performance. In addition to these basic credit assignment structures, we also evaluate the performance of several other shaped reward functions.

A **local reward** (L_i) is based on the utility of the part of a system that agent i can observe directly. Individual agents are self-interested, and each will selfishly seek to maximise its own local reward signal, often at the expense of global system performance when locally beneficial actions are in conflict with the optimal joint policy.

A **global reward** (G) provides a signal to the agents which is based on the utility of the entire system. Rewards of this form encourage all agents to act in the system's interest, with the caveat that an individual agent's contribution to the system performance is not clearly defined. All agents receive the same reward signal, regardless of whether their actions actually improved the system performance.

A **difference reward** (D_i) is a shaped reward signal that aims to quantify each agent's individual contribution to the system performance (Wolpert et al., 2000). The unshaped reward is $R = G(z)$, and the shaping term is $F = -G(z_{-i})$. Formally:

$$D_i(z) = G(z) - G(z_{-i}) \tag{6}$$

where D_i is the reward received by agent i, $G(z)$ is the global system utility, and $G(z_{-i})$ is referred to as the counterfactual. The counterfactual term represents the global utility for a theoretical system without the contribution of agent i. Here z is a general term that may represent either states or state-action pairs, depending on the specific application. Difference rewards are a well-established shaping methodology, with many successful applications in MARL (e.g. Mannion et al. (2016d); Tumer and Agogino (2007); Wolpert and Tumer (2002)).

Counterfactual as Potential (CaP), proposed by Devlin et al. (2014), is an automated method of generating multi-agent potential functions using the same knowledge represented by difference rewards. CaP automatically assigns potentials to states using the counterfactual term

$G(z_{-i})$, so that $\Phi(s) = G(z_{-i})$. In this framework, the unshaped reward $R = G(z)$, and the shaping reward F is calculated as normal in PBRS according to Eqn. 3. As $\Phi(s)$ for agent i is in fact based on the state of the other agents in the system, the potential function is dynamic, and CaP is thus an instance of Dynamic PBRS (Devlin and Kudenko, 2012). CaP therefore preserves the guarantee of consistent Nash equilibria, while incorporating knowledge based on difference rewards in an automated manner.

3 Tragic Commons Domain

3.1 Problem Description

In this section we introduce the Tragic Commons Domain (TCD), a Stochastic Game designed with the intent of studying resource dilemmas using the MAS paradigm. As mentioned earlier, this domain was inspired by N-player dilemmas and resource dilemmas from the field of Game Theory. An example of a simple dilemma is the Prisoner's Dilemma (Binmore, 2012), a well-studied problem in which two players (agents) have the options to either cooperate or defect. Mutual cooperation results in the highest global utility; however if one player defects he receives a higher reward, while the cooperating player receives the sucker's payoff. If both players choose to defect, they both receive a low payoff. The principles of the Prisoner's Dilemma can be extended to N-player dilemmas, where N players must choose to cooperate or defect (Binmore, 2012).

Resource dilemmas are an example of an N-player dilemma, where multiple self-interested agents share a common resource, and each seeks to maximise its own returns. Agents may cooperate to conserve the resource and utilise it in a sustainable manner, or they may defect and selfishly attempt to extract the maximum value possible from the resource. When a majority of agents act conservatively, there is an incentive for agents to defect so that they will receive a better than average payoff.

However, as more agents choose to defect the resource becomes over-exploited, and the global payoff is less than if all agents cooperate. Furthermore, when over-exploitation becomes the norm there is no incentive for a self-interested agent to act conservatively; to do so would reduce the individual's payoff. The dominant strategy is thus to defect but, paradoxically, if all agents cooperate they maximise the collective benefit, and each does better than if all play the dominant strategy.

There are numerous real-world examples of resource or commons dilemmas; any natural resource that is owned and used in common by multiple entities presents a dilemma of how best to utilise it in a sustainable manner. Examples include fish stocks, wildlife, water resources, woodlands and common pastures. There are also examples of negative commons, e.g. atmospheric pollution or fraudulent activities, where an individual may benefit by damaging a common resource to the detriment of all.

Previous research into commons dilemmas has used approaches such as evolutionary computation (Howley and Duggan, 2011) and learning automata (de Jong and Tuyls, 2009) to develop cooperation among agents. In this empirical study, we explore the possibility of using PBRS to encourage cooperative strategies among self-interested RL agents. Agents learning using a local reward function L are purely self-interested, and seek to maximise their own utility. As a comparison and to provide an upper bound on possible performance, we have also tested agents using reward functions that are explicitly designed to maximise the system utility (the global reward function G and difference reward function D).

In the Tragic Commons Domain, N agents each have the right to graze their livestock in a common pasture. At each timestep, the agents decide how many of their own animals will graze in the commons until the next timestep. Thus the action set for each agent is $A = \{a_{min}, ..., a_{max}\}$, where a_{min} and a_{max} correspond to the minimum and maximum number of animals that each agent is allowed to graze in the commons. The state for each agent is defined as the number of its own animals currently grazing in the commons. We define the occupancy of the commons ς
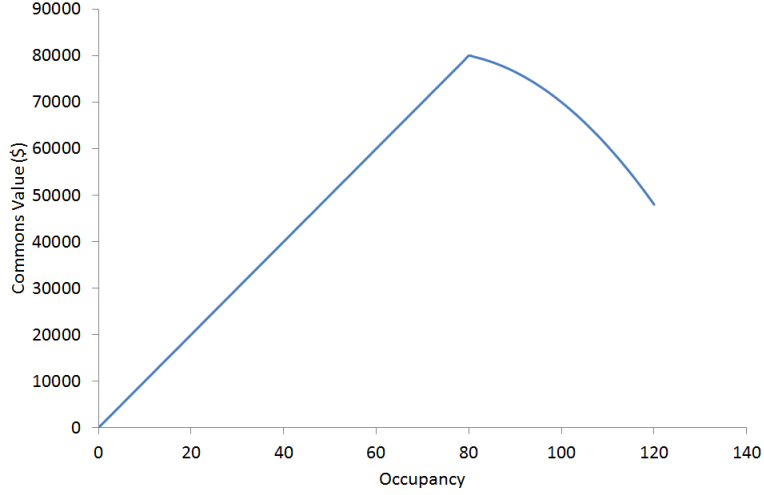
Figure 1: Occupancy vs. commons value for an entire episode of the Tragic Commons Domain

as the sum of the animals placed in the commons by all agents $n \in N$ at any particular time:

$$\varsigma = \sum_{n=1}^{n=N} s_n \tag{7}$$

Animals left grazing in the commons will increase in value between timesteps. We define χ as the increase in value for an animal left grazing in the commons for one timestep. However, there is a maximum number of animals the commons can sustain without deterioration from overgrazing. The capacity of the commons ψ is defined as the number of animals which can sustainably graze the commons without causing damage. Animals left grazing in the commons when it is at or below capacity will increase in value by the maximum χ_{max}. When the commons is over capacity, animals increase in value by a lower rate due to the lower quantity and quality of food available. The value of χ is related to the occupancy of the commons by the following equation:

$$\chi(\varsigma) = \begin{cases} \chi_{max} & \text{if } \varsigma <= \psi \\ \chi_{max} - \dfrac{(\chi_{max} - \chi_{min}) \times (\varsigma - \psi)}{\varsigma_{max} - \psi} & \text{otherwise} \end{cases} \tag{8}$$

Thus, the global benefit is maximised when $\varsigma = \psi$, as all animals in the commons increase in value by the maximum rate. Values of ς that are less than ψ do not utilise the resource to its full potential, whereas values of ς greater than ψ result in damage to the resource, and a corresponding decrease in the collective benefit. Agents which select actions less than or equal to $\frac{\psi}{N}$ are acting in a fair and conservative manner, analogously to the agents who choose to cooperate in the earlier example. Whereas agents who choose actions greater than $\frac{\psi}{N}$ are acting in an unfair and exploitative manner, similarly to agents who choose to defect in a classic N-Player dilemma.

We use the commons value as a measure for the system utility, where the commons value for an episode is the sum of the products of χ and ς for each timestep. The relationship between the commons value and ς for the parameter values that we have chosen for our experiments is shown in Fig. 1.

The local reward for a self-interested agent i is calculated based on the added value per animal multiplied by the number of animals the agent currently has in the commons. Formally:

$$L_i(s_i, a_i, s_i') = \chi(\varsigma')s_i' \tag{9}$$

The global reward is calculated based on the added value per animal multiplied by the total number of animals in the commons. This is a per-timestep version of the commons value metric that is used to measure overall episode performance. Formally:

$$G(z) = \chi(\varsigma')\varsigma' \tag{10}$$

The counterfactual for agent i, to be used with D and CaP, is calculated by assuming that the agent chose the same action as in the previous timestep i.e. $a = s = s'$. This means that a counterfactual occupancy $\varsigma(z_{-i})$ and a counterfactual animal value $\chi(z_{-i})$ must be calculated. The counterfactual term is then calculated as the product of $\varsigma(z_{-i})$ and $\chi(z_{-i})$. Formally:

$$\varsigma(z_{-i}) = \sum_{\substack{n=1 \\ n \neq i}}^{n=N} s'_n + s_i \tag{11}$$

$$\chi(z_{-i}) = \chi(\varsigma(z_{-i})) \tag{12}$$

$$G(z_{-i}) = \chi(z_{-i})\varsigma(z_{-i}) \tag{13}$$

3.2 PBRS Heuristics

To explore the effectiveness of PBRS in this domain, we apply three different manual heuristics in addition to the automated CaP heuristic. Each heuristic is expressed in both a state-based and action-based form. The heuristics used are:

- **Fair:** Agents are encouraged to act in a fair manner, where each agent is encouraged to graze the optimal number of animals (i.e. $\frac{\psi}{N}$) in the commons. This heuristic is expected to perform extremely well, and will demonstrate the effect of PBRS when advice can be given about the optimal policy.

$$\Phi(s) = \begin{cases} \dfrac{\psi\chi_{max}}{N} & \text{if } s = \dfrac{\psi}{N} \\ 0 & \text{otherwise} \end{cases} \tag{14}$$

$$\Phi(s,a) = \begin{cases} \dfrac{\psi\chi_{max}}{N} & \text{if } a = \dfrac{\psi}{N} \\ 0 & \text{otherwise} \end{cases} \tag{15}$$

- **Opportunistic:** Agents opportunistically seek to utilise any spare capacity available in the commons. As the available spare capacity is dependent on the joint actions of all agents in the system, the value of this potential function changes over time. Thus, this heuristic is an instance of dynamic PBRS Devlin and Kudenko (2012). This is a weaker shaping than Fair, and is included to demonstrate the effect of PBRS with a sub-optimal heuristic, for contexts where the optimal policy is not known.

$$\Phi(s) = \begin{cases} s\chi_{max} & \text{if } \varsigma < \psi \\ 0 & \text{otherwise} \end{cases} \tag{16}$$

$$\Phi(s,a) = \begin{cases} a\chi_{max} & \text{if } \varsigma < \psi \\ 0 & \text{otherwise} \end{cases} \tag{17}$$

- **Greedy:** Agents are encouraged to behave greedily, where each agent seeks to graze the maximum possible number of animals in the commons. This is an example of an extremely poor heuristic, and is expected to reduce the performance of all agents receiving it.

$$\Phi(s) = \begin{cases} s_{max}\chi_{max} & \text{if } s = s_{max} \\ 0 & \text{otherwise} \end{cases} \tag{18}$$

$$\Phi(s, a) = \begin{cases} a_{max}\chi_{max} & \text{if } a = a_{max} \\ 0 & \text{otherwise} \end{cases} \tag{19}$$

3.3   Experimental Procedure

We test two variations of the TCD: a single-step version with $num\_timesteps = 1$ and a multi-step version with $num\_timesteps = 12$. The other experimental parameters used were as follows: $num\_episodes = 20,000$, $N = 20$, $\psi = 80$, $\chi_{max} = \$1000/num\_timesteps$, $\chi_{min} = \$400/num\_timesteps$, $a_{min} = s_{min} = 0$, $a_{max} = s_{max} = 6$, $\varsigma_{max} = 120$ (from Eqn. 7). All agents begin each episode with their initial state set to 0 (i.e. no animals grazing in the commons).

We apply multiple individual Q-Learning agents to the Tragic Commons Domain defined above, learning with credit assignment structures L, G, D, along with various shaping heuristics. The learning parameters for all agents are set as follows: $\alpha = 0.20$, $\gamma = 0.90$, $\epsilon = 0.10$. Both α and ϵ are reduced by multiplication with their respective decay rates at the end of each episode, with $alpha\_decay\_rate = 0.9999$ and $epsilon\_decay\_rate = 0.9999$. These values were selected following parameter sweeps to determine the best performing settings. Each agent's Q values for all state action pairs are initialised to 0 at the start of each experimental run.

Along with the MARL approaches tested, we have also conducted experiments with three simple agent types: Optimal, Random, and Greedy. Optimal agents always select the action equal to $\frac{\psi}{N}$, and thus will maximise the global benefit. The random agents select actions randomly with equal probability. The greedy agents always select a_{max} i.e. they play the dominant strategy, which is to graze as many animals as possible in the commons. Optimal and Greedy give the upper and lower performance bounds for this problem, and serve as a useful comparison to the MARL approaches tested.

3.4   Experimental Results

For the single-step version of the problem, Figs. 2a to 2d show learning curves for commons value and occupancy for the approaches tested. For the multi-step version, Figs. 3a to 3d show learning curves for commons value and occupancy. The average performance of the approaches tested over the final 2000 episodes in both problem versions is shown in Table 1.

All plots include error bars representative of the standard error of the mean based on 50 statistical runs. Specifically, we calculate the error as $\sigma/\sqrt{n}$ where σ is the standard deviation and n is the number of statistical runs. Error bars are included on all plots at 200 episode intervals. The plots show the average performance across the 50 statistical runs that were conducted at 10 episode intervals. All claims of statistical significance are supported by two-tailed t-tests assuming unequal variances, with $p = 0.05$ selected as the threshold for significance.

We can see from the learning curves that L starts out at quite a high level of performance, but this quickly degrades, eventually reaching a final policy close to that of the Greedy agents. The performance of L is significantly worse than the Random baseline in both the single step ($p = 2.36 \times 10^{-57}$) and multi-step ($p = 2 \times 10^{-105}$) versions of the problem. Self-interested agents learning with L greedily seek to exploit the resource, as evidenced by the occupancy curves in Figs. 2b and 3b, where the agents consistently learn to graze close to the maximum allowed number of animals in the commons by the final training episodes.

By contrast, the approaches explicitly designed to maximise the global utility (G and D) perform quite well in the Tragic Commons Domain. Rather than working at cross-purposes and seeking to exploit the resource to the maximum extent possible, agents learning using G and D converge to policies that utilise the shared resource in a fair and conservative manner, and both come close to the optimal level of performance in the single-step and multi-step variants of the problem. In both variants of the problem, D reaches a high level of performance very quickly, and learns more quickly than G. However, in the single-step TCD, G in fact reaches a higher final performance than D (99.2% of optimal performance for G, vs. 98.5% of optimal performance for
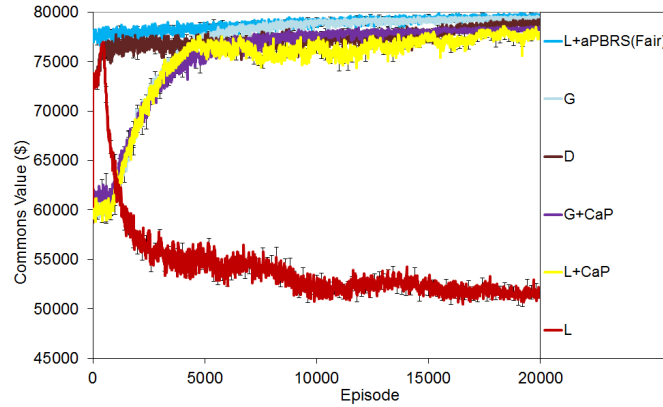
(a) Commons values for L, G, and D
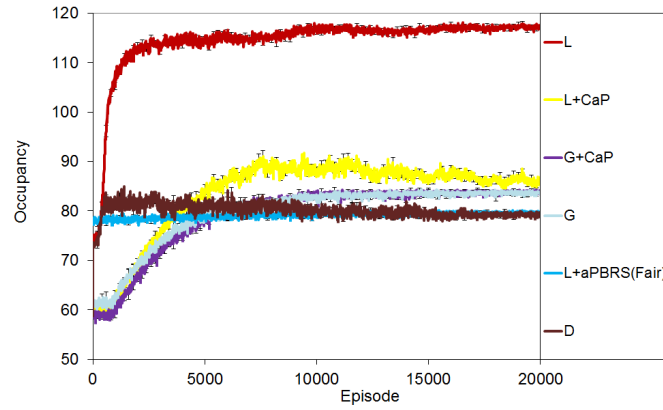
(b) Occupancy values for L, G, and D
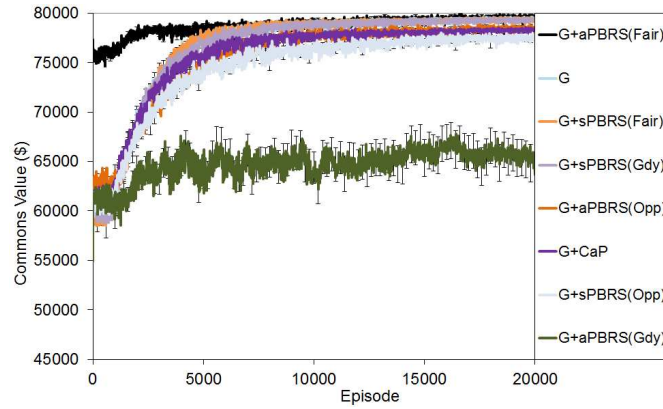
(c) Commons values for G with heuristics

(d) Commons values for L with heuristics

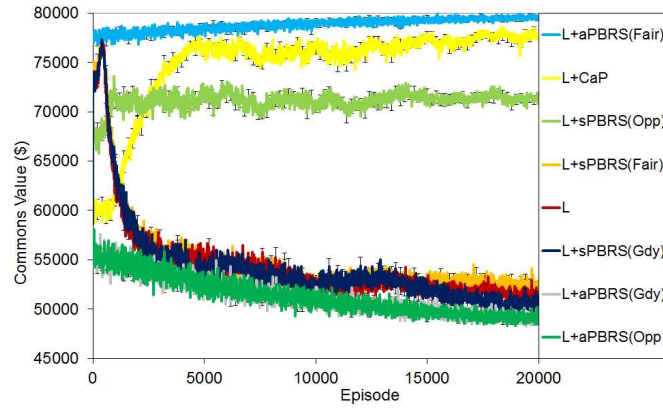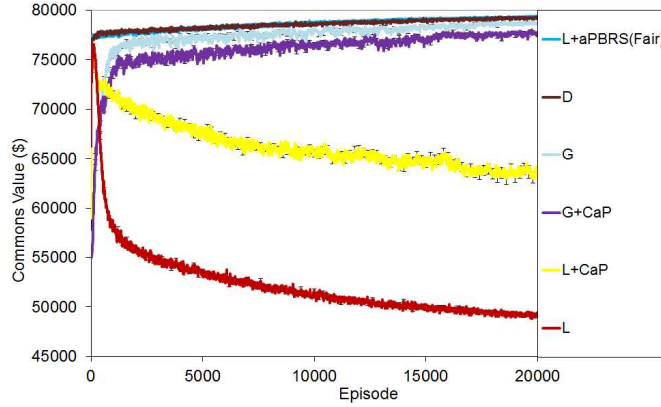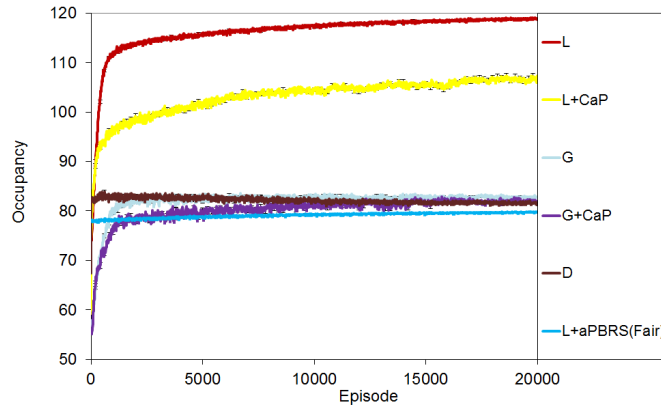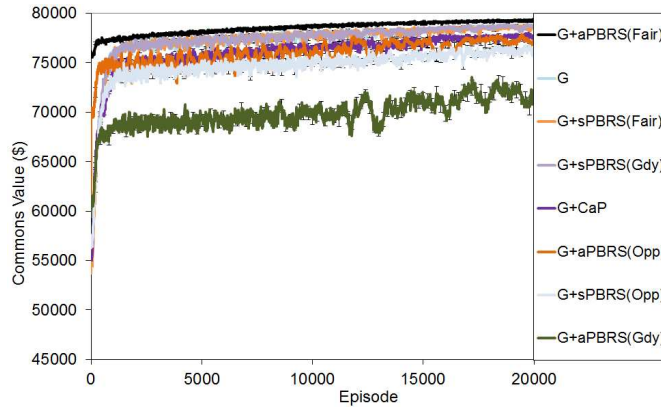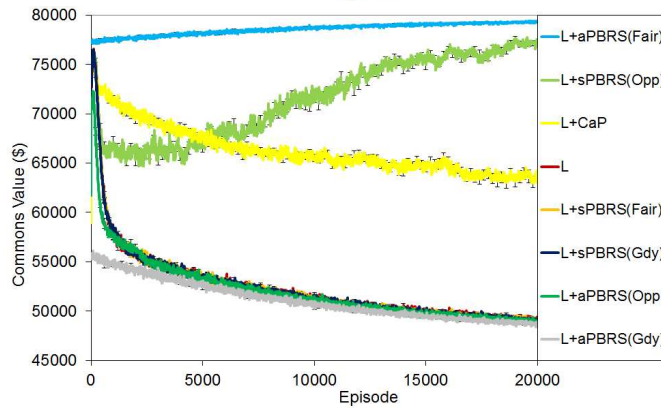Figure 2: Single-step Tragic Commons Domain results

(a) Commons values for L, G, and D

(b) Occupancy values for L, G, and D

(c) Commons values for G with heuristics

(d) Commons values for L with heuristics

Figure 3: Multi-step Tragic Commons Domain results

Table 1 Tragic Commons Domain results (Averaged over last 2000 episodes)

	$num\_timesteps = 1$		$num\_timesteps = 12$	
	Commons Value (\$)	ς	Commons Value (\$)	ς
Optimal Agents	80,000	80.00	80,000	80.00
$G + aPBRS(Fair)$	79,560	79.69	79,247	79.68
$L + aPBRS(Fair)$	79,559	79.69	79,272	79.70
G	79,367	81.14	78,657	82.79
$G + sPBRS(Fair)$	79,355	81.49	78,562	82.78
$G + sPBRS(Greedy)$	79,309	81.43	78,626	82.74
D	78,821	79.30	79,207	81.63
$G + aPBRS(Opportunistic)$	78,488	81.52	77,297	80.67
$G + CaP$	78,295	83.66	77,614	82.00
$L + CaP$	77,691	86.18	63,622	106.65
$G + sPBRS(Opportunistic)$	77,468	77.54	76,246	77.49
$L + sPBRS(Opportunistic)$	71,393	71.52	76,922	83.99
$G + aPBRS(Greedy)$	65,744	100.69	71,922	94.22
Random Agents	59,925	59.97	59,762	60.00
$L + sPBRS(Fair)$	52,456	116.53	49,238	118.86
L	51,617	117.18	49,282	118.82
$L + sPBRS(Greedy)$	50,630	117.98	49,210	118.88
$L + aPBRS(Greedy)$	49,198	119.10	48,842	119.11
$L + aPBRS(Opportunistic)$	49,172	119.11	49,085	118.97
Greedy Agents	48,000	120.00	48,000	120.00

D). This difference in final performance was found to be statistically significant ($p = 0.025$). D learned a better final policy than G in the multi-step version of the problem (99.0% of optimal performance for D, vs. 98.3% of optimal performance for G). D was found to offer statistically better performance than G in the multi-step TCD ($p = 1.41 \times 10^{-23}$).

Devlin et al. (2014) originally proposed CaP as a shaping heuristic for G; however we have also tested $L + CaP$ here in addition to the originally proposed $G + CaP$. $G + CaP$ does not perform quite as well as G in either version of this problem domain. We find that CaP is a good heuristic for shaping L, reaching 97.1% of the optimal commons value in the single-step TCD, and 79.5% of the optimal commons value in the multi-step TCD. $L + CaP$ offers a significant improvement over the Random baseline (and therefore unshaped L) in both the single-step ($p = 2.36 \times 10^{-57}$) and multi-step ($p = 1.47 \times 10^{-11}$) TCD.

The $aPBRS(Fair)$ heuristic performs exceptionally well when combined with both L and G, in all cases reaching over 99% of the optimal commons value of \$80,000. This is an extremely strong heuristic, that can guide even self-interested agents towards the optimal joint policy in the Tragic Commons Domain. $L + aPBRS(Fair)$ was found to offer statistically similar performance to $G + aPBRS(Fair)$ in the single-step version ($p = 0.94$), and statistically better performance than $G + aPBRS(Fair)$ in the multi-step ($p = 2.7 \times 10^{-6}$) version. The $sPBRS(Fair)$ shaping performs well when combined with G, although not quite as well as G without shaping. $L + sPBRS(Fair)$ performs quite similarly to L in both versions of the TCD. The aPBRS version of this heuristic is much more effective than the sPBRS version, as aPBRS specifies preferences for actions directly.

The $sPBRS(Oppor.)$ heuristic is useful when combined with L to encourage behaviour leading to a high system utility. $L + sPBRS(Oppor.)$ offers a significant improvement over the Random baseline (and therefore unshaped L) in both the single-step ($p = 3.69 \times 10^{-35}$) and multi-step ($p = 2.42 \times 10^{-66}$) TCD. $G + sPBRS(Oppor.)$, however, does not reach the same performance as unshaped G in either variant of the TCD. The action-based version of this heuristic performs

quite well when combined with G, although again not as well as the unshaped version of G. When combined with L, $aPBRS(Oppor.)$ is among the worst performing approaches tested. This is because all agents are encouraged to graze the maximum number of animals in the commons when it is below capacity using this shaping, resulting in a lower overall return. By contrast, the sPBRS version performs better, as only agents that choose to add extra animals when the commons is below capacity are given the shaping reward.

The final PBRS heuristic that we tested was designed to encourage greedy behaviour in the agents, where each agent seeks to graze the maximum allowed number of animals in the commons. When combined with L, both the sPBRS and aPBRS versions of the greedy heuristic reach a lower final level of performance than unshaped L. The aPBRS version of this heuristic learns more quickly, again as the preferences for actions are defined directly. When combined with G, the $sPBRS(Greedy)$ heuristic reduces the final performance by a small amount compared to unshaped G. $G + aPBRS(Greedy)$ results in much lower performance than unshaped G in both versions of the TCD. As G is a much better performing reward function than L, the agents learning using G manage to overcome the incorrect knowledge that is provided to them, and to converge to reasonably good final policies. By contrast, the greedy heuristic serves to accentuate the self-interested nature of agents learning using L, especially in the case of $L + aPBRS(Greedy)$.

4 Shepherd Problem Domain

4.1 Problem Description

In this section we introduce a new multi-agent congestion game, the Shepherd Problem Domain (SPD). In the SPD, the agents must coordinate their actions to maximise the social welfare or global utility of the system. The SPD models a system of interconnected pastures, where each shepherd (agent) begins in a certain pasture and must decide in which pasture it will graze its herd of animals. At each timestep each agent knows which pasture it is currently attending and has the choice to remain still, or to move with its herd to an adjacent pasture.

Once all agents have completed their selected actions they are rewarded. Each pasture has a certain capacity ψ, and the highest reward for a pasture is received when the number of herds (agents) present is equal to the capacity of the pasture. Lower rewards are received for pastures which are congested as there is less food available to each herd, which affects the health and marketable value of the animals. Pastures that are under capacity also receive lower rewards, which reflects lost utility due to under-utilisation of the resource. The local reward function (L) for a particular pasture is calculated as:

$$L(s,t) = x_t e^{\frac{-x_{s,t}}{\psi_s}} \tag{20}$$

where s is the pasture (state), $x_{s,t}$ is the number of herds (agents) present at that pasture at time t, and ψ_s is the capacity of the pasture. The global reward or capacity utility can then be calculated as the summation of $L(s,t)$ over all pastures in the SPD:

$$G(t) = \sum_{s \in S} L(s,t) \tag{21}$$

where S is the set of pastures in the SPD.

The difference reward (D) for an agent can be directly calculated by applying Equation 6. As an agent only influences the utility of the pasture it is currently attending at a particular timestep, D_i may be calculated as:

$$D_i(t) = L(s,t) - (x_{s,t} - 1)e^{\frac{-(x_{s,t}-1)}{\psi_s}} \tag{22}$$

where $x_{s,t}$ is the number of agents attending the same pasture s as agent i at timestep t. This evaluation is precisely equivalent to applying Equation 6 to Equation 21 directly. In a similar manner, the potential function for CaP may be calculated as:

Algorithm 1 Shepherd Problem Domain with $G + CaP$

 1: initialize Q-values: $\forall s, a | Q(s, a) = -1$
 2: initialize potentials: $\forall s | \Phi(s) = 0$
 3: **for** $episode = 1 \rightarrow num\_episodes$ **do**
 4:     **for** $timestep = 1 \rightarrow num\_timesteps$ **do**
 5:         **for** $i = 1 \rightarrow num\_agents$ **do**
 6:             sense current pasture s
 7:             choose action a, using ϵ-greedy
 8:             move agent to s'
 9:         **end for**
10:         **for** All pastures $s \in S$ **do**
11:             evaluate local reward $L(s)$ (Equation 20)
12:         **end for**
13:         evaluate global reward G (Equation 21)
14:         **for** $i = 1 \rightarrow total\_agents$ **do**
15:             set $\Phi(s')$ (Equation 23)
16:             set F (Equation 3)
17:             set $R' = G + F$ (Equation 2)
18:             update $Q(s, a)$ values (Equation 1)
19:         **end for**
20:         reduce ϵ by multiplication with $epsilon\_decay\_rate$
21:         reduce α by multiplication with $alpha\_decay\_rate$
22:     **end for**
23:     **for** $i = 1 \rightarrow num\_agents$ **do**
24:         choose action a, using ϵ-greedy
25:         move to absorbing state
26:         set $F = 0 - \Phi(s')$ (Equation 3)
27:         set $R' = 0 + F$ (Equation 2)
28:         update $Q(s', a)$ values (Equation 1)
29:     **end for**
30: **end for**

$$\Phi(s) = (x_{s,t} - 1)e^{\frac{-(x_{s,t}-1)}{\psi_s}} \tag{23}$$

Algorithm 1 provides additional details of the process followed in the SPD when agents are rewarded using $G + CaP$.

Other examples of congestion problems that have been studied thus far include the El-Farol Bar Problem (EBP) (Arthur, 1994), the Traffic Lane Domain (TLD) (Tumer et al., 2009) and the Beach Problem Domain (BPD) (Devlin et al., 2014). All these approaches address the need to find solutions that maximise social welfare in systems with scarce resources. This is an important goal, due to the fact that insights gained while studying these problems may be useful when tackling congestion that occurs in real world systems, e.g. in road, air or sea transportation applications.

The SPD builds upon the work presented in these previous approaches, while introducing additional complexity. In the SPD, we consider two dimensional arrangements of resources, whereas resources are arranged in a single dimension only in the TLD and BPD. The arrangement of resources in the EBP is irrelevant to the outcome, as it is a stateless single shot problem where movement between resources is not allowed. The increased number of possible resource configurations leads to a larger number of available actions in the SPD (we consider up to 5 actions in our experiments), compared to 3 available actions in the BPD and TLD (*move_left*, *stay_still*, *move_right*). This increases the difficulty of the coordination task for the agents and thus the SPD

0	1	2
3	4	5
6	7	8

Figure 4: SPD 3×3 grid topology with resource (state) numbers

allows for a more thorough evaluation of MARL approaches to tackling resource management problems. In this study, we consider a group of $|S| = 9$ resources in a 3×3 grid arrangement. The 3×3 grid topology that we have used in our experiments is shown in Fig. 4. Therefore the actions available to the agents at each timestep are: *stay_still*, *move_up*, *move_right*, *move_down*, *move_left*. Actions which would move an agent outside the resource group leave its position unchanged. We set the number of agents $N = 100$, and we assume that all resources have the same capacity $\psi = 4$, and that $N/4$ agents start in each of the pastures 1,3,5,7. The difficulty of this coordination problem could be increased further by assigning a random initial starting state to each agent at the beginning of each episode.

4.2 PBRS Heuristics

To explore the effectiveness of PBRS in this domain, we apply four different manual heuristics in addition to the automated CaP heuristic. These four heuristics are based on those proposed by Devlin et al. (2014) for the BPD, which we have adapted for the SPD. The work of Devlin et al. (2014) is the only other study to date on the application of PBRS to congestion games, and we have included modified versions of these heuristics to test their effectiveness in congestion games with more complex resource configurations. Furthermore, Devlin et al. (2014) expressed their heuristics in state-based forms only; therefore our work is the first empirical evaluation of aPBRS in congestion games. In the aPBRS heuristics below, the function $next\_state(s, a)$ returns the state that will be reached if an agent selects action a while in state s. The four heuristics that we have tested are:

- **Overcrowd One:** In the SPD, when the number of agents N is much greater than the combined capacity of the resources, the optimal solution is for $N - (|S| - 1) \times \psi$ agents to converge to a single pasture, leaving ψ agents on each of the remaining pastures. This minimises the effect of congestion on the system, and gives the highest possible social welfare ($G = 11.77$). For the 3×3 grid instance of this domain with $|S| = 9$ pastures of capacity $\psi = 4$, this means that 68 agents must overcrowd one of the pastures so that the others remain uncongested. Overcrowd One promotes this behaviour, encouraging most of the agents to congest the middle pasture ($s = 4$), and is therefore an extremely strong heuristic. It is important to note that this heuristic is manually tailored to individual agents. We expect this heuristic to achieve near-optimal performance, demonstrating the effectiveness of PBRS

when precise information is available about the optimal joint policy.

$$\Phi(s) = \begin{cases} 10 & \text{if } s = 0 \text{ and } agent\_id \in [0, \psi - 1] \\ 10 & \text{if } s = 1 \text{ and } agent\_id \in [\psi, 2\psi - 1] \\ 10 & \text{if } s = 2 \text{ and } agent\_id \in [2\psi, 3\psi - 1] \\ 10 & \text{if } s = 4 \text{ and } agent\_id \in [3\psi, N/2 - \psi - 1] \\ 10 & \text{if } s = 3 \text{ and } agent\_id \in [N/2 - \psi, N/2 - 1] \\ 10 & \text{if } s = 5 \text{ and } agent\_id \in [N/2, N/2 + \psi - 1] \\ 10 & \text{if } s = 4 \text{ and } agent\_id \in [N/2 + \psi, N - 3\psi - 1] \\ 10 & \text{if } s = 6 \text{ and } agent\_id \in [N - 3\psi, N - 2\psi - 1] \\ 10 & \text{if } s = 7 \text{ and } agent\_id \in [N - 2\psi, N - \psi - 1] \\ 10 & \text{if } s = 8 \text{ and } agent\_id \in [N - \psi, N - 1] \\ 0 & \text{otherwise} \end{cases} \tag{24}$$

$$\Phi(s, a) = \begin{cases} 10 & \text{if } next\_state(s, a) = 0 \text{ and } agent\_id \in [0, \psi - 1] \\ 10 & \text{if } next\_state(s, a) = 1 \text{ and } agent\_id \in [\psi, 2\psi - 1] \\ 10 & \text{if } next\_state(s, a) = 2 \text{ and } agent\_id \in [2\psi, 3\psi - 1] \\ 10 & \text{if } next\_state(s, a) = 4 \text{ and } agent\_id \in [3\psi, N/2 - \psi - 1] \\ 10 & \text{if } next\_state(s, a) = 3 \text{ and } agent\_id \in [N/2 - \psi, N/2 - 1] \\ 10 & \text{if } next\_state(s, a) = 5 \text{ and } agent\_id \in [N/2, N/2 + \psi - 1] \\ 10 & \text{if } next\_state(s, a) = 4 \text{ and } agent\_id \in [N/2 + \psi, N - 3\psi - 1] \\ 10 & \text{if } next\_state(s, a) = 6 \text{ and } agent\_id \in [N - 3\psi, N - 2\psi - 1] \\ 10 & \text{if } next\_state(s, a) = 7 \text{ and } agent\_id \in [N - 2\psi, N - \psi - 1] \\ 10 & \text{if } next\_state(s, a) = 8 \text{ and } agent\_id \in [N - \psi, N - 1] \\ 0 & \text{otherwise} \end{cases} \tag{25}$$

- **Middle:** This heuristic incorporates some knowledge about the optimal policy i.e. the idea that one resource should be "sacrificed" or congested for the greater good of the system. However, in contrast to Overcrowd One this shaping is not tailored to individual agents, as all agents receive PBRS based on the same potential function. Thus, this heuristic encourages all agents to go to the middle pasture ($s = 4$). We expect that this shaping will improve both the performance and learning speed of agents receiving PBRS, and will demonstrate the effect of PBRS when useful but incomplete domain knowledge is available.

$$\Phi(s) = \begin{cases} 10 & \text{if } s = 4 \\ 0 & \text{otherwise} \end{cases} \tag{26}$$

$$\Phi(s, a) = \begin{cases} 10 & \text{if } next\_state(s, a) = 4 \\ 0 & \text{otherwise} \end{cases} \tag{27}$$

- **Spread:** The Spread heuristic encourages agents to distribute themselves evenly across the pastures in the SPD. This is an example of a weak heuristic, and demonstrates the effect of PBRS in cases where very little useful domain knowledge is available. Therefore, we expect agents receiving this shaping to show modest if any improvements in learning speed and final

performance.

$$
\Phi(s) = \begin{cases}
10 & \text{if } s = 0 \text{ and } agent\_id \in [0, N/|S| - 1] \\
10 & \text{if } s = 1 \text{ and } agent\_id \in [N/|S|, 2N/|S| - 1] \\
10 & \text{if } s = 2 \text{ and } agent\_id \in [2N/|S|, 3N/|S| - 1] \\
10 & \text{if } s = 3 \text{ and } agent\_id \in [3N/|S|, 4N/|S| - 1] \\
10 & \text{if } s = 4 \text{ and } agent\_id \in [4N/|S|, 5N/|S| - 1] \\
10 & \text{if } s = 5 \text{ and } agent\_id \in [5N/|S|, 6N/|S| - 1] \\
10 & \text{if } s = 6 \text{ and } agent\_id \in [6N/|S|, 7N/|S| - 1] \\
10 & \text{if } s = 7 \text{ and } agent\_id \in [7N/|S|, 8N/|S| - 1] \\
10 & \text{if } s = 8 \text{ and } agent\_id \in [8N/|S|, N - 1] \\
0 & \text{otherwise}
\end{cases}
\tag{28}
$$

$$
\Phi(s, a) = \begin{cases}
10 & \text{if } next\_state(s, a) = 0 \text{ and } agent\_id \in [0, N/|S| - 1] \\
10 & \text{if } next\_state(s, a) = 1 \text{ and } agent\_id \in [N/|S|, 2N/|S| - 1] \\
10 & \text{if } next\_state(s, a) = 2 \text{ and } agent\_id \in [2N/|S|, 3N/|S| - 1] \\
10 & \text{if } next\_state(s, a) = 3 \text{ and } agent\_id \in [3N/|S|, 4N/|S| - 1] \\
10 & \text{if } next\_state(s, a) = 4 \text{ and } agent\_id \in [4N/|S|, 5N/|S| - 1] \\
10 & \text{if } next\_state(s, a) = 5 \text{ and } agent\_id \in [5N/|S|, 6N/|S| - 1] \\
10 & \text{if } next\_state(s, a) = 6 \text{ and } agent\_id \in [6N/|S|, 7N/|S| - 1] \\
10 & \text{if } next\_state(s, a) = 7 \text{ and } agent\_id \in [7N/|S|, 8N/|S| - 1] \\
10 & \text{if } next\_state(s, a) = 8 \text{ and } agent\_id \in [8N/|S|, N - 1] \\
0 & \text{otherwise}
\end{cases}
\tag{29}
$$

- **Overcrowd All:** This heuristic encourages agents to go to slightly overcrowded pastures, and is an instance of dynamic PBRS (Devlin and Kudenko, 2012), as potentials of states or state action pairs may change over time. We expect this heuristic to perform poorly, and to serve as an example of the effect of PBRS when misleading information is used to design potential functions.

$$
\Phi(s) = \begin{cases}
10 & \text{if } \psi < x_s < 2\psi \\
0 & \text{otherwise}
\end{cases}
\tag{30}
$$

$$
\Phi(s, a) = \begin{cases}
10 & \text{if } \psi < x_{next\_state(s,a)} < 2\psi \\
0 & \text{otherwise}
\end{cases}
\tag{31}
$$

4.3 Experimental Procedure

For all experiments, we plot the value of the global reward G (referred to as the capacity utility) against the number of completed learning episodes. All plots include error bars representative of the standard error of the mean based on 50 statistical runs. Specifically, we calculate the error as $\sigma/\sqrt{n}$ where σ is the standard deviation and n is the number of statistical runs. Error bars are included on all plots at 1000 episode intervals. The plots show the average performance across the 50 statistical runs that were conducted at 10 episode intervals. All claims of statistical significance are supported by two-tailed t-tests assuming unequal variances, with $p = 0.05$ selected as the threshold for significance. In all experiments, the number of episodes is set to $num\_episodes = 10000$, the number of timesteps is set to $num\_timesteps = 1$, the learning rate is set to $\alpha = 0.1$ with $alpha\_decay\_rate = 0.9999$, the exploration rate is set to $\epsilon = 0.05$ with $epsilon\_decay\_rate = 0.9999$ and the discount factor is set to $\gamma = 0.9$. These parameter values were chosen to match those used by Devlin et al. (2014) for the Beach Problem Domain.

4.4   Experimental Results

Figs. 5a to 5d plot the learning curves for capacity utility for the approaches tested, while Table 2 gives the average capacity utility for each approach over the last 1000 episodes. In all plots the maximum possible capacity utility of 11.77 is marked with a black dashed line. In Table 2 we have also included several other results to serve as comparisons to the MARL approaches tested; these are: Optimum, Random Agents, Even Spread, Initial Distribution and Worst. The Optimum figure shows the maximum possible capacity utility for the system, while Random Agents gives the average performance of 100 agents that select actions from a uniform random distribution in the SPD, averaged over 50 runs. Even Spread gives the capacity utility for the SPD if agents are evenly distributed among the pastures, and Initial Distribution gives the capacity utility for the starting conditions with $N/4$ agents in each of the pastures 1, 3, 5, 7. Finally, Worst gives the system utility if all 100 agents overcrowd any one of the 9 pastures.

The effectiveness of the approaches tested may be assessed by comparison with the performance of the Random Agents baseline. G significantly outperforms the Random Agents ($p = 4.43 \times 10^{-11}$), whereas $L + aPBRS(OvercrowdAll)$ fails to outperform the Random Agents ($p = 1.1 \times 10^{-5}$). Thus all approaches that perform better than G are also statistically better than choosing actions at random, whereas approaches that perform worse than $L + aPBRS(OvercrowdAll)$ are in fact worse than selecting actions at random, and therefore demonstrate extremely poor performance in the SPD.

Fig. 5a plots the performance of the typical MARL credit assignment structures L, G, and D, along with $G + CaP$ and $L + CaP$. It is immediately apparent that all of these approaches fall far short of achieving the maximum possible performance in the SPD. D offers the best performance of these approaches, reaching a capacity utility of 9.68 or 82% of the maximum possible value. The CaP heuristic again proves to be useful in this problem domain, with $G + CaP$ offering a statistically significant improvement in final performance compared to G ($p = 8.33 \times 10^{-9}$), and $L + CaP$ offering a statistically significant improvement in final performance compared to L ($p = 1.6 \times 10^{-37}$).

Fig. 5b plots learning curves for the 10 reward functions with the best final performance. As we saw in the TCD study, aPBRS when combined with good heuristic information (Overcrowd One in this case) is again the best performer. This is of course to be expected, as Overcrowd One is an extremely strong heuristic that provides knowledge about the optimal policy for the SPD, and demonstrates again the effectiveness of aPBRS when good domain knowledge is available. $G + sPBRS(Middle)$ is the next best performer, although its performance is statistically worse than $G + aPBRS(OvercrowdOne)$ ($p = 2.96 \times 10^{-27}$). Nevertheless, $G + sPBRS(Middle)$ still achieves a good level of performance, reaching a capacity utility of 10.50 or 89% percent of the optimal value. The Middle heuristic is however a more realistic evaluation of the effects of PBRS, as it is unlikely that precise information about the optimal policy (like that provided by Overcrowd One) will be available for every application domain. The performance of $G + sPBRS(Middle)$ demonstrates that shaping with PBRS is effective even with less than perfect domain knowledge.

$G + sPBRS(OvercrowdOne)$ performs statistically worse than $G + sPBRS(Middle)$ ($p = 3.48 \times 10^{-3}$); this is perhaps a surprising result, as the Overcrowd One heuristic provides more detailed information about how to behave on a per agent level. This can however be explained by the nature of shaping using sPBRS; agents receive shaping only upon reaching certain system states when implementing sPBRS. To exploit the knowledge provided by Overcrowd One fully, practically all agents would have to reach their encouraged position at the same time. It is interesting to note also that $G + sPBRS(Middle)$ initially performs worse than $G + sPBRS(OvercrowdOne)$; this is due to many of the agents initially being encouraged to converge on the middle state. $G + sPBRS(Middle)$ however quickly begins to outperform $G + sPBRS(OvercrowdOne)$, as some agents select actions at random that lead to different states and thus higher system utilities, while the majority of agents converge on the middle state.

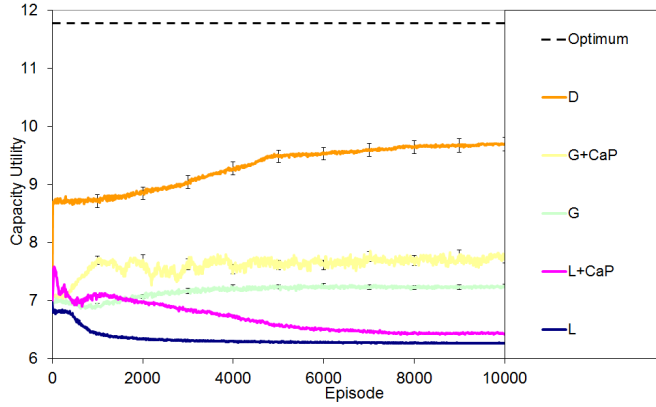**Table 2** Shepherd Problem Domain Results (Averaged over last 1000 episodes)

Reward Function	Capacity Utility
Optimum	11.77
$L + aPBRS(OvercrowdOne)$	11.74
$G + aPBRS(OvercrowdOne)$	11.73
$G + sPBRS(Middle)$	10.50
$G + sPBRS(OvercrowdOne)$	10.25
D	9.68
$G + sPBRS(Spread)$	8.31
$G + CaP$	7.71
G	7.23
Random Agents	6.95
$L + aPBRS(OvercrowdAll)$	6.72
$L + CaP$	6.44
$G + aPBRS(Spread)$	6.43
$L + aPBRS(Spread)$	6.32
$L + sPBRS(Middle)$	6.26
L	6.26
$L + sPBRS(OvercrowdOne)$	6.23
$L + sPBRS(Spread)$	6.23
Even Spread	6.22
$G + sPBRS(OvercrowdAll)$	6.22
$G + aPBRS(OvercrowdAll)$	5.06
$L + sPBRS(OvercrowdAll)$	4.91
$G + aPBRS(Middle)$	0.47
$L + aPBRS(Middle)$	0.45
Initial Distribution	0.19
Worst	1.39×10^{-9}

The performance of $G + sPBRS(OvercrowdOne)$ is statistically better than D ($p = 4.57 \times 10^{-5}$), and this demonstrates that for certain applications PBRS combined with good heuristic information is a suitable alternative to the now ubiquitous difference reward. Implementing PBRS with a manually specified heuristic for MARL also does not require precise knowledge of the mathematical form of the system evaluation function, whereas this is a requirement for D which could potentially be problematic for more complex application areas. Of course this shortcoming also affects the automated CaP PBRS heuristic, however recent work on estimating counterfactuals (Colby et al., 2016) shows promise in addressing this limitation.

5 Conclusion & Future Work

In this paper, we have explored the issue of appropriate credit assignment in stochastic resource management games. We introduced two new Stochastic Games to facilitate this research: the Tragic Commons Domain and the Shepherd Problem Domain. In the TCD we explored how best to avoid over-exploitation of a shared resource, while in the SPD our work focused on maximising the social welfare of a system in situations where congestion of a group of shared resources is inevitable. We applied MARL with various credit assignment structures to learn solutions to these problems, and our results show that systems learning using appropriate credit assignment techniques can achieve near optimal performance in stochastic resource management games, outperforming systems using unshaped local or global rewards.

An interesting finding from our work is that sPBRS and aPBRS variants using the same heuristic knowledge can encourage very different behaviours. This is the first empirical study

(a) Basic reward functions
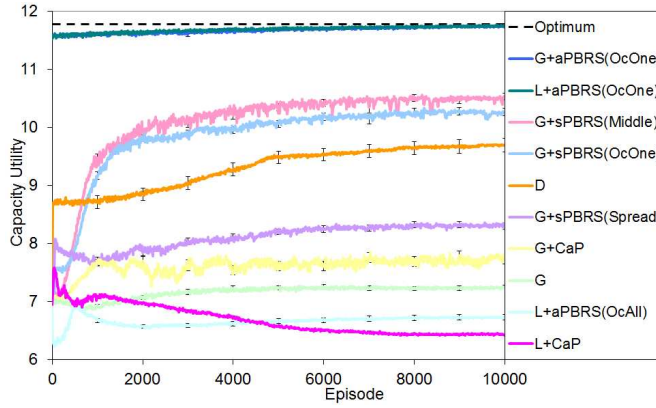
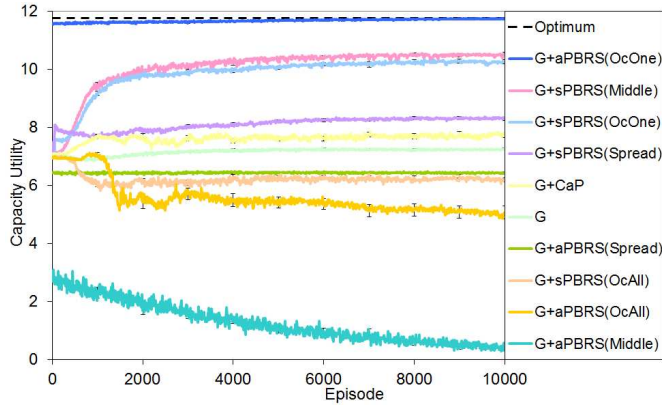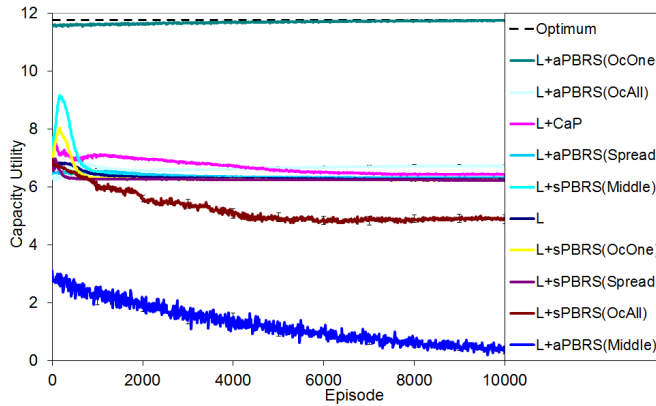(b) Best performing reward functions

(c) G with various heuristics

(d) L with various heuristics

Figure 5: Shepherd Problem Domain results

to investigate this effect, and also the largest MARL study of aPBRS in terms of the number of agents learning in the same environment (previous work by Devlin et al. (2011b) considered up to 9 agents, 4 of which implemented aPBRS). When precise knowledge about the optimal policy for a problem domain is available (e.g. the Fair Heuristic for the TCD and the Overcrowd One heuristic for the SPD), our results show that aPBRS offers the best method to exploit this knowledge. Having precise knowledge is of course the best case scenario, and many MARL problem domains exist where precise knowledge is not available. In cases where a misleading or suboptimal heuristic is used, we found that the aPBRS version generally performed more poorly than the sPBRS version. These effects can be explained by the fact that aPBRS specifies preferences for actions directly, whereas sPBRS provides shaping to agents only when certain system states are reached. Greedy action selection for aPBRS agents is biased by adding the value of the potential function to the Q(s,a) estimate (Eqn. 5). This modification influences an agent's exploration in a more direct manner than that of sPBRS, with the result that the effects of a good or bad heuristic on exploration are even more pronounced when using aPBRS instead of sPBRS.

It is also the case that partially correct knowledge can have very different effects when encoded using either sPBRS or aPBRS; the best example of this is the Middle heuristic used in the SPD. $G + sPBRS(Middle)$ was the third best performing reward function in the SPD, whereas $G + aPBRS(Middle)$ had close to the worst performance. When using sPBRS, only agents who reach the middle state are given the shaping reward, whereas with aPBRS all agents are encouraged to go to the middle state. This is the reason for the large difference in performance; good solutions to the SPD congest one of the resources, however if all agents adopt this behaviour it has a disastrous effect on the utility of the system. In summary, our results show that when implementing PBRS, it is worthwhile to test both sPBRS and aPBRS versions of the same heuristic information, as heuristics can have very different effects depending on which version of PBRS is used. Furthermore, we have demonstrated that aPBRS does scale well with larger numbers of agents when a suitable heuristic is used; Devlin et al. (2014) previously reported similar findings for sPBRS in a large scale study.

Several possibilities for further research are raised by the work presented in this paper. We intend to analyse other more complex resource management problems using the MAS paradigm. Examples that we are currently considering include management of fish stocks and water resources. In future work, we also plan extend the SPD by adding an additional objective which models the risk of predator attacks on agents' herds. In a subset of pastures in the SPD, predators may be present, and herds placed in these pastures are thus at risk of attack. A safety score that measures the risk of attack could be awarded as well as the capacity score from the standard SPD. This extension would provide a basis to study resource management problems where trade-offs between risk and reward must be considered. Recent work by Mannion et al. (2017) demonstrated that reward shaping can improve the quality of the trade-off solutions found in multi-objective multi-agent problem domains. Therefore, we expect that reward shaping will be a useful mechanism to discover good trade-offs between risk and reward in a multi-objective version of the SPD.

We also plan to investigate other larger and more complex arrangements of resources in the SPD than the 3×3 grid layout evaluated in this work. Other factors that could be investigated in future work on congestion games include the effect of the number of timesteps and the initial distribution of the agents on the performance of various credit assignment structures; these are issues that have not been investigated comprehensively in the literature to date. Finally, the effect of the scale of potential functions in MARL is an issue that should be addressed comprehensively in future work on PBRS. Work by Grześ and Kudenko (2009) examined the effect of scale in single-agent domains, and found that increasing the scale of the potential function used can improve performance when a good heuristic is used, but can damage performance when a poor

quality heuristic is used. We expect future studies will demonstrate that scaling can have similar effects in multi-agent domains.

While we have evaluated several multi-agent credit assignment structures in this study, there are other techniques that may also prove useful for solving resource management problems. In future work on stochastic resource management games, we plan to investigate the use of two recently proposed techniques: Difference Rewards incorporating Potential-Based Reward Shaping ($DRiP$) (Devlin et al., 2014) and Resource Abstraction (Malialis et al., 2016). We expect that the issue of appropriate credit assignment will become even more important as MARL is applied to more complex resource management problems, and techniques such as these offer a promising way to guide learning in complex MAS. It is inevitable that the precise mathematical form of the system evaluation function will not be known for some future resource management applications; therefore we expect that recent work on estimating counterfactuals (Colby et al., 2016; Mannion et al., 2016b) will also have an important role to play, as in these cases it will not be possible to calculate a traditional difference evaluation directly.

Acknowledgement

References

Arthur, W. B. Inductive reasoning and bounded rationality. *The American economic review*, pages 406–411, 1994.

Binmore, K. *Playing for Real: A Text on Game Theory*. Oxford University Press, 2012.

Buşoniu, L., Babuška, R., and Schutter, B. Multi-agent reinforcement learning: An overview. In Srinivasan, D. and Jain, L., editors, *Innovations in Multi-Agent Systems and Applications - 1*, volume 310 of *Studies in Computational Intelligence*, pages 183–221. Springer Berlin Heidelberg, 2010.

Colby, M., Duchow-Pressley, T., Chung, J. J., and Tumer, K. Local approximation of difference evaluation functions. In *Proceedings of the 15th International Conference on Autonomous Agents & Multiagent Systems (AAMAS)*, pages 521–529, 2016.

de Jong, S. and Tuyls, K. Learning to cooperate in a continuous tragedy of the commons. In *Proceedings of The 8th International Conference on Autonomous Agents and Multiagent Systems - Volume 2*, pages 1185–1186. International Foundation for Autonomous Agents and Multiagent Systems, 2009.

Devlin, S. and Kudenko, D. Theoretical considerations of potential-based reward shaping for multi-agent systems. In *Proceedings of the 10th International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, pages 225–232, 2011.

Devlin, S. and Kudenko, D. Dynamic potential-based reward shaping. In *Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, pages 433–440, 2012.

Devlin, S., Grzes, M., and Kudenko, D. An empirical study of potential-based reward shaping and advice in complex, multi-agent systems. *Advances in Complex Systems*, 14(2):251–278, 2011a.

Devlin, S., Grzes, M., and Kudenko, D. Multi-agent, potential-based reward shaping for robocup keepaway (extended abstract). In *Proceedings of the 10th International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, pages 1227–1228, 2011b.

Devlin, S., Yliniemi, L., Kudenko, D., and Tumer, K. Potential-based difference rewards for multiagent reinforcement learning. In *Proceedings of the 13th International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, pages 165–172, 2014.

Grześ, M. and Kudenko, D. Theoretical and empirical analysis of reward shaping in reinforcement learning. In *Machine Learning and Applications, 2009. ICMLA'09. International Conference on*, pages 337–344. IEEE, 2009.

Howley, E. and Duggan, J. Investing in the commons: A study of openness and the emergence of cooperation. *Advances in Complex Systems*, 14(02):229–250, 2011.

Malialis, K., Devlin, S., and Kudenko, D. Distributed reinforcement learning for adaptive and robust network intrusion response. *Connection Science*, 27(3):234–252, 2015.

Malialis, K., Devlin, S., and Kudenko, D. Resource abstraction for reinforcement learning in multiagent congestion problems. In *Proceedings of the 15th International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, pages 503–511, 2016.

Mannion, P., Duggan, J., and Howley, E. An experimental review of reinforcement learning algorithms for adaptive traffic signal control. In McCluskey, L. T., Kotsialos, A., Müller, P. J., Klügl, F., Rana, O., and Schumann, R., editors, *Autonomic Road Transport Support Systems*, pages 47–66. Springer International Publishing, 2016a.

Mannion, P., Duggan, J., and Howley, E. Generating multi-agent potential functions using counterfactual estimates. In *Proceedings of Learning, Inference and Control of Multi-Agent Systems (at NIPS 2016)*, December 2016b.

Mannion, P., Mason, K., Devlin, S., Duggan, J., and Howley, E. Dynamic economic emissions dispatch optimisation using multi-agent reinforcement learning. In *Proceedings of the Adaptive and Learning Agents workshop (at AAMAS 2016)*, May 2016c.

Mannion, P., Mason, K., Devlin, S., Duggan, J., and Howley, E. Multi-objective dynamic dispatch optimisation using multi-agent reinforcement learning. In *Proceedings of the 15th International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, pages 1345–1346, May 2016d.

Mannion, P., Duggan, J., and Howley, E. Analysing the effects of reward shaping in multi-objective stochastic games. In *Proceedings of the Adaptive and Learning Agents workshop (at AAMAS 2017)*, May 2017.

Mason, K., Mannion, P., Duggan, J., and Howley, E. Applying multi-agent reinforcement learning to watershed management. In *Proceedings of the Adaptive and Learning Agents workshop (at AAMAS 2016)*, May 2016.

Ng, A. Y., Harada, D., and Russell, S. J. Policy invariance under reward transformations: Theory and application to reward shaping. In *Proceedings of the Sixteenth International Conference on Machine Learning*, ICML '99, pages 278–287, San Francisco, CA, USA, 1999. Morgan Kaufmann Publishers Inc.

Puterman, M. L. *Markov Decision Processes: Discrete Stochastic Dynamic Programming.* John Wiley & Sons, Inc., New York, NY, USA, 1st edition, 1994.

Randløv, J. and Alstrøm, P. Learning to drive a bicycle using reinforcement learning and shaping. In *Proceedings of the Fifteenth International Conference on Machine Learning*, ICML '98, pages 463–471, San Francisco, CA, USA, 1998. Morgan Kaufmann Publishers Inc.

Shoham, Y., Powers, R., and Grenager, T. If multi-agent learning is the answer, what is the question? *Artificial Intelligence*, 171(7):365–377, 2007. ISSN 0004-3702.

Tumer, K. and Agogino, A. Distributed agent-based air traffic flow management. In *Proceedings of the 6th International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, pages 330–337, Honolulu, HI, May 2007.

Tumer, K., Agogino, A. K., and Welch, Z. Traffic congestion management as a learning agent coordination problem. In Bazzan, A. and Kluegl, F., editors, *Multiagent Architectures for Traffic and Transportation Engineering*, pages 261–279. Lecture notes in AI, Springer, 2009.

Watkins, C. J. and Dayan, P. Technical note: Q-learning. *Machine Learning*, 8(3-4):279–292, 1992. ISSN 0885-6125.

Watkins, C. J. C. H. *Learning from Delayed Rewards*. PhD thesis, King's College, Cambridge, UK, 1989.

Wiering, M. and van Otterlo, M., editors. *Reinforcement Learning: State-of-the-Art*. Springer, 2012.

Wiewiora, E., Cottrell, G., and Elkan, C. Principled methods for advising reinforcement learning agents. In *Proceedings of the Twentieth International Conference on Machine Learning*, pages 792–799, 2003.

Wolpert, D. H. and Tumer, K. Collective intelligence, data routing and braess' paradox. *Journal of Artificial Intelligence Research*, pages 359–387, 2002.

Wolpert, D. H., Wheeler, K. R., and Tumer, K. Collective intelligence for control of distributed dynamical systems. *EPL (Europhysics Letters)*, 49(6):708, 2000.

Wooldridge, M. *Introduction to Multiagent Systems*. John Wiley & Sons, Inc., New York, NY, USA, 2001.