Knowledge-Based Multi-Objective Multi-Agent Reinforcement Learning

A thesis submitted for the degree of Doctor of Philosophy

by

PATRICK MANNION BENG HDIPAPPSC

Discipline of Information Technology
College of Engineering & Informatics
National University of Ireland Galway

August 2017

Supervisors: Dr. Enda Howley
Dr. Jim Duggan

OÉ Gaillimh
NUI Galway

Abstract

Multi-Agent Reinforcement Learning (MARL) is a powerful Machine Learning paradigm, where multiple autonomous agents can learn to improve the performance of a system through experience. The majority of MARL implementations aim to optimise systems with respect to a single objective, despite the fact that many real world problems are inherently multi-objective in nature. Examples of multi-objective problems where MARL may be applied include water resource management, traffic signal control, electricity generator scheduling and robot coordination tasks. Compromises between conflicting objectives may be defined using the concept of Pareto dominance. The Pareto optimal or non-dominated set consists of solutions that are incomparable, where each solution in the set is not dominated by any of the others on every objective.

Reward shaping has been proposed as a means to address the credit assignment problem in single-objective MARL, however it has been shown to alter the intended goals of the domain if misused, leading to unintended behaviour. Potential-Based Reward Shaping ($PBRS$) and difference rewards (D) are commonly used shaping methods for MARL, both of which have been repeatedly shown to improve learning speed and the quality of joint policies learned by agents in single-objective problems. Research into multi-objective MARL is still in its infancy, and very few studies have dealt with the issue of credit assignment in this context.

This thesis explores the possibility of using reward shaping to improve agent coordination in multi-objective MARL domains. The implications of using either D or $PBRS$ are evaluated from a theoretical perspective, and the results of several empirical studies support the conclusion that these shaping techniques do not alter the true Pareto optimal solutions in multi-objective MARL domains. Therefore, the benefits of reward shaping can now be leveraged in a broader range of application domains, without the risk of altering the agents' intended goals.

Contents

List of Figures

List of Tables

Acknowledgements

Firstly, I would like to thank my supervisor Dr. Enda Howley. Enda, you an excellent mentor, educator and researcher, and above all a great friend. I look forward to many more years of fruitful collaboration with you in the future.

I also want to acknowledge the contributions made by Dr. Jim Duggan, who co-supervised this research, and was always available to offer advice and encouragement.

I am very grateful to Dr. Sam Devlin (University of York). Sam, thank you for the numerous interesting discussions we have had over the years, and for giving feedback on my work.

Throughout my studies I had many stimulating conversations about this research with friends located in Galway and further afield, each of which helped me to bring this work to a successful conclusion. In particular I would like to extend a special thanks to Mr. Ian Broderick for his attention to detail when proof reading this thesis, and to Mr. Karl Mason for kindly providing source code which was used in one of the experimental studies in this work.

On the day of my viva voce, I was very fortunate to have such a courteous and professional examination team; Prof. Karl Tuyls (Deepmind & University of Liverpool) as External Examiner, Dr. Enda Barrett as Internal Examiner and Prof. Michael Madden as Chair. Thank you all for the time and effort which you put into the examination process, and for your kind feedback afterwards.

My PhD studies were supported by two separate sources of funding, for which I am grateful. The first 12 months of this work were funded by a NUI Galway College of Engineering & Informatics Postgraduate Scholarship. The Irish Research Council also funded 28 months of this research through the Government of Ireland Postgraduate Scholarship Scheme (Project ID GOIPG/2014/174).

Last, but not least, I would like to thank my parents Dr. Joe & Mrs. Breeda Mannion for their unconditional love and support throughout the last 27 years. Without you, none of this would have been possible.

Declaration

This thesis has not previously been accepted in substance for any degree and is not being concurrently submitted in candidature for any degree other than Doctor of Philosophy of the National University of Ireland Galway. This thesis is the result of my own investigations, except where otherwise stated. Other sources are acknowledged by explicit references.

Dr. Sam Devlin (University of York) advised on the design of the experiments described in Sections 3.3, 5.1 and 5.3, and assisted with the theoretical analysis presented in Section 4.1.

Mr. Karl Mason (National University of Ireland Galway) provided part of the source code which was used for the Dynamic Economic Emissions Dispatch domain in Section 5.3.

Some of the material contained in this thesis has appeared in the following published or awaiting publication papers:

1. P. Mannion, S. Devlin, K. Mason, J. Duggan, and E. Howley. Policy Invariance under Reward Transformations for Multi-Objective Reinforcement Learning. Neurocomputing, 263, pp. 60-73, 2017.

2. P. Mannion, S. Devlin, J. Duggan, and E. Howley. Multi-Agent Credit Assignment in Stochastic Resource Management Games. The Knowledge Engineering Review, 32, pp. 1-21, 2017.

3. P. Mannion, J. Duggan, and E. Howley. A Theoretical and Empirical Analysis of Reward Transformations in Multi-Objective Stochastic Games. In: Proceedings of the 16th International Conference on Autonomous Agents and Multiagent Systems (AAMAS). May 2017, pp. 1625-1627.

4. P. Mannion, J. Duggan, and E. Howley. 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.

5. P. Mannion, S. Devlin, K. Mason, J. Duggan, and E. Howley. Potential-Based Reward Shaping Preserves Pareto Optimal Policies. In: Proceedings of the Adaptive and Learning Agents workshop (at AAMAS 2017). May 2017.

6. P. Mannion, J. Duggan, and E. Howley. Generating Multi-Agent Potential Functions using Counterfactual Estimates. In: Proceedings of Learning, Inference and Control of Multi-Agent Systems (at NIPS 2016). Dec. 2016.

7. P. Mannion, K. Mason, S. Devlin, J. Duggan, and E. Howley. Multi-Objective Dynamic Dispatch Optimisation using Multi-Agent Reinforcement Learning. In: Proceedings of the 15th International Conference on Autonomous Agents and Multiagent Systems (AAMAS). May 2016, pp. 1345-1346.

8. P. Mannion, K. Mason, S. Devlin, J. Duggan, and E. Howley. Dynamic Economic Emissions Dispatch Optimisation using Multi-Agent Reinforcement Learning. In: Proceedings of the Adaptive and Learning Agents workshop (at AAMAS 2016). May 2016.

9. P. Mannion, J. Duggan, and E. Howley. An Experimental Review of Reinforcement Learning Algorithms for Adaptive Traffic Signal Control. In: Autonomic Road Transport Support Systems. Ed. by L. T. McCluskey, A. Kotsialos, P. J. Muller, F. Klugl, O. Rana, and R. Schumann. Springer International Publishing, 2016, pp. 47-66.

10. P. Mannion, J. Duggan, and E. Howley. Learning Traffic Signal Control with Advice. In: Proceedings of the Adaptive and Learning Agents workshop (at AAMAS 2015). May 2015.

CHAPTER 1

Introduction and Motivation

Machine Learning is a process whereby a computer program learns from experience to improve its performance at a specified task (Mitchell 1997). Reinforcement Learning (RL) is a Machine Learning paradigm, where an autonomous agent learns to improve its performance at an assigned task by interacting with its environment. For each state experienced, a RL agent chooses an action and receives a reward from its environment based on the utility of its decision. Gradually, a RL agent can increase its long-term reward by exploiting knowledge learned about the expected utility of different state-action pairs.

RL is becoming increasingly common as a method to develop joint policies for cooperative Multi-Agent Systems (MAS). In a cooperative MAS multiple agents are deployed into a common environment, and must coordinate their actions to maximise the utility of the system. Multi-Agent Reinforcement Learning (MARL) has been successfully applied to a wide range of complex problem domains, including air traffic control (Tumer & Agogino 2007), data routing in networks (Wolpert & Tumer 2002), and RoboCup soccer (Devlin et al. 2011b). Two common assumptions in both single- and multi-agent RL are that agents learn without the benefit of any prior knowledge of how to behave in an application domain, and that the rewards received from an environment are scalar i.e. there is only one objective that agents must consider.

Typically, the system designer will have some degree of heuristic knowledge about how a problem may be solved, so the assumption that RL agents must learn without any prior knowledge is often unnecessary (Devlin 2013). After all, knowledge of the problem domain is required for the system designer to select the features necessary for a RL agent to function i.e. the state representation, available actions and reward function, so it is not unreasonable to expect that

the designer will also have some intuition about how an agent should act. Knowledge-Based Reinforcement Learning techniques seek to guide agents' exploration of their environments using prior knowledge, with the goal of improving learning speed and/or final performance. Reward shaping is one such method. The basic premise of reward shaping is to add an additional shaping reward to the reward naturally received from the environment. Potential-Based Reward Shaping ($PBRS$) and difference rewards (D) are two commonly used reward shaping techniques, both of which have had numerous successful applications in RL domains.

The majority of RL implementations aim to optimise systems with respect to a single objective, despite the fact that many real world problems are inherently multi-objective in nature. Multi-Objective Reinforcement Learning (MORL) encompasses a broad family of techniques which seek to address this deficit, and consider compromises between competing objectives which are defined using the concept of Pareto dominance (Pareto 1971). The Pareto optimal or non-dominated set consists of solutions that are incomparable, where each solution in the set is not dominated by any of the others on every objective. Examples of multi-objective problems where RL may be applied include water resource management (Mason et al. 2016), traffic signal control (Khamis & Gomaa 2014), electricity generator scheduling (Basu 2008), supply chain management (Duggan 2008) and robot coordination tasks (Yliniemi & Tumer 2016).

This thesis focuses on the intersection between these emerging topics in RL. Specifically, I wish to investigate whether reward shaping techniques can improve agent coordination in cooprative Multi-Objective Multi-Agent Reinforcement Learning (MOMARL) domains, in order to increase learning speed and performance. However, it is important that any proposed modifications to reward functions in MORL domains are theoretically sound; applying reward shaping in a careless manner has previously been demonstrated to alter an agent's original goals (Randløv & Alstrøm 1998). The combination of reward shaping and MORL is an exciting one, which has the potential to make RL a viable solution for an even broader range of complex application domains.

1.1 Research Questions

This work aims to answer the following research questions:

1. To what extent can reward shaping be applied to MORL in a theoretically sound manner, without the risk of altering an agent's intended goals? (RQ1)

2. Will existing reward shaping methods improve performance and/or learning speed in MORL domains? (RQ2)

3. Can a benchmark problem be established for MOMARL where the true Pareto optimal solutions are known? (RQ3)

1.2 Hypotheses

Following from the research questions above, during my investigations I expect to demonstrate that:

1. It can be theoretically proven that principled reward shaping techniques (i.e. $PBRS$ and D) will not alter agents' originally intended goals in MORL domains.

2. In single-agent MORL domains, $PBRS$ can improve an agent's learning speed compared to that of an unshaped agent, but will not alter the set of Pareto optimal policies learned upon convergence. In multi-agent MORL domains, both $PBRS$ and D can improve learning speed and converged performance when compared to agents learning without reward shaping.

3. It is possible to design a MOMARL benchmark problem with a finite number of discrete states and actions, for which all true Pareto optimal system utilities may be computed.

1.3 Scope

Although many Knowledge-Based Reinforcement Learning methods exist which may be suitable for improving MORL performance, the focus of this thesis is solely on reward shaping. Other Knowledge-Based RL methods are discussed in Chapter 2, along with the reasons why reward shaping was identified as the most promising candidate.

A single-policy algorithm (Q-learning) along with various scalarisation functions is employed in the MORL empirical studies in this thesis. As the focus of this work is on the design of reward functions for MORL, rather than the development of new MORL algorithms, the simplest and most common MORL algorithm in the literature was used. An overview of other MORL specific algorithms is provided in Chapter 2.

Most of the empirical work in this thesis is conducted in domains with multiple agents, rather than a single agent acting alone. There is already a large body of work on single-agent MORL; this is not the case for multi-agent MORL. Multi-agent domains are also arguably a more interesting topic of study, given that computing good policies becomes more difficult when there are multiple agents simultaneously learning to adapt their behaviour. This additional challenge is not present in Single-Agent Reinforcement Learning (SARL) domains.

Agents in a MAS may be cooperative, competitive, or may exhibit some mixture of these behaviours. Throughout this thesis, empirical and theoretical comparisons are drawn between D and $PBRS$. D is only applicable to cooperative MAS; therefore, only fully cooperative multi-agent domains are considered in this work. Furthermore, many complex systems (including all of the applications mentioned at the beginning of this chapter apart from RoboCup) may be modelled as fully cooperative MAS, meaning that this paradigm is directly applicable to the real-world.

Finally, the theoretical and empirical investigations in this thesis are confined to domains with discrete state and action spaces. Prior to this investigation, very little was understood about the effect of reward shaping in MORL. Therefore, it was necessary to test these methods initially on relatively simple discrete domains, to be certain that any observed effects were the result of the reward shaping techniques applied, rather than the choice of state representation or the value function approximation method used. Neural Networks have long been popular as a method of value function approximation for RL, and since this work began Deep Reinforcement Learning has matured significantly. Reward shaping in combination with Deep RL methods is a promising direction for future research, as I will discuss in Chapter 6.

1.4 Thesis Overview

The next chapter of this thesis introduces the terminology and concepts that these investigations build upon, covering all necessary material to make this work accessible to non-specialist readers.

Afterwards, Chapter 3 will explore the question of how to design useful potential functions when implementing $PBRS$. This chapter contributes three separate empirical studies which add to the existing body of work that justifies the use of $PBRS$ to improve the performance of Reinforcement Learning agents.

Chapter 4 considers the theoretical implications of applying reward shaping in single- and multi-agent MORL domains. Building upon previous guarantees, the theoretical analysis in this chapter will prove that $PBRS$ does not alter the set of Pareto optimal policies when it is applied to MORL domains. A separate analysis will demonstrate that D preserves the relative ordering of expected rewards in cooperative Multi-Objective Stochastic Games, which leads to the conclusion that the Pareto relation between actions is invariant when agents are rewarded with D instead of the system evaluation function G.

Finally, Chapter 5 presents a collection of empirical studies in both single- and multi-agent MORL domains. These studies demonstrate that both D and $PBRS$ can improve MOMARL performance, while also contributing the first empirical results showing that these techniques can guide agents towards true Pareto optimal solutions in MOMARL domains.

Chapter 6 concludes with a summary of the main contributions of this thesis, a discussion of the limitations of this work and an outline of some promising directions for future research.

CHAPTER 2

Background and Literature Review

This chapter introduces and explains the concepts that are required to understand the investigations presented later in this thesis. No prior knowledge is assumed on the part of the reader, apart from standard mathematical notation, a grasp of basic algebra, and some general knowledge of computing principles. Readers who are already familiar with standard model-free RL may wish to skip Sections 2.1 to 2.3, and begin instead at Section 2.4 where the first significant deviation from the usual paradigm is introduced.

Section 2.1 introduces the concepts of agency and Multi-Agent Systems. Section 2.2 deals with Reinforcement Learning in a single-agent context, which is formalised using Markov Decision Processes. Section 2.3 provides a definition of Multi-Agent Reinforcement Learning problems using the framework of Stochastic Games. In Section 2.4, the concepts discussed in the previous sections are extended to domains with multiple competing objectives.

Section 2.5 provides an overview of the broad spectrum of techniques that have been developed to incorporate prior knowledge into Reinforcement Learning agents. Section 2.6 introduces the fundamental theories underpinning Potential-Based Reward Shaping, while Section 2.7 discusses how these theories have been applied in the literature. Afterwards, in Section 2.8 the concept of the difference reward is introduced, along with a summary of its applications in cooperative Multi-Agent Systems. Section 2.9 discusses prior work that is both multi-objective and multi-agent in nature.

Finally, Section 2.10 provides a summary of the most important concepts introduced in this chapter, and identifies the gaps in the current literature that this thesis is intended to address.

2.1 Autonomous Agents & Multi-Agent Systems

The first and most essential concept which I wish to introduce is that of an agent; the notion of agency features prominently in Artificial Intelligence (AI) research. So what does the term "agent" actually mean?

Russell & Norvig (2009) define an agent as "anything that can be viewed as perceiving its environment through sensors and acting upon that environment through actuators". This definition is quite broad, and could include any control system (e.g. the much used "thermostat example").

Wooldridge (2001) provides a somewhat different definition: "An agent is a computer system that is situated in some environment, and that is capable of autonomous action in this environment in order to meet its design objectives."

In reality there is no single universally accepted definition of an agent, and it is quite likely that different researchers will have somewhat different ideas of what constitutes an agent. Some common attributes and abilities of an agent include:

- **Perception**: An agent perceives the state of its environment using sensors (Russell & Norvig 2009).
- **Actuation**: An agent uses its actuators to act upon its environment (Russell & Norvig 2009).
- **Rationality**: A rational agent selects actions that are expected to maximise its performance measure (Russell & Norvig 2009).
- **Autonomy**: An autonomous agent is capable of making decisions and responding to changes in its environment (Wooldridge 2001).
- **Goal-oriented**: Agents are able to exhibit goal-oriented behaviour in order to satisfy their design objectives (Wooldridge 2001).

Arguably, an agent could be hand-coded to satisfy its design objectives, although this process could become very time consuming for non-trivial applications. The ability to learn from past experiences can mitigate against this difficulty by allowing an agent to improve its own behaviour over time, thereby reducing the effort required on the part of the system designer. In my opinion, learning is an essential capability for a truly intelligent agent.

In many larger problem domains, single-agent solutions are not feasible (or desirable), due to concerns regarding scalability and/or robustness. A Multi-Agent System (MAS) features multiple agents deployed into a common environment. This is an inherently distributed paradigm, which benefits from scalability (agents can be added as required) and fault tolerance (the failure of any one agent does not imply failure of the whole system). The agents within a MAS may act cooperatively, competitively, or may exhibit a mixture of these behaviours (Wooldridge 2001).

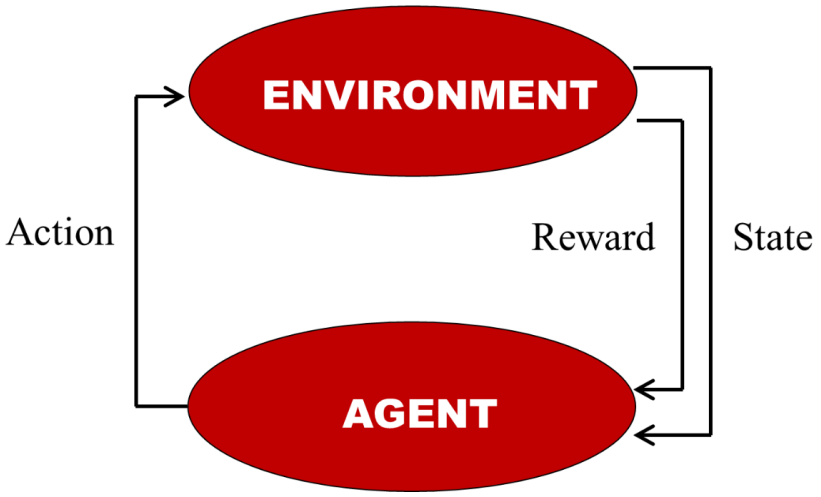


Figure 2.1: Interaction between an agent and its environment

2.2 Reinforcement Learning

Machine Learning is a process whereby a computer program learns from experience to improve its performance at a specified task (Mitchell 1997). 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. Agents receive 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 sequential decision making problems involving a single RL agent (Wiering & 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 $r = R(s, a, s')$. This process is illustrated in Fig. 2.1 above.

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 an 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 & van Otterlo 2012). The optimal policy for a MDP is denoted π*.

Designing an appropriate reward function for the environment is important, as a RL agent will attempt to maximise the return from this function, which will determine the policy learned. The value function V gives the expected discounted return for following policy π from state s onwards:

$$V^{\pi}(s) = E^{\pi}\left\{ \sum_{k=0}^{\infty} \gamma^{k} r_{t+k+1} \mid s_t = s \right\} \tag{2.1}$$

where $\gamma \in [0, 1]$ is the discount factor, which controls how an agent regards future rewards. Low values of γ encourage mypoic behaviour where an agent will aim to maximise short term rewards, whereas high values of γ cause agents to be more forward-looking and to maximise rewards over a longer timeframe.

The horizon H refers to the number of timesteps in a RL problem. In infinite-horizon problems $H = \infty$, whereas in episodic domains H has a finite value. Episodic domains may terminate after a fixed number of timesteps, or when an agent reaches a specified goal state. The last state reached in an episodic domain is referred to as the terminal state.

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 the model-free approach 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 utility of each state-action pair, which aid an agent in deciding which action is most desirable to select when in a certain state.

A RL 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 which are commonly used to manage the exploration exploitation trade-off are ϵ-greedy and softmax (Wiering & van Otterlo 2012). The ϵ-greedy strategy selects the action with the highest expected value with probability $1 - \epsilon$, or a randomly selected action with the remaining probability ϵ.

Q-learning (Watkins & Dayan 1992) is one of the most commonly used RL algorithms. It is a model-free algorithm that has been shown to converge to the optimum action-values for a MDP with probability 1, so long as all actions in all states are sampled infinitely often and the action-values are represented discretely (Watkins & Dayan 1992). In practice, Q-learning will learn good policies provided a sufficient number of samples are obtained for each state-action pair. Agents implementing Q-learning update their Q values according to the update rule in Eqn. 4.11 below:

$$Q(s, a) \leftarrow Q(s, a) + \alpha[r + \gamma \max_{a'} Q(s', a') - Q(s, a)] \tag{2.2}$$

where $Q(s, a)$ is an estimate of the utility of selecting action a in state s, $\alpha \in [0, 1]$ is the learning rate which controls the degree to which Q values are updated at each timestep, and $\gamma \in [0, 1]$ is the same discount factor used in Eqn. 2.1 above.

Q values may be initialised in a number of different ways. The simplest method is to set the values for all state-action pairs to zero i.e. $\forall s, a | Q(s, a) = 0$. Value function estimates may also be initialised using random values, optimistic values or pessimistic values. Optimistic initialisation sets the value for each state-action pair to the maximum possible reward; conversely, pessimistic initialisation sets the value for each state-action pair to the minimum possible reward.

Other methods of value function initialisation may be used to speed up learning, as will be discussed in Section 2.5. The theoretical guarantees of Q-learning hold with any arbitrary initial Q values; therefore the optimal policy for a MDP can be learned with any initial value function estimates.

Tabular representations are the simplest way to store value function estimates, where each state-action pair has a discrete Q value associated with it. When Q values are represented discretely, each additional feature tracked in the state leads to an exponential growth in the number of state-action pair values that must be stored (Sutton & Barto 1998). This problem is commonly referred to in the literature as the "curse of dimensionality", a term originally coined by Bellman (1957). In simple environments this is rarely an issue, but it may lead to an intractable problem in real-world applications, due to memory or computational constraints. Learning over a large state-action space is possible, but may take an unacceptably long time to learn useful policies.

Alternatively, function approximation may be used to generalise across states and/or actions, whereby a Q function is used to store and retrieve estimates of the utility of state-action pairs. Function approximation therefore offers a way to mitigate against the state-action space explosion, and is an active area of research in RL. Tile coding is one of the simplest forms of function approximation, where one tile represents multiple states or state-action pairs (Sutton & Barto 1998).

Neural Neworks are also commonly used to implement Q functions, one of the most famous examples being Tesuaro's application of RL to backgammon (Tesauro 1994). Recent work has applied Deep Neural Networks as a function approximation method; this emerging paradigm is known as Deep Reinforcement Learning. Deep RL has achieved human level performance (or above) on complex tasks such as playing Atari games (Mnih et al. 2015) and playing the board game Go (Silver et al. 2016).

Tabular representations are used exclusively throughout this thesis for a number of reasons, including the fact that Q values must be represented discretely to preserve the theoretical guarantees offered by Q-learning (Watkins & Dayan 1992). Although reward shaping could be applied in cases where function approximation is used, its use presents difficulties in relation to developing theoretical guarantees for the techniques considered (Potential-based Reward Shaping and difference rewards).

One of the primary aims of this thesis is to provide theoretical justification and empirical evidence supporting the use of reward shaping in multi-objective domains, especially concerning the effect of reward shaping on the Pareto optimal solutions of a domain, which the use of

function approximation would preclude. Nevertheless, reward shaping is a promising candidate to improve the performance of Deep RL agents, as I will discuss in the conclusion of this thesis (Chapter 6).

2.3 Multi-Agent Reinforcement Learning

In Multi-Agent Reinforcement Learning (MARL), multiple RL agents are deployed into a common environment. The single-agent MDP framework becomes inadequate when multiple autonomous agents act simultaneously in the same domain. 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 system 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 system state and the rewards received by each agent depend on the joint action a of all of the agents in a SG, where a is derived from the combination of the individual actions a_i for each agent in the system. Each agent may have its own local state perception s_i, which is different to the system state s (i.e. individual agents are not assumed to have full observability of the system).

Note also that each agent may receive a different reward for the same system state transition, as each agent has its own separate reward function R_i. 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). Whether RL agents in a MAS will learn to act together or at cross-purposes depends on the reward scheme used for a specific application.

At each timestep in a SG, agents sense their own local state information s_i and each agent chooses an action $a_i \in A_i$ according to the policy $\pi_i \in \Pi_i$ that it is currently following. The product of all the individual actions results in a system joint action $a \in A$, and the joint action a selected in a system state s determines the next system state s'. All agents in the SG are then rewarded for this transition between system states based on the return from their individual reward functions R_i.

Multiple individual learners and joint action learners are two commonly used approaches when RL is applied to MAS. In the former case multiple agents are deployed into a common environment, each using a single-agent RL algorithm (e.g. Q-learning), whereas joint action learners use multi-agent specific algorithms which take account of the presence of other agents. A significant drawback inherent in the joint action learners approach is the exponential increase in the size of the state-action space for each additional agent that is added to a system (Claus & Boutilier 1998). Some algorithms are in between the individual and joint action approaches (see e.g. the FMQ heuristic (Kapetanakis & Kudenko 2002)); however they typically require agents

to learn and maintain additional stores of information beyond the tabular Q-values required by individual learners. For these reasons multiple individual learners are used exclusively throughout this thesis.

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). This notion of equilibrium was first introduced by Nash (1951), and is one of the most important concepts used to analyse MAS (Wooldridge 2001).

Formally, a joint policy π^{NE} leads to a Nash equilibrium when:

$$\forall i \in \{1, ..., N\}, \pi_i \in \Pi_i \big| R_i(\pi_i^{NE} \cup \pi_{-i}^{NE}) \geq R_i(\pi_i \cup \pi_{-i}^{NE}) \tag{2.3}$$

where Π is the set of joint policies, Π_{-i} is the set of joint policies excluding the set of policies Π_i for agent i, and π_i^{NE} is a specific policy followed by agent i when all other agents follow their policies from π_{-i}^{NE}.

Whenever the above inequality holds true for all possible policies for all agents in a MAS, a Nash equilibrium exists. In other words, a Nash equilibrium occurs whenever any individual agent cannot improve its own utility by changing its behaviour, assuming that all other agents in the MAS continue to behave in the same way.

In cooperative MAS, coordinating agents' actions to achieve the highest possible system utility is a difficult problem. While it is possible for multiple individual Q-learners in a cooperative MAS to converge to a point of equilibrium, there is no theoretical guarantee that the agents will converge to an optimal joint policy (one which maximises the system utility).

As RL agents seek to maximise the reward they receive, the design of the reward function directly affects the joint policies learned, and thus the issue of credit assignment in MARL is an area of active research. Two typical reward functions for MARL exist: local rewards unique to each agent and global rewards representative of the group's performance (Devlin et al. 2014).

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, resulting in a low "signal to noise ratio" (Agogino & Tumer 2008).

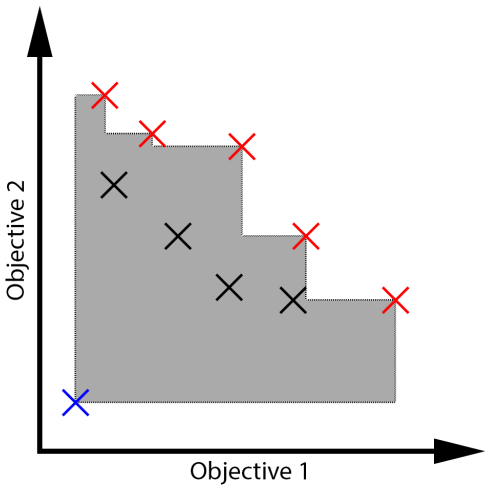


Figure 2.2: Solutions in red form the NDS, while solutions in black are said to be dominated. The shaded area denotes the hypervolume of the NDS with respect to the reference point (shown in blue).

2.4 Multi-Objective Reinforcement Learning

The majority of RL implementations aim to optimise systems with respect to a single objective, despite the fact that many real world problems are inherently multi-objective in nature. Single-objective approaches seek to find a single solution to a problem, whereas in reality a system may have multiple conflicting objectives that could be optimised. Multi-Objective Optimisation (MOO) approaches on the other hand address the requirement to make a trade-off between competing objectives.

Compromises between competing objectives can be defined using the concept of Pareto dominance (Pareto 1971). The Pareto optimal set or non-dominated set (NDS) consists of solutions which are incomparable, where each solution in the set is not dominated by any of the others on every objective. In Multi-Objective Reinforcement Learning (MORL) the reward signal is a vector, where each component represents the performance on a different objective.

The hypervolume metric measures the spread of a given set of non-dominated solutions; therefore, the diversity and accuracy of any set of solutions produced by an algorithm can easily be evaluated, by comparing its hypervolume with that of the the NDS produced by a competing algorithm, or with that of the true Pareto front of the application domain (if known).

In domains where the true Pareto front is known, its hypervolume represents an absolute maximum level of performance that may be achieved by an agent. Fig. 2.2 illustrates the process of determining which solutions constitute a NDS in a bi-objective maximisation problem, and of calculating its hypervolume with respect to a given reference point.

MORL problems may be defined using the MDP or SG framework as appropriate, in a sim-

ilar manner to single-objective problems. The main difference lies in the definition of the reward function: instead of returning a single scalar value r, the reward function $\mathbf{R}$ in multi-objective domains returns a vector $\mathbf{r}$ consisting of the rewards for each individual objective $c \in C$. Therefore, a regular MDP or SG can be extended to a Multi-Objective MDP (MOMDP) or Multi-Objective SG (MOSG) by modifying the return of the reward function.

It follows that the value function $\mathbf{V}^{\pi}(s)$ in multi-objective domains returns a vector $\mathbf{v}$ whose components are the expected discounted returns for each objective when starting in state s and following a policy π (Roijers et al. 2013):

$$\mathbf{V}^{\pi}(s) = E^{\pi}\left\{ \sum_{k=0}^{\infty} \gamma^{k} \mathbf{r}_{t+k+1} \mid s_t = s \right\} \tag{2.4}$$

A policy $\pi^* \in \Pi$ (where Π is the set of possible policies) is Pareto optimal if for every $\pi \in \Pi$ either,

$$\forall_{c \in C}[\mathbf{V}_c^{\pi}(s_0) = \mathbf{V}_c^{\pi^*}(s_0)] \tag{2.5}$$

or, there is at least one $c \in C$ such that

$$\mathbf{V}_c^{\pi}(s_0) < \mathbf{V}_c^{\pi^*}(s_0) \tag{2.6}$$

where $\mathbf{V}_c^{\pi}(s_0)$ is the expected discounted return for objective c when starting in state s_0 and following the policy π. That is, π^* is Pareto optimal if there exists no feasible policy π which would increase the value of one objective beyond that of π^* without causing a simultaneous decrease in the value of another objective. A policy that does not meet these criteria is dominated by another policy in Π. All policies not dominated by another are part of the NDS.

For a more complete overview of MORL beyond the summary presented in this section, the interested reader is referred to a survey article by Roijers et al. (2013).

2.4.1 Scalarised MORL

The majority of MORL approaches make use of single-policy algorithms in order to learn Pareto optimal solutions. Examples of single-policy algorithms include traditional temporal difference methods such as Q-learning and SARSA. In order to apply single-policy algorithms to MORL problems, scalarisation functions may be used to transform a reward vector $\mathbf{r}$ into a scalar reward signal r (Roijers et al. 2013). An agent learns using the scalarised version of the reward vector, and selects actions as normal by comparing the scalarised Q values for actions in a given state (using e.g. ϵ-greedy).

Linear scalarisation is commonly used in MORL literature (see e.g. Vamplew et al. (2010); Roijers et al. (2013); Van Moffaert et al. (2013b); Brys et al. (2014); Mason et al. (2016); Yliniemi & Tumer (2016)), and is defined in Eqn. 2.7 below:

$$r_+ = \sum_{c \in C} \mathbf{w}_c \mathbf{r}_c \tag{2.7}$$

where $\mathbf{w}$ is the objective weight vector, $\mathbf{w}_c$ is the weight for objective c, r_+ is the scalarised reward signal, $\mathbf{r}_c$ is the component of the reward vector $\mathbf{r}$ for objective c, and C is the set of objectives.

When using linear scalarisation, altering the weights in the weight vector allows the user to express the relative importance of the objectives. Although popular among MORL researchers, linear scalarised single-policy algorithms have the fundamental limitation that they can only learn solutions located in convex regions of the Pareto front (Vamplew et al. 2008).

Hypervolume scalarisation is also widely used throughout MORL literature (see e.g. Van Moffaert et al. (2013a); Wang & Sebag (2013); Yliniemi & Tumer (2016)), and the scalarised return r_λ from this function may be calculated as:

$$r_\lambda = \prod_{c \in C} \mathbf{r}_c \tag{2.8}$$

Scalarised MORL approaches sometimes make use of normalisation where the scale of the expected returns varies between objectives, or to simplify the process of selecting objective weights in the case of linear scalarisation. The normalised score on objective c may be calculated as (Marler & Arora 2004):

$$\mathbf{r}_c^{norm} = \frac{\mathbf{r}_c - \mathbf{r}_c^{min}}{\mathbf{r}_c^{max} - \mathbf{r}_c^{min}} \tag{2.9}$$

where $\mathbf{r}_c^{norm}$ is the normalised score on objective c, and $\mathbf{r}_c^{max}$ and $\mathbf{r}_c^{min}$ are the utopia and nadir values for objective c.

MOO approaches typically seek to produce a set of solutions that approximate the true Pareto front of a problem. In order to produce a set of Pareto optimal solutions using scalarised single-policy RL algorithms, researchers typically record the best non-dominated solutions found during a number of independent runs (see e.g. Vamplew et al. (2010); Van Moffaert et al. (2013b); Yliniemi & Tumer (2016)). These solutions are then compared with one another to produce an approximation of the Pareto front.

2.4.2 Multi-Policy Algorithms

Another approach taken by researchers is to apply multi-policy RL algorithms in order to learn a set of optimal policies in a single run. Examples of such algorithms include Convex Hull Value Iteration (Barrett & Narayanan 2008), Pareto Q-learning (Van Moffaert & Nowé 2014) and the work of Shelton (2001).

As the focus of this work is on the design of reward functions for MORL, and not on overcoming the limitations of scalarised MORL, a single-policy algorithm (scalarised Q-learning) is employed in the empirical sections of this thesis. Furthermore, multi-policy algorithms have not been studied in a multi-agent context to date, although this may be a promising avenue for future research, as will be outlined in Chapter 6.

2.5 Knowledge-Based Reinforcement Learning

As discussed earlier, it is commonly assumed that a RL agent must learn how to act in its environment with no prior knowledge. However, system designers typically do have at least some basic intuition about how a RL problem may be solved, rendering this assumption unnecessary in most application domains (Devlin 2013).

In Knowledge-Based Reinforcement Learning (KBRL), the system designer provides domain knowledge to an agent in order to guide its behaviour, with the goal of improving learning speed and/or the final policy learned. The term domain knowledge encompasses any additional information provided before, during or after a learning episode which is intended to influence the decisions of an agent. This information may be specified directly by the system designer, or be may derived in an automated or semi-automated manner (e.g. policy transfer between agents, or shaping functions that are learned or generated).

Domain knowledge which is specified directly by the system designer is referred to as a heuristic. Heuristics may be simple "rules of thumb" which provide general guidance about how to act in a domain (e.g. "move towards the goal"), or may be more specific (e.g. "move North to reach the goal"). If they are incomplete or misleading, heuristics could potentially damage the performance and/or learning speed of an agent. Therefore, it is important that any methods which provide additional information to agents are robust to poorly-designed heuristics.

This section will give an overview of the broad field of KBRL, highlighting the most significant developments to date.

2.5.1 Q Value Initialisation

Recall from Section 2.2 that in model-free RL, an agent stores estimates of the expected utility for each state-action pair in the form of Q values (or a Q function in non-tabular representations). All that a model-free agent learns about its environment is encoded in this form, and the choice of initial value function estimates has been shown to have a considerable effect on learning performance in goal-orientated tasks (Koenig & Simmons 1996).

Therefore, it is intuitive that if this internal memory could be "bootstrapped" or initialised with some domain-specific information, an agent's performance could be improved over that achieved using optimistic, pessimistic or arbitrary Q value initialisation.

This method does not affect the convergence guarantees for Q-learning in MDPs, as in theory the required condition of sampling all state-action pairs infinitely often can be satisfied regardless

of which initial Q values are used. The most basic implementation of this form of KBRL requires the system designer to convert heuristic information into either a tabular representation or a Q function so that it can be used by an agent. This is an unintuitive and painstaking process, and therefore will not be covered further in this thesis.

2.5.2 Transfer Learning

In Machine Learning research, the term Transfer Learning (TL) describes a family of techniques that improve learning in an unseen task, through the transfer of knowledge from a related task that has been learned previously (Torrey & Shavlik 2010). The main intuition behind TL is that generalisation may happen across different tasks, as well as within tasks (Taylor & Stone 2009).

Although TL is a field of study in its own right, in a RL context it is essentially a form of KBRL. Rather than incorporating information specified directly by the system designer, TL instead uses knowledge learned by an agent in simple domains to improve performance in more complex domains.

2.5.3 Probabilistic Policy Reuse

Fernández & Veloso (2006) proposed Probabilistic Policy Reuse (PPR), which functions in a similar vein to the Transfer Learning techniques discussed above. PPR allows policies that are learned in a source task to be reused in a previously unseen, but related target task. When implementing PPR, an agent selects an action according to the reused policy with probability ψ, or selects an action using a standard strategy such as ϵ-greedy with probability $1 - \psi$. The value of ψ can then be decayed over time to reduce the effect of the reused policy.

As long as ψ is decayed appropriately, it eventually ceases to affect the learning process, and therefore will not affect the convergence guarantees provided by Q-learning. Although PPR was originally proposed as a method to transfer policies learned in one task to a similar task, it could in also be used to incorporate domain knowledge in the form of a policy that is handcrafted by the system designer, as noted by Brys (2016).

2.5.4 Extra Action Method

Taylor & Stone (2007) proposed the "extra action technique" which biases an agent's exploration by introducing an additional action. This additional action, when selected, chooses one of the regular actions on the agent's behalf, according to heuristic information provided by an expert. As this technique simply duplicates one of the existing actions in each environmental state, it does not have any effect on the convergence guarantees of Q-learning.

2.5.5 Heuristically Accelerated Q-learning

Bianchi et al. (2004) proposed Heuristically Accelerated Q-learning, also known as heuristic selection of actions, or Heuristically Accelerated Reinforcement Learning (HARL). This technique is similar to PPR, in that it alters the action selection process directly. HARL modifies an agent's

greedy mechanism by instead selecting the action that maximises the sum of Q value and a heuristic function for a particular state-action pair, rather than the action that has the largest Q value. However, it can only incorporate knowledge regarding preferred actions, and not preferred states.

HARL does not affect the optimal policy of a single-agent domain, as it only influences an agent's exploration, and does not modify the update rule, rewards or MDP (Bianchi et al. 2012). This method has also been applied to MARL, by altering the action selection method of mini-max Q-learning (Bianchi et al. 2007, 2014).

2.5.6 Integrated Partial Model

Tamar et al. (2012) proposed the Integrated Partial Model (IPM) technique, which augments model-free RL with a partial model. In contrast to the other methods surveyed, IPM works by modifying the update rule, and domain knowledge may be supplied by the system designer in the form of a partial model of the transition function of the environment. IPM then uses this knowledge to reduce the noise in updates to value function estimates for states where information is available. However, Tamar et al. (2012) give little guidance about how partial models may best be designed or acquired, other than suggesting the use of TL in single-agent setings, or sharing components of partial models between agents in the case of a MAS.

2.5.7 Reward Shaping

As discussed in Sections 2.2 and 2.3, the reward function R determines the reward that an agent receives from its environment for each state transition. RL agents typically learn how to act in their environment guided by the reward signal alone. Additional knowledge can be provided to a learner by the addition of a shaping reward to the reward naturally received from the environment. This principle is referred to as reward shaping.

The term shaping has its origins in the field of experimental psychology, and describes the idea of rewarding all behaviour that leads to the desired behaviour. Skinner (1938) discovered while training a rat to push a lever that any movement in the direction of the lever had to be rewarded to encourage the rat to complete the task. Analogously to the rat, an RL agent may take an unacceptably long time to discover its goal when learning from delayed rewards, and shaping offers an opportunity to speed up the learning process.

Reward shaping allows a reward function to be engineered in order to provide more frequent feedback on appropriate behaviours (Wiewiora 2017), which is especially useful in domains with sparse rewards. Generally, the return from the reward function is modified as follows:

$$r' = r + f \tag{2.10}$$

where r is the return from the original reward function R, f is the additional reward from a shaping function F, and r' is the signal given to the agent by the augmented reward function R'.

It follows that an augmented value function V' must now be defined (Wiewiora 2017):

$$V'^{\pi}(s) = E^{\pi}\left\{ \sum_{k=0}^{\infty} \gamma^k (r_{t+k+1} + f_{t+k+1}) \mid s_t = s \right\} \tag{2.11}$$

Empirical evidence has shown that reward shaping can be a powerful tool to improve the learning speed of RL agents (Randløv & Alstrøm 1998); however, it can have unintended consequences. The implication of modifying the value function as in Eqn. 2.11 is that a policy which is optimal for the augmented reward function R' may not in fact also be optimal for the original reward function R.

A classic example of reward shaping gone wrong for this exact reason is reported by Randløv & 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 discussed earlier, an RL agent will attempt to maximise the reward received during its lifetime, so the policy learned depends directly on the definition of the reward function. Thus, shaping rewards in an arbitrary fashion can modify the optimal policy and cause unintended behaviour. For this reason, only two specific and principled forms of reward shaping (Potential-Based Reward Shaping and difference rewards) will be considered in this thesis, due to their pre-existing theoretical guarantees when applied in single-objective domains.

2.5.8 Summary

The techniques described above each take a distinct approach to incorporating knowledge, and have all been demonstrated to improve the performance of RL agents during learning. Although any of them could plausibly be applied to MORL domains, this thesis will focus exclusively on reward shaping. As I will discuss in Sections 2.6 to 2.8, reward shaping has developed into a distinct field of study within RL, and has proven to be an extremely successful technique to improve learning performance in traditional single-objective MDPs and SGs.

Reward shaping methods have been developed that allow heuristic information to be encoded as preferences for states, actions, or state-action pairs, arguably making it one of the most flexible KBRL techniques surveyed. Some specific forms of reward shaping also have strong theoretical guarantees. Furthermore, reward shaping can fulfill the same purpose as other KBRL techniques such as TL or Q value initialisation, as will be explained later in this chapter. These successes, especially in multi-agent domains, led to my choice of reward shaping as the most promising candidate to improve MORL using domain knowledge. The conclusion of this thesis (Chapter 6) outlines how other KBRL methods could potentially be adapted for MORL in future work.

2.6 Potential-Based Reward Shaping

Now that the concepts of Knowledge-Based Reinforcement Learning and reward shaping have been introduced, it is timely to consider the first of the two specific reward shaping techniques that will be evaluated throughout this thesis. This section will cover all of the prior theoretical work on Potential-Based Reward Shaping, explaining the implications of its guarantees, and under what conditions they hold true.

2.6.1 Fundamental Theories

To avoid the problems associated with arbitrary shaping rewards outlined in Section 2.5.7 above, Ng et al. (1999) proposed Potential-Based Reward Shaping ($PBRS$). 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 an episodic 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. 2.12 below:

$$F(s, s') = \gamma\Phi(s') - \Phi(s) \tag{2.12}$$

where $\Phi(s)$ is the potential function which returns the potential for 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, and thus does not suffer from the problems of arbitrary reward shaping approaches outlined above. 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.

Wiewiora (2003) proved equivalence between $PBRS$ and Q-value initialisation, stating that an agent learning with $PBRS$ and zero Q-table initialisation will behave identically to an agent learning without $PBRS$ if the latter agent's Q-table is initialised using the same potential function. Given the equivalence between $PBRS$ and Q table initialisation, and the effect of initial value function estimates demonstrated by Koenig & Simmons (1996), care must be taken when designing potential functions to avoid reducing learning performance.

The form of $PBRS$ described above can however only represent domain knowledge as a preference for different states, and does not provide any information to the agent regarding specific actions that may be beneficial. Wiewiora et al. (2003) extended $PBRS$ to allow knowledge regarding favourable actions to be included. This extension is called Potential-Based Advice. As part of this extension, the potential function is expressed for a state-action pair i.e. $\Phi(s, a)$, rather than for a state only as per the definition $\Phi(s)$ used by Ng et al. (1999). Wiewiora et al. (2003) proposed two different forms of Potential-Based Advice: Look-Ahead Advice and Look-Back Advice. The former method defines the additional shaping function F as shown in Eqn. 2.13:

$$F(s, a, s', a') = \gamma\Phi(s', a') - \Phi(s, a) \tag{2.13}$$

In order to guarantee policy invariance for Look-Ahead Advice in single-agent learning scenarios, certain additional criteria must be satisfied. Specifically, the way in which an agent selects actions must be modified, so that its policy π chooses the action that will maximise the sum of Q-value and potential. The required modification to π is described formally below:

$$\pi(s) = argmax_a(Q(s, a) + \Phi(s, a)) \tag{2.14}$$

Here $\pi(s)$ is the agent's policy in state s (the action that will be chosen by the agent in state s). This modification is necessary to maintain policy invariance, because when using Look-Ahead Advice the value of all state-action pairs becomes:

$$Q_\Phi(s, a) = Q^*(s, a) - \Phi(s, a) \tag{2.15}$$

where $Q_\Phi(s, a)$ is the value of selecting action a in state s, $Q^*(s, a)$ is the actual value of selecting action a in state s when not receiving $PBRS$, and $\Phi(s, a)$ is the potential of the state-action pair (s, a). As the addition of shaping may modify the value of different actions within the same state by different amounts, the ordering of action preferences may be changed when selecting actions using $Q_\Phi(s, a)$ rather than $Q^*(s, a)$. By selecting actions according to Eqn. 2.14 the correct order of action preferences is restored upon convergence. This is referred to as a biased-greedy policy, although naturally biased variants of other greedy-based policies (e.g. ϵ-greedy or softmax) can easily be created and used instead.

For the second form of Potential-Based Advice proposed by Wiewiora et al. (2003), the additional shaping reward f when receiving Look-Back Advice is computed as shown in Eqn. 2.16:

$$F(s, a, s', a') = \Phi(s', a') - \gamma^{-1}\Phi(s, a) \tag{2.16}$$

Unlike Look-Ahead Advice, no corresponding proof of policy invariance for a single agent learning with Look-Back Advice has been published, although empirical results suggest that Look-Back Advice does not modify the optimal policy in single-agent learning scenarios (Wiewiora et al. 2003). Also, no counter example has yet been published showing a case where Look-Back Advice causes an agent to diverge from learning the optimal policy. When using Look-Back Advice, Wiewiora et al. (2003) suggest the use of an on-policy algorithm such as SARSA.

Wiewiora et al. (2003) recommended the use of Look-Ahead Advice when the prior knowledge predominately consists of information regarding state preferences, and Look-Back Advice for prior knowledge which is mostly comprised of preferences for action choices. The standard form of $PBRS$ proposed by Ng et al. (1999) is of course still sufficient when representing prior knowledge consisting of preferences regarding states only.

While Look-Ahead Advice has received some attention in empirical studies (see e.g. Devlin et al. (2011a); Harutyunyan et al. (2015); Brys et al. (2015)), Look-Back Advice has not been adopted by researchers to the same degree, most likely due to the lack of theoretical guarantees. For this reason, Look-Ahead Advice has been used in the empirical sections of this thesis where action-based potential functions are evaluated. From this point forth, the abbreviations $sPBRS$ and $aPBRS$ will be used to refer to $PBRS$ approaches with state-based (Eqn. 2.12) and action-based (Eqn. 2.13) potential functions respectively.

2.6.2 Potential-Based Reward Shaping for MARL

Up to now, $PBRS$ has been discussed purely in the context of SARL problems. Some early attempts were made to apply $PBRS$ to MARL (Marthi 2007; Babes et al. 2008; Devlin et al. 2010); however, these works did not consider the theoretical implications of applying $PBRS$ to MARL, namely if the proofs originally derived for single-agent problem domains also hold in MARL scenarios.

Work by Devlin & Kudenko (2011) and by Lu et al. (2011) explored this question, and both sets of authors proved independently that $PBRS$ does not alter the set of Nash equilibria of an infinite-horizon SG, extending the results of the fundamental work on $PBRS$ for single-agent problems by Ng et al. (1999). Whether the agents in the MAS will actually converge to one of these equilibria is dependent on the specific learning algorithms used.

It is important to note that each agent in a MAS may have its own specific potential function Φ_i without affecting the theoretical guarantees (Lu et al. 2011); therefore $PBRS$ could potentially be used to encourage individual agents in the same MAS to adopt different behaviours. Devlin & Kudenko (2011) also showed that an agent receiving $PBRS$ is still equivalent to an agent with Q-values initialised to the same potential function regardless of the number of agents acting in the same environment, confirming that the original proof by Wiewiora (2003) can be extended to MARL scenarios.

While it has been proven that rewarding any number of agents in a MAS with additional potential-based rewards has no effect on the set of Nash equilibria for a SG, empirical evidence has shown that adding $PBRS$ can alter the joint policy that the agents converge to (Devlin et al. 2011a; Devlin 2013). However, this policy is guaranteed to be from the same set of policies that the agents could have learned without the influence of $PBRS$. This was one of the main findings reported by Devlin (2013) in his PhD thesis on $PBRS$ for MARL. As with SARL, care must be taken when designing potential functions for MARL applications. $PBRS$ in a MAS can increase the final performance and/or decrease the time to convergence of the agents, but using a poorly designed heuristic can be detrimental to performance and learning speed.

2.6.3 Dynamic Potential Functions

The proofs discussed in sections 2.6.1 and 2.6.2 above all hold a common assumption: that the potential function is static (i.e. it does not change during learning). This can be considered to be a limitation, as the potentials cannot be updated dynamically online. Several authors showed that it can be beneficial to break this assumption and use a dynamic potential function when applying $PBRS$ to a problem (Laud 2004; Marthi 2007; Grześ & Kudenko 2008, 2010).

One of the challenges of implementing $PBRS$ is choosing a suitable potential function, and these applications of $PBRS$ attempt to generate a suitable potential function automatically online. However, these authors did not consider the theoretical implications of breaking the assumption of a static potential function. Devlin & Kudenko (2012) proved that the use of a dynamic potential function does not alter the optimal policy in SARL, or the set of Nash equilibria in MARL provided the shaping given is of the form:

$$F(s,t,s',t') = \gamma\Phi(s',t') - \Phi(s,t) \tag{2.17}$$

Devlin & Kudenko (2012) found, however, that the use of a dynamic potential function breaks the equivalence between Q value initialisation and $PBRS$, which had been proven for SARL (Wiewiora 2003) and MARL (Devlin & Kudenko 2011) scenarios using a static potential function. No prior initialisation can replicate the behaviour of an agent receiving dynamic $PBRS$, as the changes to the potentials cannot be known before learning begins. Thus dynamic $PBRS$ offers a unique way to provide an agent with knowledge that can be updated online during learning, while still preserving the guarantees of policy invariance in SARL and consistent Nash equilibria in MARL. This proof provides a theoretical foundation for the empirically demonstrated implementations of dynamic $PBRS$ discussed above (Laud 2004; Marthi 2007; Grześ & Kudenko 2008, 2010).

Another interesting finding by Devlin & Kudenko (2012) is that the dynamic potential function does not need to converge before the agent receiving shaping can converge, a point which they supported with theoretical and empirical evidence. This is an important finding, contrary to previous arguments about the need for dynamic potential functions to be convergent made by Laud (2004).

Harutyunyan et al. (2015) extended the Look-Ahead Advice method proposed by Wiewiora et al. (2003), to allow the use of dynamic potential functions incorporating information about both states and actions. Analogously to the extension of $sPBRS$ proposed by Devlin & Kudenko (2012), Harutyunyan et al. (2015) augment Wiewiora's Look-Ahead Advice function with a time parameter as shown in Eqn. 2.18 below:

$$F(s,a,t,s',a',t') = \gamma\Phi(s',a',t') - \Phi(s,a,t) \tag{2.18}$$

As with the static potential version of Look-Ahead Advice, an agent's policy must be modified to preserve optimality (in a similar manner to Eqn. 2.14).

$$\pi(s) = argmax_a(Q(s,a) + \Phi_0(s,a)) \tag{2.19}$$

where $\Phi_0(s,a)$ is the initial dynamic potential function.

2.6.4 Potential-Based Reward Shaping in Finite-Horizon Domains

Many of the domains to which RL is applied are episodic, i.e. they consist of a finite number of timesteps. However, much of the theoretical work on $PBRS$ considered its effect on infinite-horizon domains only (see e.g. Wiewiora et al. (2003); Devlin & Kudenko (2011, 2012)).

In his PhD thesis, Devlin (2013) demonstrated that by removing the assumption of an infinite horizon, the relative ordering of expected rewards could be altered by the addition of a potential-based shaping function, thus breaking the guarantees of policy invariance in MDPs and consistent Nash equilibria in SGs. Therefore, additional assumptions are required to preserve these guarantees in finite-horizon domains.

Specifically, Devlin (2013) proved that when the potential of the final state reached is equal to zero, these guarantees are preserved in finite-horizon domains. An independent theoretical analysis by Grześ (2017) also reached the same conclusion as Devlin (2013).

Three different methods were suggested by Devlin (2013) to ensure that the theoretical guarantees hold in episodic domains. Firstly, the zero potential requirement could be taken account during the design of the potential function, in cases where the terminal states are known. Alternatively, the potential of the last state reached could also be changed to zero dynamically in cases where the terminal states are not known beforehand.

Finally, an absorbing state with zero potential could be implemented. Upon reaching a terminal state, all agents in the environment select actions as normal and transition to an absorbing state. Agents do not receive any reward from the environment for this transition.

When the final potential is set to zero using any of these methods, the ordering of expected returns is not altered by the use of a potential-based shaping function. In all finite-horizon empirical work in this thesis, an absorbing state with zero potential is implemented to ensure that the theoretical guarantees of $PBRS$ hold. The theoretical analysis of $PBRS$ in MORL domains in Chapter 4 considers both infinite-horizon and finite-horizon cases.

2.7 Applications of Potential-Based Reward Shaping

Now that the fundamental theories of $PBRS$ have been collected together and discussed, this section will consider how other authors have applied these theories. $PBRS$ has achieved broad applicability across a number of different areas, beyond the model-free SARL applications for which it was originally developed by Ng et al. (1999).

2.7.1 Model-based Reinforcement Learning with PBRS

Although $PBRS$ was originally devised for use with model-free RL, a number of researchers have applied it in conjunction with model-based algorithms. Asmuth et al. (2008) first investigated the use of $PBRS$ with a model-based RL algorithm, namely Rmax. Empirical results in Gridworld domains presented by Asmuth et al. (2008) demonstrated that the addition of $sPBRS$ with a suitable heuristic can improve performance compared with that of unshaped Rmax.

The theoretical guarantees provided by Rmax are of a different form to those of Q-learning, and are referred to as PAC-MDP (probably approximately correct for MDPs) (Brafman & Tennenholtz 2003). Asmuth et al. (2008) argued that for Rmax's PAC-MDP guarantees to be preserved, the potential function must be admissible. Russell & Norvig (2009) define an admissible heuristic as is one that never overestimates the cost to reach the goal. In other words, a potential function is admissible if it assigns a value to every state that is no larger than the shortest distance to the goal from that state (Asmuth et al. 2008).

Grześ (2017) later proved that potential functions for Rmax do not in fact have to be admissible to preserve PAC-MDP, and that setting the potential of all unknown states to a minimum of zero is sufficient to guarantee optimistic exploration.

Kim et al. (2015) demonstrated the use of $PBRS$ with model-based Bayesian Reinforcement Learning (BRL). The use of shaping was found to mitigate against sparsity and delay in the rewards received, assisting the algorithm to find good actions without requiring large search trees for long-horizon planning. A number of domain-independent potential functions suited to BRL were proposed and evaluated on five different problem domains, and $PBRS$ was found to improve learning performance when compared to an agent without shaping.

2.7.2 Learning and Generating Potential Functions

One of the main challenges of implementing $PBRS$ successfully is designing an appropriate potential function. Numerous authors have recognised this challenge, leading to several proposals for methods capable of generating potential functions automatically without the need for specific input from the system designer.

Marthi (2007) proposed an automated method to generate potential functions, where an abstract MDP is generated which is far simpler than the application domain MDP. This abstraction of the original problem domain is first solved, and the final learned value function for the abstract MDP is used to shape the agent learning in the actual problem domain, dramatically decreasing the time required to learn. Efthymiadis & Kudenko (2014) also investigated abstract MDP reward shaping. In contrast to Marthi's approach, the authors developed a modified abstract MDP reward shaping method designed to incorporate prior knowledge.

Grześ & Kudenko (2010) avoid the need for prior knowledge by learning two Q functions concurrently from the MDP, both with different levels of discretisation. The more abstract Q function converges more quickly than the low-level Q function, due to a smaller state-action

space. This abstract Q function is used to shape the agent's rewards while it learns the low-level Q function, improving learning speed without the need to design a specific $PBRS$ implementation for the problem domain.

A second automated method of generating potential functions was proposed by Grześ & Kudenko (2010), this time specifically designed for use with model-based RL agents. They based their approach on the R-max algorithm, and it is designed for cases where an admissible heuristic cannot be easily created. The authors found that their approach led to a reduction in sample complexity, while not requiring a pre-specified heuristic, as the potential function is learned concurrently online.

2.7.3 Arbitrary Reward Functions as Potential-Based Advice

One drawback of the $PBRS$ framework is that encoding domain knowledge as a potential function is often difficult. Expressing a heuristic as a potential function requires another layer of abstraction beyond what is required in simple additive reward shaping approaches.

Harutyunyan et al. (2015) developed a technique which allows any arbitrary reward function to be used as a potential-based shaping function. Their method learns a secondary Q function w.r.t. a variant of an arbitrary reward function, and uses consecutive estimates of this Q function as dynamic advice potentials. Significantly, this makes the use of $PBRS$ more intuitive by allowing any non-potential-based shaping function to benefit from the theoretical guarantees of $PBRS$.

Brys et al. (2015) demonstrated that the contribution of Harutyunyan et al. (2015) could be used as a means of policy transfer via reward shaping. In Transfer Learning, experience gained by an agent in a source task is transferred to an agent in a target task, with the goal of improving learning speed in the target task. Typically, the source task is a simpler version of the target task. Even so, the state and/or action representations may be different in source and target tasks, which creates the challenge of determining an appropriate mapping.

Brys et al. (2015) developed an approach where transferred policies from the source task are represented as dynamic potential-based shaping functions, and used to shape the exploration of the agent in the target task. The authors evaluated policy transfer using reward shaping empirically in the Mountain Car 3D, Cart Pole and Mario problem domains.

Harutyunyan et al. (2015) applied the work on arbitrary potential-based shaping functions (Harutyunyan et al. 2015) to incorporate human feedback into an agent learning online to play Mario. Significantly, this method allows the incorporation of human feedback during learning (a field of RL research in its own right) while preserving the theoretical guarantees of $PBRS$.

2.7.4 Plan-Based Reward Shaping

Grześ & Kudenko (2008) proposed Plan-Based Reward Shaping (PlanBRS), an extension of $PBRS$ that uses a STRIPS planner to generate high level plans. It is a semi-automated method,

that attempts to simplify the process of creating a potential function from prior knowledge. A STRIPS plan is converted into a state-based representation, where each state in the high level plan maps to one or more in the environment. The state-based representation is used to create a potential function, where states later in the plan are assigned a higher potential than those earlier or not included in the plan. The potential function is defined as shown in Eqn. 2.20 below:

$$\Phi(s) = CurrentStepInPlan \times w \tag{2.20}$$

where $CurrentStepInPlan$ is the corresponding state in the state-based representation of the agent's plan, and w is a scaling factor.

If the agent's current state is not in the plan, the value of potential used is that of the last state experienced that was in the plan. One option is to assign a potential of zero to states not in the plan; however, Grześ & Kudenko (2008) chose the potential of the last state experienced in the plan, as it does not discourage the agent from exploring off the plan. This is important in cases where the plan contains incorrect knowledge, as use of a zero potential for states can strongly bias the agent towards following the given (possibly incorrect or incomplete) plan. As with $PBRS$, the potential of all goal states is set to zero in PlanBRS to ensure that theoretical guarantees are preserved.

Devlin & Kudenko (2016) investigated the use of Plan-Based Reward Shaping in a multi-agent setting. They proposed two methods of extending PlanBRS to MARL, inspired by the two opposing methods in multi-agent planning literature: centralised planning and decentralised planning. Joint PlanBRS is centrally planned, and attempts to create multi-agent plans without conflicts.

However, it is not reasonable to expect that self-interested agents acting in a MAS will always share knowledge with each other. Individual PlanBRS is a decentralised method, and requires no cooperation as each agent plans as if it is alone in the environment. As individually generated plans will likely contain conflicts between agents, strict adherence to the plan by every agent in the MAS will not be possible. The authors also note that PlanBRS is actually an instance of dynamic $PBRS$ (Devlin & Kudenko 2012).

Devlin & Kudenko (2016) tested their two proposed PlanBRS methods on two different flag collection tasks, where agents must coordinate their actions to collect all of the flags in the domain. Joint PlanBRS offered the best performance, collecting a higher average reward per episode than both Individual PlanBRS and an unshaped benchmark. Despite conflicts in the plans of the Individual PlanBRS agents, they learned to overcome those conflicts and outperformed agents with no prior knowledge. This suggests that Individual PlanBRS is a viable method for agents that are unwilling to share information to adopt, although Joint PlanBRS offers better performance for situations when information can be shared between agents at the planning stage.

2.7.5 Knowledge Revision

In the original work on PlanBRS by Grześ & Kudenko (2008), there was no explicit mechanism to deal with incorrect guidance from a faulty plan. An agent shaped by a poorly designed plan could suffer long convergence times and poor returns in terms of total reward received. Furthermore, in the original implementation of PlanBRS, plans had to be produced using a manual transformation between action-based and state-based plans.

Efthymiadis et al. (2016) extended PlanBRS to overcome these issues, by incorporating an automatic knowledge revision process and by automating the process of plan transformation. The authors compared agents learning in a flag collection domain using the correct plan, an incorrect plan, and an incorrect plan with knowledge revision, and demonstrated empirically that agents learning with their knowledge revision method can improve their incorrect plans sufficiently to reach the same level of performance as agents learning using the correct plan.

One drawback of the improved PlanBRS method above, is that it is incapable of dealing with non-deterministic environments. Efthymiadis & Kudenko (2015) identified this shortcoming, and applied the principles of knowledge revision to abstract MDP reward shaping also. Abstract MDP reward shaping can deal with non-determinism, as it is not dependent on a prespecified plan as with PlanBRS, but rather on an additional high level Q function. At the end of each episode, the potential function is revised to improve the quality of the shaping, which makes this method an instance of dynamic $PBRS$ (Devlin & Kudenko 2012).

Efthymiadis & Kudenko (2015) demonstrated empirically that abstract MDP reward shaping with knowledge revision can achieve similar results to PlanBRS with knowledge revision in deterministic environments. When applied to a non-deterministic Micro UAV problem, Efthymiadis & Kudenko (2015) found that agents learning using an incorrect abstract MDP shaping and knowledge revision could quickly reach a similar level of performance as that achieved by agents learning with the correct abstract MDP shaping.

2.7.6 Planning with PBRS

Eck et al. (2013, 2016) extended $sPBRS$ to an application in the field of online Partialy Observabve Markov Decision Process (POMDP) planning. This is the first extension of $PBRS$ to explicitly account for partially observable environments. The proposed approach defines the potential function using the agent's belief states, indicating the ability of the agent to earn future rewards. Eck et al. (2016) reported that the effect of $sPBRS$ is more significant when using shorter planning horizons. The authors provide theoretical results for their adaption of $sPBRS$ to POMDP planning, proving that planning with $sPBRS$ also still optimises the original unshaped reward function in infinite-horizon domains. However, they found that finite-horizon planning using $sPBRS$ can lead to different policies than planning with the original unshaped rewards; Grześ (2017) later noted that policy invariance can be preserved in POMDP planning if the final potential is set to zero.

2.7.7 Multi-Objectivisation with PBRS

All of the theoretical and empirical work on $PBRS$ discussed thus far has considered domains with a single objective only. Multi-objectivisation is the process of adding additional utility functions to a task, beginning from a single one (Knowles et al. 2001). Brys et al. (2014, 2017) used $PBRS$ to incorporate multi-objectivisation into single-objective SARL. In this way, an agent can be encouraged to optimise additional psuedo objectives in order to speed up learning. This approach has successfully been applied to traditional single-objective SARL domains such as Cart Pole and Pursuit, significantly improving performance when compared with an unshaped agent.

However, this work does not evaluate $PBRS$ in true multi-objective problems (i.e. where the objectives are in conflict). Besides the work of Brys et al. (2014, 2017) there have been no prior attempts to use $PBRS$ in domains with multiple objectives, apart from the investigations in this thesis. Therefore, the area of reward shaping for MORL merits substantial further study.

2.8 Difference Rewards

Difference rewards are a form of reward shaping specific to MAS for cooperative Stochastic Games. A MAS is often designed with individual reward functions for each agent, and the resulting emergent behaviour is studied. Difference rewards are inspired by the intuition that all agents in a cooperative MAS should try to contribute to the global utility, and make use of a global utility function G.

A difference reward (D_i) is a shaped reward signal that aims to quantify each agent's individual contribution to the system performance in a cooperative MAS (Wolpert et al. 2000). Formally:

$$D_i(s_i, a_i) = G(s, a) - G(s_{-i} \cup s_i^c, a_{-i} \cup a_i^c) \tag{2.21}$$

where $G(s, a)$ is the global system utility, s is the system state, a is the joint action, and $G(s_{-i} \cup s_i^c, a_{-i} \cup a_i^c)$ is the counterfactual which represents the global utility for a theoretical system without the contribution of agent i.

The terms s_{-i} and a_{-i} refer to all the states and actions not involving agent i, while s_i^c and a_i^c are fixed states and actions not dependent on agent i. Typically, the counterfactual system utility is calculated with agent i removed, or by assuming a default state/action for agent i.

Difference rewards are a well-established shaping methodology, with many successful applications in MARL (see e.g. (Wolpert & Tumer 2002; Tumer & Agogino 2007; Devlin et al. 2014; Malialis et al. 2015)).

Recent work by Colby & Tumer (2015) proved that the relative ordering of expected returns (and therefore the Nash equilibria) are not altered when agents are rewarded using D instead of G in a two-player single-objective matrix game. D has also recently been extended to increase its

effectiveness in problem domains where agents' actions must be tightly coordinated to achieve a high level of system performance (Rahmattalabi et al. 2016).

2.8.1 Approximated Difference Rewards

Although difference evaluations have been shown to successfully improve MARL performance, they suffer from some notable limitations: global knowledge about the system state and joint action must be available, and the precise form of the system evaluation function G must be known in order to calculate D.

Furthermore, D requires the assumption that a centralised mechanism is available to provide tailored feedback to individual agents (Colby et al. 2016). Therefore it is difficult to apply D in situations where communication is limited, the system evaluation function is not known, or where global state and action information is unavailable, as may be the case in practice as MARL is applied to more complex MAS.

Recent work by Colby et al. (2016) attempted to address these limitations by approximating the counterfactual term, giving an estimated difference reward $\hat{D}$ as shown in Eqn. 2.22, where $\hat{G}_i(s_i^c, a_i^c)$ is the approximated counterfactual. This method requires each agent to maintain a private approximation $\hat{G}_i(s_i, a_i)$ of the system evaluation function. A simple tabular approximation of $\hat{G}_i(s_i, a_i)$ can be maintained using a temporal difference update as shown in Eqn. 2.23, where β is the $\hat{G}_i(s_i, a_i)$ learning rate. To evaluate $\hat{G}_i(s_i^c, a_i^c)$, default states s_i^c and default actions a_i^c are chosen for each agent.

$$\hat{D}_i(s, a) = G(s, a) - \hat{G}_i(s_i^c, a_i^c) \tag{2.22}$$

$$\hat{G}_i(s_i, a_i) \leftarrow \hat{G}_i(s_i, a_i) + \beta[G(s, a) - \hat{G}_i(s_i, a_i)] \tag{2.23}$$

While $\hat{D}$ has proven to be successful in shaping agents' behaviour in several problem domains, and requires much less information than D for successful implementation, it does not provide any theoretical guarantees. Therefore it may alter the Nash equilibria of a SG (and/or the Pareto optimal solutions in a MOSG), and cause agents to learn behaviours that would not be learned when using the unshaped system evaluation function G.

2.8.2 Potential-Based Difference Rewards

Potential-Based Difference Rewards (PBDR) were proposed by Devlin et al. (2014), in order to leverage the benefits of both $PBRS$ and difference rewards (Eqn. 2.21) in cooperative multi-agent problem domains. Devlin et al. (2014) introduced two different methods that could combine $PBRS$ and difference rewards: Counterfactual as Potential (CaP) and Difference Rewards incorporating Potential-Based Reward Shaping ($DRiP$).

The former is an automated method of generating multi-agent potential functions using the same knowledge represented by D, and so could be seen as a way of addressing the challenge of designing potential functions. CaP automatically assigns potentials to states using the counterfactual term. Formally:

$$\Phi_i(s_i) = G(s_{-i} \cup s_i^c, a_{-i} \cup a_i^c) \tag{2.24}$$

where $\Phi_i(s_i)$ is the potential function for agent i in its percieved local state s_i. In this framework, the unshaped reward $R_i(s_i, a_i, s_i') = G(s, a, s')$, and the shaping reward $F_i(s_i, s_i')$ is calculated as normal in $sPBRS$ according to Eqn. 2.12.

As $\Phi_i(s_i)$ 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 $sPBRS$. CaP therefore preserves the guarantee of consistent Nash equilibria, while incorporating knowledge based on D in an automated manner.

$DRiP$ on the other hand offers a method to augment the traditional difference reward signal with heuristic knowledge specified by the system designer in the form of a potential function. The form of the shaping reward for agent i in this case is:

$$F_i(s_i, s_i') = -G(s_{-i} \cup s_i^c, a_{-i} \cup a_i^c) + \gamma\Phi_i(s_i') - \Phi(s_i) \tag{2.25}$$

i.e. a simple addition of the standard $sPBRS$ shaping in Eqn. 2.12 to the counterfactual term normally used when calculating D. This approach allows the use of prior knowledge in the form of a potential function, as well as leveraging the benefit of difference rewards, but does not preserve the guarantee of consistent Nash equilibria provided by $PBRS$, as not all of the shaping terms are potential-based.

Empirical evaluations of both PBDR methods proposed demonstrated their viability, with $DRiP$ consistently outperforming both CaP and traditional difference rewards (Devlin et al. 2014). Of course, the performance of $DRiP$ is highly dependent on the specific potential function used, as with all $PBRS$ methods. CaP may be useful in cooperative multi-agent domains where theoretical guarantees are required, and also avoids the need to implement a bespoke potential function. Alternatively, $DRiP$ offers significantly better performance for applications where preserving the theoretical guarantees of $PBRS$ are not a concern.

Devlin et al. (2014) noted that PBDR could also be extended to incorporate knowledge regarding actions (in a similar fashion to Potential-Based Advice (Wiewiora et al. 2003)), although this has not yet been empirically tested.

2.9 Multi-Objective Multi-Agent Systems

The majority of MAS research focuses on optimising systems with respect to a single objective, despite the fact that many real world problems are inherently multi-objective in nature. This section discusses related work that takes a multi-objective perspective on cooperative MAS, with a specific focus on the works that are most closely related to the contributions of this thesis.

Brys et al. (2014) applied MORL to a traffic signal control problem, where each intersection in a 2×2 grid is controlled by an individual agent. Their work demonstrated that rewarding agents with a linear scalarised combination of delay and throughput improved delay times when compared to agents rewarded using delay alone. However, their approach uses local rewards (i.e. each agent is rewarded based on conditions at its assigned intersection only), and does not make any attempt to encourage coordination between the agents.

Van Moffaert et al. (2014) tested MORL on a multi-objective multi-agent smart camera problem. They developed an adaptive weight algorithm (AWA) which is used to choose the weighting between the two system objectives when linear scalarisation is applied. The AWA algorithm was found to improve learning speed, obtaining improved solutions in terms of hypervolume and a better spread in the objective space, when compared with other weight selection methods that were tested.

Taylor et al. (2014) proposed Parallel Transfer Learning (PTL) as a mechanism to accelerate learning, by sharing experience among agents. PTL was tested on a multi-objective multi-agent smart grid problem, and was found to improve learning speed and final performance when compared to agents learning without PTL.

In contrast to the works above, Roijers et al. (2013, 2014, 2015) do consider the effect of encouraging coordination in their research on cooperative multi-objective MAS. Their approach makes use of multi-objective coordination graphs (MO-CoGs) which model the interactions between agents. MO-CoGs are used to calculate joint actions that will cover the Pareto front, or a subset of the Pareto front. Multiple individual learners are used throughout this thesis, and therefore models of the agents' interactions are not required; instead, coordination between agents will be encouraged through reward function engineering.

None of the works discussed thus far considered the effect of credit assignment in multi-objective MAS. Yliniemi (2015) and Yliniemi & Tumer (2016) present the first work that considers the design of reward structures in a MOMARL setting. Their work compared the effectiveness of D with that of the typical MARL reward structures L and G. Experimental work conducted in a multi-objective congestion problem, and a multi-objective robot coordination problem confirmed that D can improve MOMARL performance when compared to L or G, both in terms of learning speed and the quality of the non-dominated solutions found. Yliniemi & Tumer (2016) also demonstrated that D can be used effectively with multi-objective GAs, in a series of experiments where it was applied to shape the fitness function of NSGA-II.

2.10 Summary & Contributions

In summary, the most important concepts introduced in this section are:

- **Agents**: An agent is an autonomous entity capable of acting on environmental stimuli.
- **Multi-Agent Systems**: Systems which consist of multiple agents sharing a common environment.
- **Reinforcement Learning**: A process whereby an agent learns to maximise the rewards received over its lifetime.
- **Multi-Agent Reinforcement Learning**: A paradigm which consists of multiple RL agents situated in a common environment.
- **Nash Equilibria**: A Nash equilibrium occurs when no agent in a MAS can improve its reward by altering its policy, assuming that all other agents in the MAS continue to follow their current policies.
- **Multi-Objective Reinforcement Learning**: The adaption of the RL process to explicitly account for the presence of multiple system objectives.
- **Pareto front**: The set of solutions which represent the optimal tradeoffs between objectives in a MORL domain (also referred to as the non-dominated set).
- **Knowledge-Based Reinforcement Learning**: A family of techniques which aim to improve the performance of RL agents by providing additional domain knowledge.
- **Reward Shaping**: A KBRL technique which provides additional knowledge to RL agents by modifying the reward function.
- **Potential-Based Reward Shaping**: A form of reward shaping where the additional reward is calculated as the difference in potential between two states or state-action pairs.
- **Difference rewards**: A form of reward shaping specific to cooperative MAS, where agents are rewarded based on their individual contribution to the system utility.

The investigations documented later in this thesis will build upon these key concepts, extending prior theoretical work on reward shaping to cover MORL domains. To recap, these prior guarantees include policy invariance in MDPs (Ng et al. 1999) and consistent Nash equilibria in SGs (Devlin & Kudenko 2011) when agents receive $PBRS$. Other theoretical work confirms that these guarantees hold when a dynamic potential function is used (Devlin & Kudenko 2012), and when the application domains are episodic in nature (Devlin 2013; Grześ 2017). In the case

of D, the preexisting guarantees are weaker; it has been proven that the relative ordering of expected returns (and therefore the Nash equilibria) are not altered when agents are rewarded using D instead of G in two-player single-objective matrix games (Colby & Tumer 2015).

This thesis will consider the issue of credit assignment in multi-objective MAS, where each agent is an individual Q-learner. Clearly, the most closely related empirical works discussed in this chapter are those of Yliniemi (2015) and Yliniemi & Tumer (2016). While these works have already demonstrated the importance of appropriate credit assignment and the effectiveness of D in MOMARL, they lack:

- A theoretical evaluation of the effect of D in MOSGs. Specifically, the question of whether applying D to MOSGs will alter the true Pareto optimal joint policies remains open.
- An empirical evaluation of D in a MOSG where the true Pareto optimal system utilities are known.
- Empirical comparisons between D and alternative reward shaping methods for MOMARL.

In contrast to these works, this thesis will consider the issue of credit assignment for Reinforcement Learners in MOSGs using two popular reward shaping techniques: D and $PBRS$. Theoretical analysis of the effects of these reward shaping techniques which justifies their use in MOSGs (Chapter 4) is supported by empirical studies in domains where the true Pareto optimal solutions are known, and in a realistic application domain (Chapter 5).

CHAPTER 3

Designing Potential Functions

As discussed in the previous chapter, the question of how to design useful potential functions is a significant challenge that must be overcome when implementing $PBRS$. This chapter contributes empirical studies that aim to address this question from three different perspectives:

1. Section 3.1 describes the process involved in designing a useful potential function for a real world application, namely Traffic Signal Control. A case study involving three different intersections is presented, and empirical results confirm that $aPBRS$ with a suitable heuristic can significantly improve the performance of RL agents in this domain.

2. In Section 3.2, a method to automatically generate potential functions for co-operative MAS is proposed. Estimated Counterfactual as Potential is evaluated using the Beach Problem domain, and is found to achieve similar levels of performance to those of the analytically computed Counterfactual as Potential method.

3. Finally, Section 3.3 investigates the effect of encoding the same heuristic knowledge in either state-based or action-based formats. Experiments are conducted in a new congestion game, the Shepherd Problem Domain. This study represents the largest scale evaluation of $aPBRS$ to date, in a MAS with 100 agents (the previous largest study by Devlin et al. (2011a) had 9 agents). The experimental results offer new insights into the behaviour of heuristics when encoded in different formats, as well as demonstrating that the beneficial effects of $aPBRS$ with a good heuristic do in fact scale up to larger MAS.

3.1 Traffic Signal Control Domain

This section will demonstrate the steps required to successfully apply $PBRS$ to a complex real world problem, namely Traffic Signal Control (TSC). Traffic congestion is one of the major issues currently faced by modern cities. The negative environmental, social and economic consequences of urban traffic congestion are well documented. High vehicle usage rates, along with the lack of space and public funds available to construct new transport infrastructure add to the significant challenges currently faced by traffic engineers. Against this backdrop, it is now necessary to develop intelligent and economical solutions to improve the quality of service for road users.

One relatively inexpensive way to alleviate the problem is to ensure optimal use of the existing road network, e.g. using Adaptive Traffic Signal Control (ATSC). Continued improvements in ATSC will have an important part to play in the future development of Smart Cities, especially in light of the current EU-wide emphasis on the theme of Smart, Green and Integrated Transport in Horizon 2020[1]. Developing ATSC strategies for efficient urban traffic management is a challenging problem, and one which is not easily solved.

In recent years, AI methods such as Fuzzy Logic, Neural Networks, Genetic Algorithms (GAs) and RL have all been applied successfully to traffic control problems. These developments coincide with an increasing interest among researchers in the broader field of Intelligent Transportation Systems (ITS). In Reinforcement Learning for Traffic Signal Control (RL-TSC), each intersection is typically controlled by a single autonomous agent. Each agent has the responsibility of determining the light switching sequence at its assigned intersection, and learns a control policy by a process of continuous interaction with its environment. A network of traffic signal control agents may be considered as a MAS, which opens up possibilities in relation to developing agent coordination strategies to reach a global rather than local optimum.

RL-TSC approaches offer many benefits; RL agents have the capability to learn online to continuously improve their performance and thus adapt readily to changes in traffic demand patterns, and are capable of dealing with the incomplete information and stochastic nature inherent in this problem domain. Traffic control problems are a very attractive testbed for emerging RL approaches (Bazzan 2009), because they present a number of interesting challenges such as developing strategies for coordination and information sharing between individual agents. The complexity and uncertainty of traffic signal control problems make them an extremely interesting application area for AI researchers to investigate.

The number of possible state action combinations for complex intersections with many phases present a significant challenge when applying RL to traffic signal control. RL literature refers to this problem as the Curse of Dimensionality. As RL agents are presented with increasingly complex problems, convergence times and the quality of the policy learned tend to degrade. When dealing with very large state action spaces, it may not be possible for an agent to sample

[1] See http://ec.europa.eu/programmes/horizon2020/en for further details

each state-action pair sufficiently often to learn a good policy within a reasonable timeframe. In general, as the problem complexity is increased an agent will require more experience in order to learn a good policy, which necessitates more training time. $PBRS$ has the potential to mitigate against the problems described above by guiding an agent's exploration, thus improving both learning speed and performance.

3.1.1 Reinforcement Learning for Traffic Signal Control

Numerous authors have studied the application of Reinforcement Learning to Traffic Signal Control problems in the last two decades, coinciding with an increased interest in ITS among researchers. Thorpe & Anderson (1996) presented some of the earliest work on RL-TSC, in which an approach based on SARSA was proposed. This algorithm was benchmarked against a fixed time control scheme, and was found to offer significant performance increases compared to the latter approach.

Wiering (2000) developed a model-based approach to RL-TSC, and found that the RL systems clearly outperformed fixed time controllers at high levels of network saturation, when testing on a simple 3 x 2 grid network.

Dresner & Stone (2006) suggest the use of RL in combination with their Autonomous Intersection Management architecture. Here an intersection is treated as a marketplace where vehicles pay for passage or pay a premium for priority, and the RL agent's goal is to maximise the revenue collected. Thus in future, revenue collected could be used in reward functions for RL-TSC.

Abdoos et al. (2014) developed an RL-TSC approach based on Q-learning using tile coding as a method of Q function approximation. Their approach consists of an agent controlling each intersection, and these agents are grouped together under the control of superior agents. This hierarchical approach was tested against a standard Q-learning approach on a 3 x 3 grid of intersections, and was found to reduce delay times in the network.

Pham et al. (2013) present an RL-TSC system based on SARSA which also uses tile coding as a method of Q function approximation. In contrast to the approach above, in this system each SARSA agent is completely independent, and tile coding is used as a method of approximating the value function for the agent's local states.

El-Tantawy et al. (2013) proposed MARLIN-ATSC, which is a coordinated multi-agent RL-TSC architecture. This is a model-free approach based on Q-learning, where the state definition is based on queue length, and the reward definition is based on Total Cumulative Delay. The system is tested on a simulated network of 59 intersections in Downtown Toronto, and outperformed the currently implemented real world control scheme, resulting in a reduction in average delay, average stop time, travel times, queue lengths and emissions.

Brys et al. (2014) observed in their experiments that the objectives throughput and delay are correlated. They implemented a multi-objective RL-TSC algorithm, where the single-objective reward signal is replaced with a scalarised signal, which was a weighted sum of the reward due

to both objectives. They report that the proposed multi-objective approach exhibits a reduced convergence time, as well as decreasing the average delay in the network when compared to a single objective approach.

Mannion et al. (2015a,b) proposed a Parallel Reinforcement Learning algorithm for TSC. This framework allows multiple agents to learn in parallel on separate instances of the same TSC problem while sharing experience, with the goal of improving learning speed and exploration. The algorithm was tested on three intersections of varying complexity, and was found to offer statistically significant reductions in delay times and queue lengths as well as increasing learning and exploration rates when compared to a standard single RL agent approach.

Work by van der Pol & Oliehoek (2016) combined Deep Q-learning with transfer planning to develop a scalable approach for multi-agent traffic light control. However, they found that the performance of the Deep Q-learning algorithm can oscillate in this application domain, as with earlier work on Deep Reinforcement Learning in other problem domains.

For a more comprehensive review of the usage of RL agents in Traffic Signal Control beyond the brief summary presented here, the interested reader‘is referred to a review paper published by Bazzan & Klugl (2014).

3.1.2 Learning Traffic Signal Control with Advice

Traffic Signal Control presents a number of significant challenges when compared with the abstract problem domains (e.g. Gridworld) traditionally studied by RL researchers, due to the high degree of complexity and stochastic behaviour exhibited. In the simplest 2 phase Traffic Signal Control scenario considered in this study, the number of possible discrete system states is of the order of 1.8×10^4, rising to approximately 8×10^5 states for the 3 phase case. By contrast, a typical 50×50 Gridworld experiment has only 2.5×10^3 states. Traffic Signal Control is a infinite-horizon optimisation problem with no terminal goal state, whereas many traditional abstract problem domains have a goal state that an agent must reach.

The scale of transportation networks and the number of independent entities involved mean that an RL agent cannot possibly keep track of every detail about the environment state; therefore these problems are classified as Partially Observable Markov Decision Processes. Actions in Traffic Signal Control problems are not deterministic - i.e. the agent's action choice in a specific state is not the only factor that determines the next system state. Every additional variable considered in the representation of the environmental state increases the number of possible state action combinations, and transportation optimisation problems are thus challenging application domains for RL agents. The use of model-based RL algorithms in a highly stochastic problem domain like traffic control has been found to add unnecessary extra complexity when compared with model-free techniques (El-Tantawy et al. 2013). For this reason a model-free RL method has been applied to this domain, namely Q-learning.

The agents tested in this study will be identical in all respects, except that one of the agents

receives a shaped reward signal, and the other does not. The state, action and reward function definitions used for agents in this study are similar to those used in previously published works in RL-TSC (Abdulhai et al. 2003; El-Tantawy & Abdulhai 2010, 2012; El-Tantawy et al. 2013). Thus, the agent learning with shaping in this study can be considered to be an extension of these state-of-the-art works. RL-TSC agents using these state, action and reward definitions have already been proven to offer considerable performance improvements compared to real world traffic control systems based on fixed-time control, semiactuated control, and SCOOT control (El-Tantawy et al. 2013). In this study, the agent learning with shaping is evaluated against this existing and proven approach, as other authors have already dealt with the efficacy of RL versus other techniques for TSC in a comprehensive manner.

Traffic engineers use the term phase to refer to a specific traffic movement through an intersection, and determining an appropriate phasing sequence (order and duration in which phases move through the junction) is the main goal in TSC. The environmental state is defined as a vector of dimension 2 + P, shown formally in Eqn. 3.1 below, where P is the number of phases at the junction. The first two components in the state definition are the index of the current phase (P_c) and the elapsed time in the current phase (PTE), while the remaining P components represent the queue lengths (QL_i) for each phase at the junction. The state vector is thus constructed as follows for a given state s:

$$s = \Big[P_c, \quad PTE, \quad QL_1, \quad ..., \quad QL_n \Big] \tag{3.1}$$

By making use of a mixed radix conversion the state vector for each possible state is represented as a single number, which is used when setting and retrieving values in the Q values table.

The maximum number of queueing vehicles considered is limited to 20, and the maximum phase elapsed time considered is limited to 30 seconds. By imposing these limits, the number of possible environmental states considered by an agent is reduced. Even with these limits, over 18,500 discrete states are possible for a two phase junction. A vehicle is considered to be queueing at a junction if its approach speed is less than 10 km/hr. Limiting the number of queueing vehicles an agent senses to 20 adds further realism to these experiments, as in practice it would be prohibitively complex and expensive to detect queueing vehicles along the entire length of the approach lane.

At each time step t, the actions available to the agents are: to keep the currently displayed green and red signals, or to set a green light for a different phase. To eliminate unreasonably low durations from consideration, phases are subject to a minimum length. In the case of the 2 phase test junction, the minimum phase length is 10 seconds, while the minimum phase length for the 3 and 4 phase intersections is 5 seconds. Agents are free to extend the current phase or switch to the next phase as they see fit, and there is no fixed cycle length. If an agent decides to switch phases, an amber signal is displayed for 3 seconds, followed by an all red period of 2 seconds,

followed by a green signal to the next phase. This adds greater realism as it accounts for lost time due to phase switching, along with reducing the chances of vehicle collisions occurring.

Each agent selects actions using the ϵ-greedy strategy, where a random action is chosen with probability ϵ, or the action with the best expected reward is chosen with the remaining probability $1 - \epsilon$. The reward function used by all agents is shown in Eqn. 3.2 below. When an agent selects an action a in a given state s and transitions to a resultant state s', the reward received is defined as the difference between the current and previous cumulative waiting times (CWT) of vehicles queueing at the junction. Therefore, actions that decrease the cumulative waiting time receive a positive reward, while actions that increase the cumulative waiting time incur a negative reward (or penalty).

$$R(s, a, s') = CWT_s - CWT_{s'} \tag{3.2}$$

As agents are rewarded for reducing the total waiting time at a junction, it follows that agents will try to select actions that reduce waiting time as much as possible. When designing a potential function, it is important to keep the goal that is specified in the reward function in mind; therefore the potential function chosen for this application should include domain knowledge that will aid an agent in achieving its goal of reducing waiting time. Perhaps the simplest heuristic that could be used in a TSC scenario is a longest queue first rule; i.e. a heuristic that encourages agents to give more green time to the junction approaches with the longest queues. As this heuristic can be used to encourage specific actions, $aPBRS$ was used in this implementation. The potential function for any state action pair can easily be generated as shown:

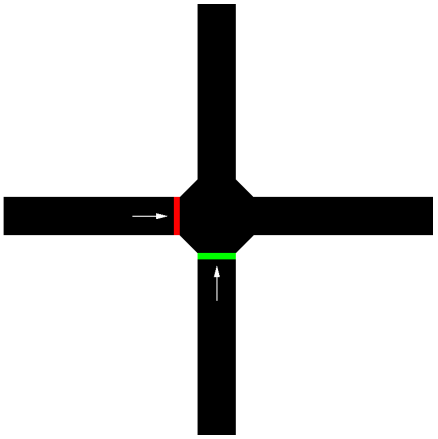
$$\Phi(s, a) = \frac{QL_a}{\Sigma QL} \tag{3.3}$$

where QL_a is the queue length corresponding to the phase a, and ΣQL is the sum of the queue lengths for all of the phases at an junction. This definition assigns a higher potential to phases with a higher proportion of the total vehicles queuing at an intersection. Therefore, an agent learning with $aPBRS$ and this potential function will be encouraged to give more green time to phases that have a higher number of vehicles waiting than the other phases. This shaping is expected to improve the learning speed of all agents receiving it, when compared to agents learning without shaping.

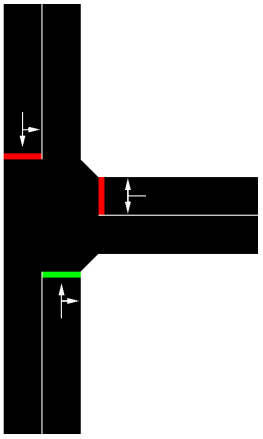
3.1.3 Experimental Procedure

The experimental setup is based around the microscopic traffic simulation package SUMO (Simulation of Urban MObility) (Krajzewicz et al. 2012), and agent logic is defined externally. The TraaS[2] library is used to make simulation parameters available to the agents, and also to send signal switching instructions from the agents back to SUMO. The simulation timestep length is set to 1 second for all experiments.

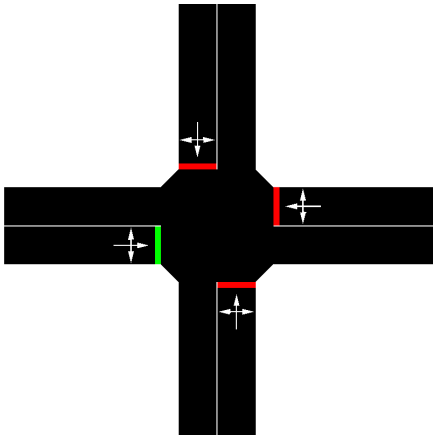
[2]The TraaS library is available from http://traas.sourceforge.net/cms/



(a) 2 phase



(b) 3 phase



(c) 4 phase

Figure 3.1: Junction configurations

All agents begin each experiment with their Q values for each state action pair initialised to zero. The values used for the learning rate α, discount factor γ and exploration rate ϵ are 0.08 and 0.8 and 0.05 respectively. All agents tested use these values of α, γ and ϵ, which were selected following parameter sweeps to determine the best performing values. Specifically, these parameter sweeps were conducted by first running the experiments using reference values for each parameter (e.g. $\alpha = 0.1$, $\gamma = 0.9$ and $\epsilon = 0.1$), and adjusting each parameter in turn to improve performance. This method of manual parameter tuning was used for all the experimental work in this thesis; however, it should be noted that this process could potentially be automated using meta-optimisation (i.e. using another optimisation algorithm to determine the best learning paramenters for the agents).

The RL agents learning with and without $aPBRS$ are evaluated experimentally using three different scenarios, which are based on the intersections shown in Figs. 3.1a to 3.1c. The number of phases, lane configuration, and traffic demand levels differ for each test intersection. All experiments are conducted on isolated intersections; work by other researchers has already proven the efficacy of RL-TSC approaches in test scenarios with multiple intersections (see e.g. El-Tantawy et al. (2013); Pham et al. (2013); Abdoos et al. (2014); Brys et al. (2014)).

The traffic demand Y (measured in vehicles per hour) at each junction is generated using a step function. The demand is calculated as a function of episode length e, base flow b (baseline flow of vehicles through the junction), step demand increase h (the additional demand introduced at each step in the function), and a step interval δ (duration in seconds between demand steps). Thus the demand at any time t into a particular episode can be calculated according to Eqn. 3.4 below:

$$Y(t) = \begin{cases} b + \dfrac{h \times t}{\delta} & t < \dfrac{e}{2} \\ b + h \times \left(\dfrac{e}{2 \times \delta} - 1 - \dfrac{t - 0.5 \times e}{\delta} \right) & \text{otherwise} \end{cases} \tag{3.4}$$

The increase in demand due to this function is computed at intervals equal to δ. This time-varying traffic demand presents a more challenging flow pattern for the agents to control, compared to a constant hourly demand definition. These step functions aim to emulate peaks in demand similar to those during a morning or evening rush hour. Agents are trained on a junction for a number of successive 2 hour episodes, and the same demand step function is repeated during each episode. An agent then builds up experience gradually over the training duration. Agents are trained for a period of 75 episodes on each intersection. There are a fixed number of possible routes through each intersection, and vehicles are randomly assigned to one of the possible routes upon insertion into the network.

The first test case is a simplified junction with 2 phases: North and East (see Fig. 3.1a). Here two intersecting one-way streets are controlled by a set of traffic lights, with the number of phases

Table 3.1: Waiting Time (averaged over final 10 episodes)

Experiment	AWT	% Reduction
2 phase	41.28	-
2 phase $aPBRS$	36.96	10.46 %
3 phase	102.11	-
3 phase $aPBRS$	100.46	1.62 %
4 phase	171.99	-
4 phase $aPBRS$	162.92	5.27 %

Table 3.2: Queue Length (averaged over final 10 episodes)

Experiment	AQL	% Reduction
2 phase	12.75	-
2 phase $aPBRS$	11.67	8.52 %
3 phase	17.13	-
3 phase $aPBRS$	16.53	3.49 %
4 phase	19.73	-
4 phase $aPBRS$	19.70	0.15 %

$P = 2$. The base flow for the step function is 1000 vehicles per hour (veh/hr). The demand level rises by 250 veh/hr every 15 minutes, reaching a peak of 1750 veh/hr, before stepping down again to a value of 1000 veh/hr. There are four possible routes through this intersection.

The second intersection is a T junction, with three intersecting two-way streets (shown in Fig. 3.1b). There are 6 possible routes through this junction, and three phases: North, East and South. The base flow is set to 1000 veh/hr, rising by 100 veh/hr every 15 minutes to a peak of 1300 veh/hr, before returning to 1000 veh/hr.

The final scenario joins four streets with two-way traffic (see Fig. 3.1c). This is divided into four phases: North, East, South and West. Here there are 12 possible routes through the intersection. The base flow is set to 1000 veh/hr, which rises by 100 veh/hr every 15 minutes, giving a peak demand of 1300 veh/hr before reducing to the original value of 1000 veh/hr.

All plots include error bars representative of the standard error of the mean based on 10 statistical runs at 5 episode intervals. Specifically, the error is calculated as $\sigma/\sqrt{n}$ where σ is the standard deviation and n is the number of statistical runs. 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.

3.1.4 Experimental Results

Figs. 3.2 and 3.3 plot the Average Waiting Time (AWT) and Average Queue Length (AQL) for the agents tested on each of the three intersections, averaged over 10 statistical runs. The results for AWT and AQL are summarised in Tables 3.1 and 3.2.

The plots show a distinct improvement in learning speed for the agents receiving $aPBRS$, compared to the agents learning without shaping. This difference is especially noticeable in the AWT plots, where there is a clear separation between the learning curves for agents with and without shaping in all three test scenarios. In the AQL plots, the beneficial effect of $aPBRS$ is also evident in the 2 phase and 3 phase tests, although it is less noticeable in the 4 phase test.

The agents receiving $aPBRS$ reached a better policy on all test intersections by the end of the 75 episode training period when compared to the unshaped agents. Reductions of up to 10% in waiting times were observed for the 2 phase test case at the end of the training period, with noticeable reductions for the 3 and 4 phase intersections also.

The mean values of AWT and AQL over the final 10 episodes of each experiment were tested for statistical significance. In all three test scenarios, the agents receiving $aPBRS$ were found to have statistically better AWT performance that the unshaped agents at the end of the 75 episode training period. The reductions in AQL for both the 2 and 3 phase intersections were also deemed to be statistically significant. In the case of the 4 phase junction, there was no significant difference between the mean queue lengths achieved by agents learning with and without $aPBRS$ at the end of the training period.

The improvements in learning speed demonstrate how $aPBRS$ with a potential function using very simple domain knowledge can improve RL performance in a complex real world application. The longest queue first rule is an example of an approximate heuristic; i.e. it provides a general guide about how to behave, and is not meticulously hand-coded to encourage the best action in every possible situation. Nevertheless, it encourages beneficial actions often enough to improve performance in all scenarios tested.

The effect of $aPBRS$ on learning speed is due the bias it introduces towards the action with the highest potential; recall from Eqn. 2.14 that agents learning with $aPBRS$ greedily select the action which maximises the sum of Q value and potential. During the initial stages of learning when all unseen state action pairs have a Q value equal to zero, this bias confers an advantage to the agents receiving $aPBRS$. In cases where the heuristic damages performance (i.e. waiting time is increased), a sub-optimal action is less likely to be selected in the future due to a decreased sum of potential and Q value. These results demonstrate that learning with an approximate heuristic is preferable to learning without any prior knowledge; provided of course that the heuristic contains at least some useful information.

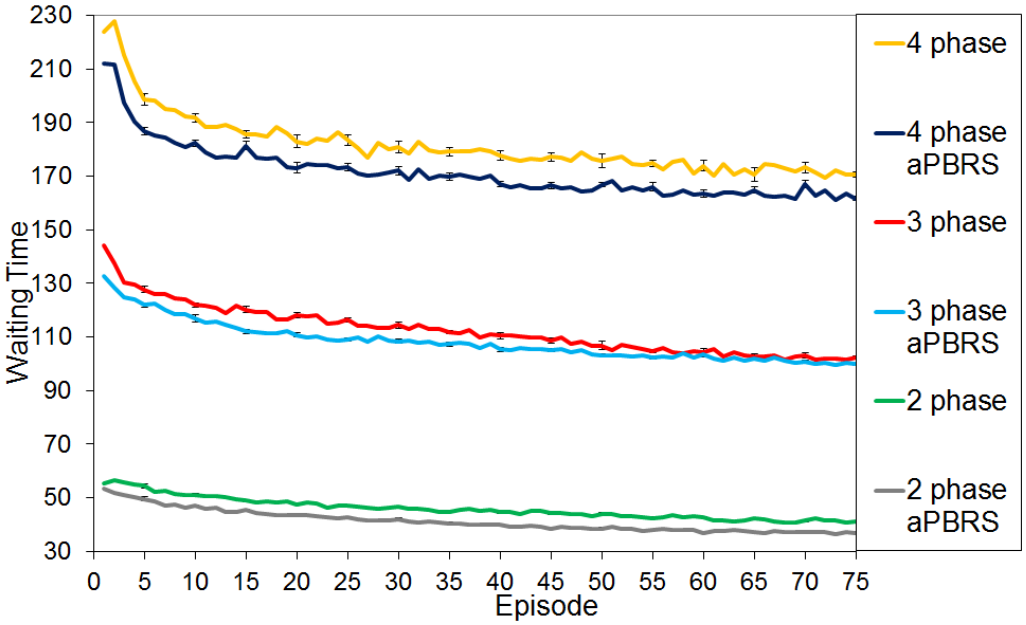


Figure 3.2: Average Waiting Time

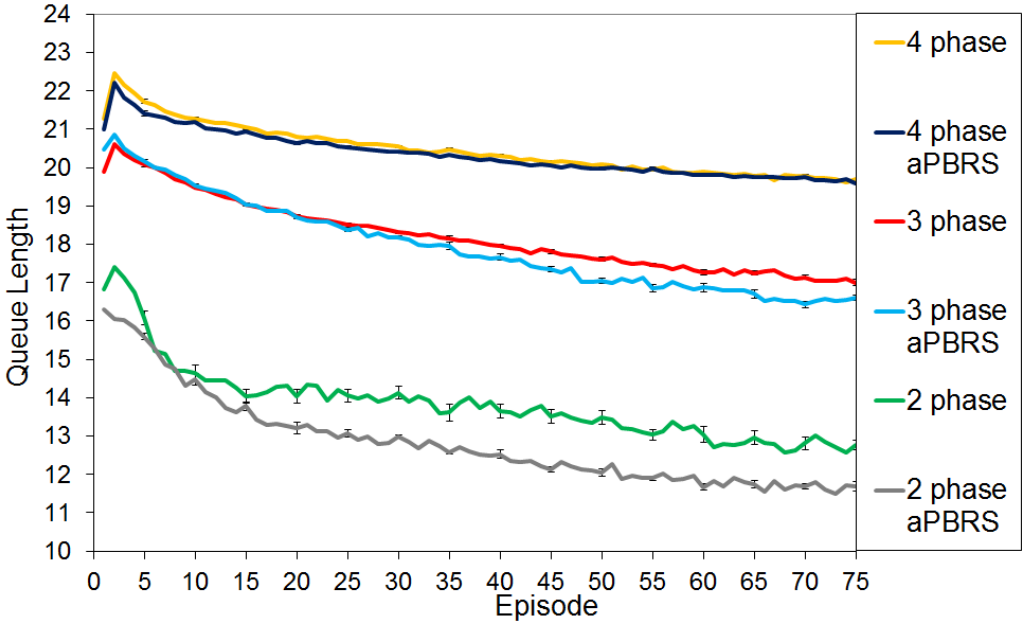


Figure 3.3: Average Queue Length

3.1.5 Discussion

This study demonstrated how $aPBRS$ can be applied to a realistic Traffic Signal Control problem. A simple longest queue first heuristic was encoded into an action-based potential function, which improved both the learning speed and final performance of all agents receiving $aPBRS$ in three different test scenarios. The agent receiving $aPBRS$ extended and improved upon a published state-of-the-art RL-TSC agent, demonstrating the benefit of incorporating domain knowledge in RL applications via $PBRS$.

The simple longest queue first heuristic was most effective in the 2 phase test, and was somewhat less useful in the 3 and 4 phase tests. This is most likely due to the fact that this heuristic encourages agents to switch phases quite often. As the number of phases at an intersection increases, the effect of lost time (amber and red time during phase changes) becomes more significant; therefore switching phases too frequently limits the efficiency of an intersection. Potential functions which encourage agents to use longer phase durations may therefore be more effective than the simple longest queue first heuristic for junctions that have many phases and/or a high level of vehicular traffic.

It would also be worthwhile to test $aPBRS$ on simulated real world traffic networks with multiple signalised intersections. More advanced potential functions could be developed for multi-agent TSC scenarios which explicitly encourage cooperation among agents in the same locality, to ensure safe and efficient control at a network level, and to aid in the creation of green waves.

3.2 Generating Potential Functions using Counterfactual Estimates

As discused in the previous chapter, several efforts have been made to remove or reduce the need to hand code potential functions for each new $PBRS$ application. Counterfactual as Potential (Devlin et al. 2014) is one such method, where potentials are generated using the counterfactual term computed in a difference evaluation (Eqn. 2.21).

CaP benefits from the theoretical guarantees of $PBRS$ and has been demonstrated to increase performance in MAS (Devlin et al. 2014), but unfortunately it suffers from the same limitations as D: global knowledge about the system state and joint action must be available, and the precise form of the system evaluation function G must be known in order to calculate the counterfactuals for each agent. To implement CaP successfully, each agent requires the value of G and the value of its individual counterfactual to be broadcast to it.

Colby et al. (2016) attempted to address these inherent limitations of D by approximating the counterfactual term, giving an estimated difference reward $\hat{D}$ (Eqn. 2.22). A simple tabular approximation of the counterfactual term can be maintained by each agent using a temporal difference update (as shown in Eqn. 2.23). While $\hat{D}$ has proven to be successful in shaping agents'

behaviour in several problem domains, and requires much less information than D for successful implementation, it does not provide any theoretical guarantees. Here Estimated Counterfactual as Potential ($\hat{C}aP$) is proposed, a method for generating multi-agent potential functions using counterfactual estimates. Formally:

$$\Phi_i(s_i) = \hat{G}_i(s_i^c, a_i^c) \tag{3.5}$$

where $\Phi_i(s_i)$ is the potential for state s_i for agent i, $\hat{G}_i(s_i^c, a_i^c)$ is the approximated counterfactual for agent i (calculated using Eqn. 2.23), and s_i^c and a_i^c are default states and actions chosen for a specific application.

In this framework, the unshaped reward $R = G$, and the shaping reward F is calculated as usual when using $sPBRS$ (Eqn. 2.12). The goal of this approach is to replicate the performance of CaP while removing the need for precise knowledge about the form of the system evaluation function and the global system state and joint action.

$\hat{C}aP$ assumes that only the value of $G(s, a)$ is broadcast to all agents at each timestep, and that each agent maintains a private approximation $\hat{G}_i(s_i, a_i)$ of the system evaluation function in order to generate their own potential function. Furthermore, $\hat{C}aP$ benefits from the theoretical guarantees of $PBRS$, whereas no such guarantees are available for $\hat{D}$.

Just as CaP does not replicate the behaviour of D, $\hat{C}aP$ is not expected to replicate the behaviour of $\hat{D}$. I hypothesise that $C\hat{a}P$ will be a more informative learning signal than G alone, and will therefore improve system performance when combined with G. Typical $PBRS$ implementations require the system designer to specify a bespoke potential function manually for each new application domain, which can be a time-consuming task. $\hat{C}aP$ addresses this challenge by offering a semi-automated method to generate multi-agent potential functions that can be applied to any cooperative MAS, once suitable default states and actions are chosen by the system designer.

3.2.1 Beach Problem Domain

The Beach Problem Domain (BPD) (Devlin et al. 2014) is a congestion game in which the agents must coordinate their actions to achieve the maximum global utility. It extends the classic El-Farol Bar Problem (EBP) (Arthur 1994) by including multiple states. In the BPD, each tourist (agent) begins at a hotel on a specific beach section, and then decides at which section of the beach they will spend their day. At each timestep each agent knows which beach section $b \in B$ it is currently attending, and can choose to move to an adjacent section ($move\_left$ or $move\_right$), or to $stay\_still$. Once all agents have completed their selected actions they are rewarded.

Each beach section has a certain capacity ψ, and the reward for a beach section is maximised when the number of agents attending that section is equal to the capacity. Sections which are over or under capacity receive a lower reward, reflecting the decreased desirability of beaches which

are too crowded or too empty. When the total number of agents is greater than the combined capacity of all beach sections this constitutes a congestion problem, and the highest system utility is achieved when any one of the beach sections is overcrowded by most of the agents, and exactly ψ agents attend each of the other beach sections. The local reward function (L) for a beach section is calculated as:

$$L(b,t) = x_t e^{\frac{-x_{b,t}}{\psi}} \tag{3.6}$$

where b is the beach section (local state), $x_{b,t}$ is the attendance at that beach section at time t, and ψ is the capacity of the beach section. The global reward or capacity utility can then be calculated as the summation of $L(b,t)$ over all sections of the beach:

$$G(t) = \sum_{b \in B} L(b,t) \tag{3.7}$$

The difference reward (D_i) for an agent can be directly calculated by applying Eqn. 2.21; every section not currently attended by the agent at a particular timestep cancels out, as the agent has no impact on their evaluations, leaving:

$$D_i(t) = L(b,t) - (x_{b,t} - 1)e^{\frac{-(x_{b,t}-1)}{\psi}} \tag{3.8}$$

where $x_{b,t}$ is the number of agents attending the same beach section b as agent i at timestep t. This evaluation is precisely equivalent to applying Eqn. 2.21 to Eqn. 3.7 directly. In a similar manner, the potential function for CaP may be calculated as:

$$\Phi(b) = (x_{b,t} - 1)e^{\frac{-(x_{b,t}-1)}{\psi_b}} \tag{3.9}$$

The potential function for $\hat{C}aP$ is calculated as per Eqn. 3.5 above. The process followed in the BPD when agents are rewarded using $G + \hat{C}aP$ is described in detail in Algorithm 1.

3.2.2 Experimental Procedure

Two different empirical studies were conducted in the BPD, the first with $num\_timesteps = 1$ and $\gamma = 0.9$, and the second with $num\_timesteps = 5$ and $\gamma = 1.0$. In all experiments, the number of agents is set to $N = 100$, the number of beach sections is set to $|B| = 5$, the capacity for each section is set to $\psi = 7$, $num\_episodes = 10000$, $\alpha = 0.1$ with $alpha\_decay\_rate = 0.9999$, and $\epsilon = 0.05$ with $epsilon\_decay\_rate = 0.9999$. All $Q(s,a)$ values for all agents are initialised to -1 at the start of each experimental run. The first $N/2$ agents begin each episode at beach section 1, while the rest begin at beach section 3. These parameters were chosen in order to replicate the experimental setup used by Devlin et al. (2014) in the work where CaP was originally proposed.

Multiple individual Q-learning agents were applied to the BPD, using the ϵ-greedy explora-

Algorithm 1 Beach Problem Domain with $G + \hat{C}aP$

initialise Q-values: $\forall b, a | Q(b, a) = -1$
initialise $\hat{G}$-values: $\forall b, a | \hat{G}(b, a) = 0$
initialise potentials: $\forall b | \Phi(b) = 0$
for $episode = 1 \rightarrow num\_episodes$ **do**
    for $timestep = 1 \rightarrow num\_timesteps$ **do**
        for $i = 1 \rightarrow num\_agents$ **do**
            sense current beach section b
            choose action a, using ϵ-greedy
            move agent to b'
        end for
        for all beach sections $b \in B$ **do**
            evaluate local reward $L(b)$ (Eqn. 3.6)
        end for
        evaluate global reward G (Eqn. 3.7)
        for $i = 1 \rightarrow num\_agents$ **do**
            update $\hat{G}(b, a)$ value (Eqn. 2.23)
            set $\Phi(b')$ (Eqn. 3.5)
            set F (Eqn. 2.12)
            set $R' = G + F$ (Eqn. 2.10)
            update $Q(b, a)$ value (Eqn. 2.2)
        end for
        reduce ϵ by multiplication with $epsilon\_decay\_rate$
        reduce α by multiplication with $alpha\_decay\_rate$
    end for
    for $i = 1 \rightarrow num\_agents$ **do**
        choose action a, using ϵ-greedy
        move to absorbing state
        set $F = 0 - \Phi(b')$ (Eqn. 2.12)
        set $R' = 0 + F$ (Eqn. 2.10)
        update $Q(b', a)$ values (Eqn. 2.2)
    end for
end for

tion strategy and learning using the credit assignment structures L, G, $G + CaP$, $G + \hat{C}aP$, D, and $\hat{D}$. When testing $G + \hat{C}aP$ and $\hat{D}$, all $\hat{G}_i(s_i, a_i)$ values were initialised to 0 with $\beta = 0.1$ and $beta\_decay\_rate = 0.9999$. The default state s_i^c was set to the agent's starting state, and the default action a_i^c was set to $stay\_still$.

All plots show the average value of the system utility, and include error bars representative of the standard error of the mean based on 50 statistical runs at 1000 episode intervals. Specifically, the error is calculated as $\sigma/\sqrt{n}$ where σ is the standard deviation and n is the number of statistical runs. 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.

3.2.3 Experimental Results

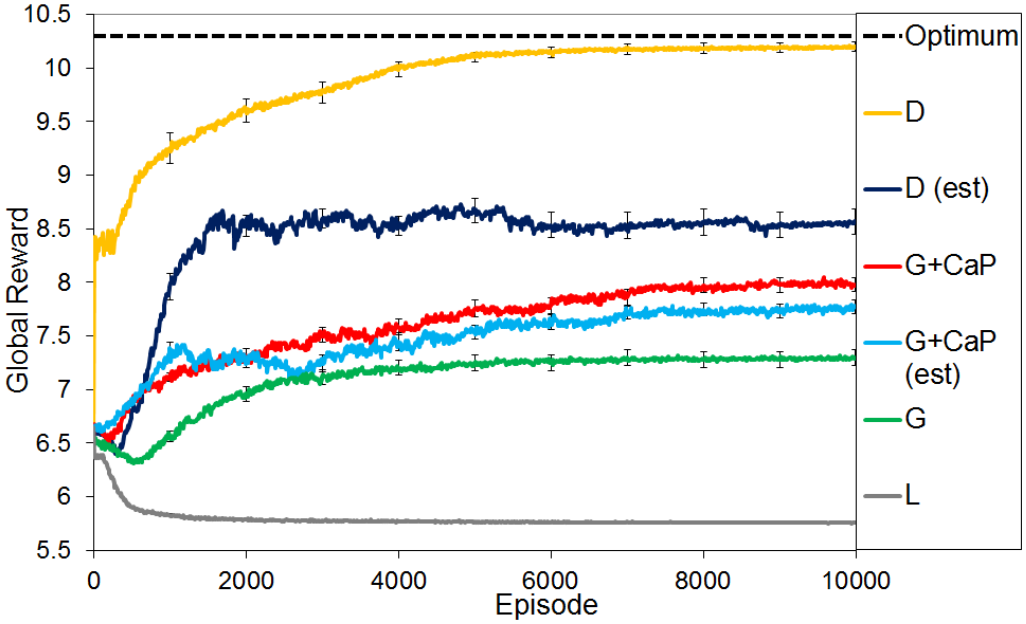
The learning curves for the approaches tested are shown in Figs. 3.4a and 3.4b, and the average performance of each approach over the last 1000 learning episodes is shown in Table 3.3. The maximum possible global reward or system utility for the parameters used is 10.30; this is shown with a black dashed line in both figures.

As expected, L offers the poorest performance of all approaches tested, as it does not effectively encourage the agents to act in the system's best interest. G offers increased performance compared to L, but still falls far short of the maximum possible system utility in both experiments. Agents learning using G are encouraged to act in the system's interest; however it is clear that G alone is not an informative enough learning signal to allow the agents to converge on joint policies with a high utility in the BPD.

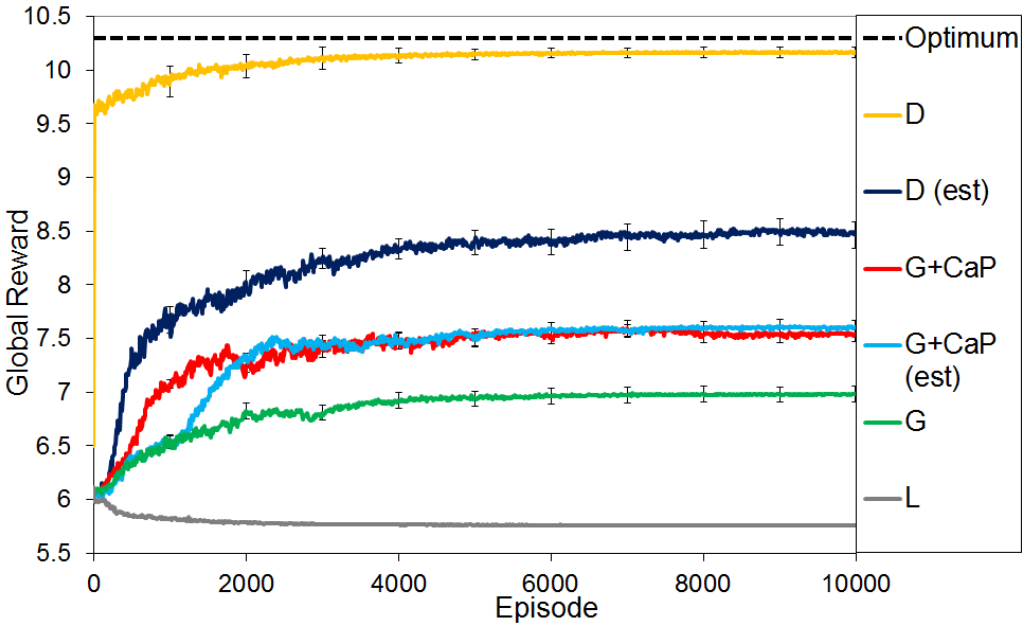
D is considered to be the state of the art for solving congestion problems, and demonstrates the quickest learning rate, reaching close to the maximum possible system utility in both the single- and multi-step problems. $\hat{D}$ does not perform as well as D in either BPD instance, but its performance demonstrates that reaching a high system utility is possible when shaping G with a sufficiently accurate approximation of the counterfactual term.

In both experiments, the final performance of $G + \hat{C}aP$ closely matches that of $G + CaP$. This is impressive, considering that $G + \hat{C}aP$ only requires the value of $G(s, a)$ to be broadcast to each agent, whereas for $G + CaP$ each agent requires $G(s, a)$ and the value of its individual counterfactual term to be broadcast to it. In the single-step BPD, the performance of $G + CaP$ over the last 1000 episodes is statistically better than that of $G + \hat{C}aP$ ($p = 0.01$). There was no statistical difference in the final performance of both approaches in the multi-step BPD ($p = 0.41$). $G + \hat{C}aP$ also offered a statistically significant improvement over unshaped G in both the single-step ($p = 3.71 \times 10^{-6}$) and multi-step ($p = 5.99 \times 10^{-11}$) experiments.

As a single default state and default action were chosen for each agent in this implementation, each agent using $\hat{C}aP$ needed to keep track of just a single $G(s, a)$ approximation to generate its own potential function. This approach is computationally cheap to implement, and signific-



(a) $num\_timesteps = 1$



(b) $num\_timesteps = 5$

Figure 3.4: Beach Problem Domain results

Table 3.3: Beach Problem Domain results (averaged over final 1000 episodes)

	Global Reward	
	$num\_timesteps = 1$	$num\_timesteps = 5$
Optimum	10.30	10.30
D	10.19	10.16
$\hat{D}$	8.54	8.49
$G + CaP$	7.98	7.54
$G + \hat{C}aP$	7.75	7.60
G	7.29	6.98
L	5.75	5.76

antly improves performance over G alone without broadcasting any additional information to the agents. During testing, $\hat{D}$ appeared to be more sensitive to the chosen s_i^c, a_i^c, β and initial $\hat{G}_i(s_i, a_i)$ values than $G + \hat{C}aP$. In some cases $\hat{D}$ converged to lower levels of performance than unshaped G if these parameters were not chosen carefully, whereas $G + \hat{C}aP$ maintained a more consistent level of performance regardless of the parameter values used. It appears that $\hat{C}aP$ may be more robust than $\hat{D}$ to changes in the learning parameters used, however no claims can be made in this regard without further empirical investigation.

3.2.4 Discussion

In this section, I proposed a method for generating multi-agent potential functions based on approximated counterfactuals. Estimated Counterfactual as Potential offers a unique way to shape the behaviour of agents using approximations of the same knowledge represented by difference evaluations, without requiring the mathematical form of the system evaluation function to be known, while also leveraging the theoretical guarantees of $PBRS$. Empirical results demonstrated that agents shaped using $\hat{C}aP$ can reach similar levels of performance to agents shaped using the previously proposed Counterfactual as Potential method, while requiring much less information to be broadcast to the agents. In future, this method could be developed further using more complex $\hat{G}(s, a)$ estimation methods than the tabular approximation used in this study (e.g. using Neural Networks).

3.3 State-Based vs. Action-Based Potential Functions

The question of which form of $PBRS$ is most appropriate to use with a given heuristic has not been addressed in the literature to date. This study will investigate the effect of encoding the same heuristic knowledge using either state-based or action-based potential functions. I expect that this will lead to new insights about the design of potential functions, which will inform future work. This study also constitutes the largest scale evaluation of $aPBRS$ in a MAS to date. The previous largest study by Devlin et al. (2011b) considered up to 9 agents, 4 of which

implemented $aPBRS$; this study will consider a MAS where 100 agents implement $aPBRS$. Therefore, I also hope to demonstrate for the first time that the beneficial effects of $aPBRS$ with a well designed heuristic can scale up to large Multi-Agent Systems.

3.3.1 Shepherd Problem Domain

This section introduces a new multi-agent congestion game, the Shepherd Problem Domain (SPD), which further extends the EBP and BPD by adding additional states and actions. 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 $b \in B$ 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:

$$L(b,t) = x_t e^{\frac{-x_{b,t}}{\psi_b}} \tag{3.10}$$

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

$$G(t) = \sum_{b \in B} L(b,t) \tag{3.11}$$

where B is the set of pastures in the SPD.

The difference reward (D) for an agent may be calculated by applying Eqn. 2.21. As an agent only influences the utility of the pasture it is currently attending at a particular timestep, D_i may be calculated as:

$$D_i(t) = L(b,t) - (x_{b,t} - 1)e^{\frac{-(x_{b,t}-1)}{\psi_s}} \tag{3.12}$$

where $x_{b,t}$ is the number of agents attending the same pasture b as agent i at timestep t. This evaluation is precisely equivalent to applying Eqn. 2.21 to Eqn. 3.11 directly. In a similar manner, the potential function for CaP may be calculated as shown in Eqn. 3.13. Algorithm 2 provides additional details of the process followed in the SPD when agents are rewarded using $G + CaP$.

Algorithm 2 Shepherd Problem Domain with $G + CaP$

initialize Q-values: $\forall b, a | Q(b, a) = -1$
initialize potentials: $\forall b | \Phi(b) = 0$
for $episode = 1 \rightarrow num\_episodes$ **do**
    for $timestep = 1 \rightarrow num\_timesteps$ **do**
        for $i = 1 \rightarrow num\_agents$ **do**
            sense current pasture b
            choose action a, using ϵ-greedy
            move agent to b'
        end for
        for all pastures $b \in B$ **do**
            evaluate local reward $L(b)$ (Eqn. 3.10)
        end for
        evaluate global reward G (Eqn. 3.11)
        for $i = 1 \rightarrow num\_agents$ **do**
            set $\Phi(b')$ (Eqn. 3.13)
            set F (Eqn. 2.12)
            set $R' = G + F$ (Eqn. 2.10)
            update $Q(b, a)$ values (Eqn. 2.2)
        end for
        reduce ϵ by multiplication with $epsilon\_decay\_rate$
        reduce α by multiplication with $alpha\_decay\_rate$
    end for
    for $i = 1 \rightarrow num\_agents$ **do**
        choose action a, using ϵ-greedy
        move to absorbing state
        set $F = 0 - \Phi(b')$ (Eqn. 2.12)
        set $R' = 0 + F$ (Eqn. 2.10)
        update $Q(b', a)$ values (Eqn. 2.2)
    end for
end for

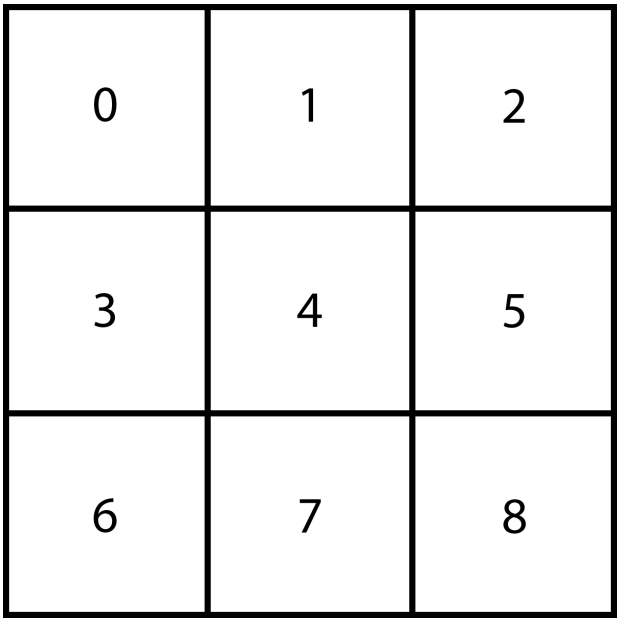


Figure 3.5: SPD 3×3 grid topology with resource (state) numbers

$$\Phi(s) = (x_{b,t} - 1)e^{\frac{-(x_{b,t}-1)}{\psi_b}} \tag{3.13}$$

Other examples of congestion problems that have been studied thus far include the El-Farol Bar Problem (Arthur 1994), the Traffic Lane Domain (TLD) (Tumer et al. 2009) and the Beach Problem Domain (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. This domain features 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 (up to 5 actions in this study), 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 allows for a more thorough evaluation of MARL approaches to tackling resource management problems.

This study features a group of $|B| = 9$ resources in a 3×3 grid arrangement. The 3×3 grid topology used in these experiments is shown in Fig. 3.5. 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. The number of agents is set to $N = 100$, all resources are assumed to have the same capacity $\psi = 4$, and $N/4$ 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.

3.3.2 PBRS Heuristics

To explore the effectiveness of $PBRS$ in this domain, four different manual heuristics will be applied in addition to the automated CaP heuristic. These four manual heuristics are based on those proposed by Devlin et al. (2014) for the BPD, which have been 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 modified versions of these heuristics have been included 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 this study is the first empirical evaluation of $aPBRS$ in congestion games. In the $aPBRS$ heuristics below, the function $next\_state(b, a)$ returns the pasture (local state) that will be reached if an agent selects action a while in state b. The four heuristics that were 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 $|B| = 9$ 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 ($b = 4$), and is therefore an extremely strong heuristic. It is important to note that this heuristic is manually tailored to individual agents. This heuristic is expected to achieve near-optimal performance, demonstrating the effectiveness of $PBRS$ when precise information is available about the optimal joint policy.

$$\Phi(b) = \begin{cases} 10 & \text{if } b = 0 \text{ and } agent\_id \in [0, \psi - 1] \\ 10 & \text{if } b = 1 \text{ and } agent\_id \in [\psi, 2\psi - 1] \\ 10 & \text{if } b = 2 \text{ and } agent\_id \in [2\psi, 3\psi - 1] \\ 10 & \text{if } b = 4 \text{ and } agent\_id \in [3\psi, N/2 - \psi - 1] \\ 10 & \text{if } b = 3 \text{ and } agent\_id \in [N/2 - \psi, N/2 - 1] \\ 10 & \text{if } b = 5 \text{ and } agent\_id \in [N/2, N/2 + \psi - 1] \\ 10 & \text{if } b = 4 \text{ and } agent\_id \in [N/2 + \psi, N - 3\psi - 1] \\ 10 & \text{if } b = 6 \text{ and } agent\_id \in [N - 3\psi, N - 2\psi - 1] \\ 10 & \text{if } b = 7 \text{ and } agent\_id \in [N - 2\psi, N - \psi - 1] \\ 10 & \text{if } b = 8 \text{ and } agent\_id \in [N - \psi, N - 1] \\ 0 & \text{otherwise} \end{cases} \tag{3.14}$$

$$\Phi(b, a) = \begin{cases} 10 & \text{if } next\_state(b, a) = 0 \text{ and } agent\_id \in [0, \psi - 1] \\ 10 & \text{if } next\_state(b, a) = 1 \text{ and } agent\_id \in [\psi, 2\psi - 1] \\ 10 & \text{if } next\_state(b, a) = 2 \text{ and } agent\_id \in [2\psi, 3\psi - 1] \\ 10 & \text{if } next\_state(b, a) = 4 \text{ and } agent\_id \in [3\psi, N/2 - \psi - 1] \\ 10 & \text{if } next\_state(b, a) = 3 \text{ and } agent\_id \in [N/2 - \psi, N/2 - 1] \\ 10 & \text{if } next\_state(b, a) = 5 \text{ and } agent\_id \in [N/2, N/2 + \psi - 1] \\ 10 & \text{if } next\_state(b, a) = 4 \text{ and } agent\_id \in [N/2 + \psi, N - 3\psi - 1] \\ 10 & \text{if } next\_state(b, a) = 6 \text{ and } agent\_id \in [N - 3\psi, N - 2\psi - 1] \\ 10 & \text{if } next\_state(b, a) = 7 \text{ and } agent\_id \in [N - 2\psi, N - \psi - 1] \\ 10 & \text{if } next\_state(b, a) = 8 \text{ and } agent\_id \in [N - \psi, N - 1] \\ 0 & \text{otherwise} \end{cases} \tag{3.15}$$

- **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 ($b = 4$). This shaping is expected to 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.

$$\Phi(b) = \begin{cases} 10 & \text{if } b = 4 \\ 0 & \text{otherwise} \end{cases} \tag{3.16}$$

$$\Phi(b, a) = \begin{cases} 10 & \text{if } next\_state(b, a) = 4 \\ 0 & \text{otherwise} \end{cases} \tag{3.17}$$

- **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, agents receiving this shaping are expected to show modest if any improvements in learning speed and final performance.

$$\Phi(b) = \begin{cases} 10 & \text{if } b = 0 \text{ and } agent\_id \in [0, N/|B| - 1] \\ 10 & \text{if } b = 1 \text{ and } agent\_id \in [N/|B|, 2N/|B| - 1] \\ 10 & \text{if } b = 2 \text{ and } agent\_id \in [2N/|B|, 3N/|B| - 1] \\ 10 & \text{if } b = 3 \text{ and } agent\_id \in [3N/|B|, 4N/|B| - 1] \\ 10 & \text{if } b = 4 \text{ and } agent\_id \in [4N/|B|, 5N/|B| - 1] \\ 10 & \text{if } b = 5 \text{ and } agent\_id \in [5N/|B|, 6N/|B| - 1] \\ 10 & \text{if } b = 6 \text{ and } agent\_id \in [6N/|B|, 7N/|B| - 1] \\ 10 & \text{if } b = 7 \text{ and } agent\_id \in [7N/|B|, 8N/|B| - 1] \\ 10 & \text{if } b = 8 \text{ and } agent\_id \in [8N/|B|, N - 1] \\ 0 & \text{otherwise} \end{cases} \tag{3.18}$$

$$\Phi(b, a) = \begin{cases} 10 & \text{if } next\_state(b, a) = 0 \text{ and } agent\_id \in [0, N/|B| - 1] \\ 10 & \text{if } next\_state(b, a) = 1 \text{ and } agent\_id \in [N/|B|, 2N/|B| - 1] \\ 10 & \text{if } next\_state(b, a) = 2 \text{ and } agent\_id \in [2N/|B|, 3N/|B| - 1] \\ 10 & \text{if } next\_state(b, a) = 3 \text{ and } agent\_id \in [3N/|B|, 4N/|B| - 1] \\ 10 & \text{if } next\_state(b, a) = 4 \text{ and } agent\_id \in [4N/|B|, 5N/|B| - 1] \\ 10 & \text{if } next\_state(b, a) = 5 \text{ and } agent\_id \in [5N/|B|, 6N/|B| - 1] \\ 10 & \text{if } next\_state(b, a) = 6 \text{ and } agent\_id \in [6N/|B|, 7N/|B| - 1] \\ 10 & \text{if } next\_state(b, a) = 7 \text{ and } agent\_id \in [7N/|B|, 8N/|B| - 1] \\ 10 & \text{if } next\_state(b, a) = 8 \text{ and } agent\_id \in [8N/|B|, N - 1] \\ 0 & \text{otherwise} \end{cases} \tag{3.19}$$

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

$$\Phi(b) = \begin{cases} 10 & \text{if } \psi < x_b < 2\psi \\ 0 & \text{otherwise} \end{cases} \tag{3.20}$$

$$\Phi(b,a) = \begin{cases} 10 & \text{if } \psi < x_{next\_state(b,a)} < 2\psi \\ 0 & \text{otherwise} \end{cases} \tag{3.21}$$

3.3.3 Experimental Procedure

For all experiments, the value of the global reward G (referred to as the capacity utility) is plotted 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, the error is calculated 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.

3.3.4 Experimental Results

Figs. 3.6 to 3.9 plot the learning curves for capacity utility for the approaches tested, while Table 3.4 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. Table 3.4 also includes 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. 3.6 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. 3.7 plots learning curves for the 10 reward functions with the best final performance. $G+aPBRS(OvercrowdOne)$ is 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 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.

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 this limitation may be addressed using Estimated Counterfactual as Potential, as was demonstrated in the previous empirical study (Section 3.2).

3.3.5 Discussion

This study explored the issue of appropriate credit assignment in a stochastic resource management game, with a specific focus on the effect of encoding the same heuristic information in $sPBRS$ or $aPBRS$ formats. An interesting finding is that $sPBRS$ and $aPBRS$ variants using the same domain knowledge can encourage very different behaviours in a MAS. This is the first empirical study 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 Overcrowd One heuristic for the SPD), the empirical 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, 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 value function estimate $Q(s, a)$ (Eqn. 2.14). 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

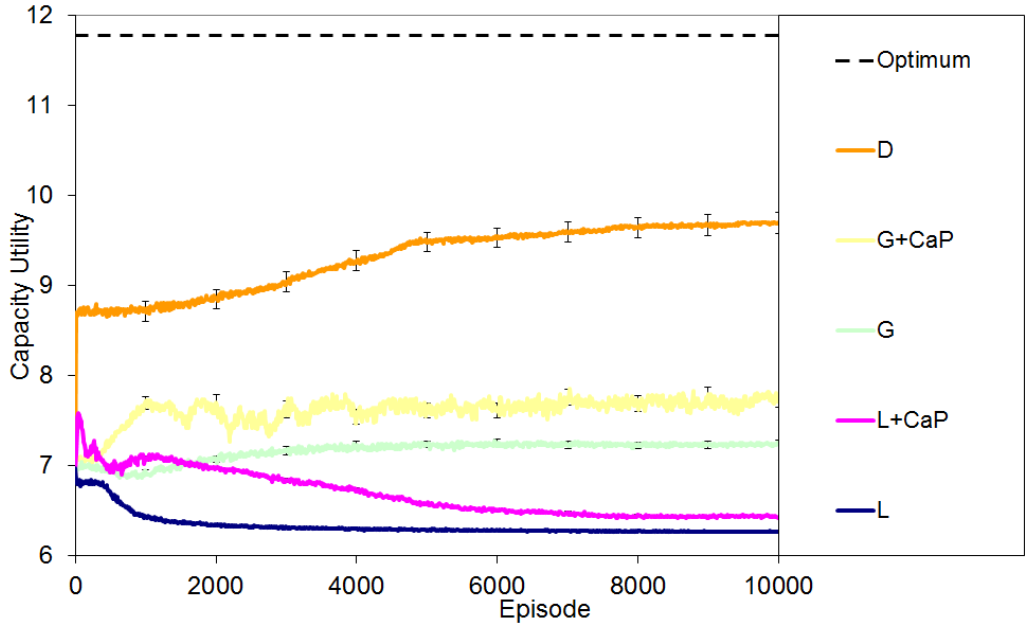


Figure 3.6: Basic reward functions

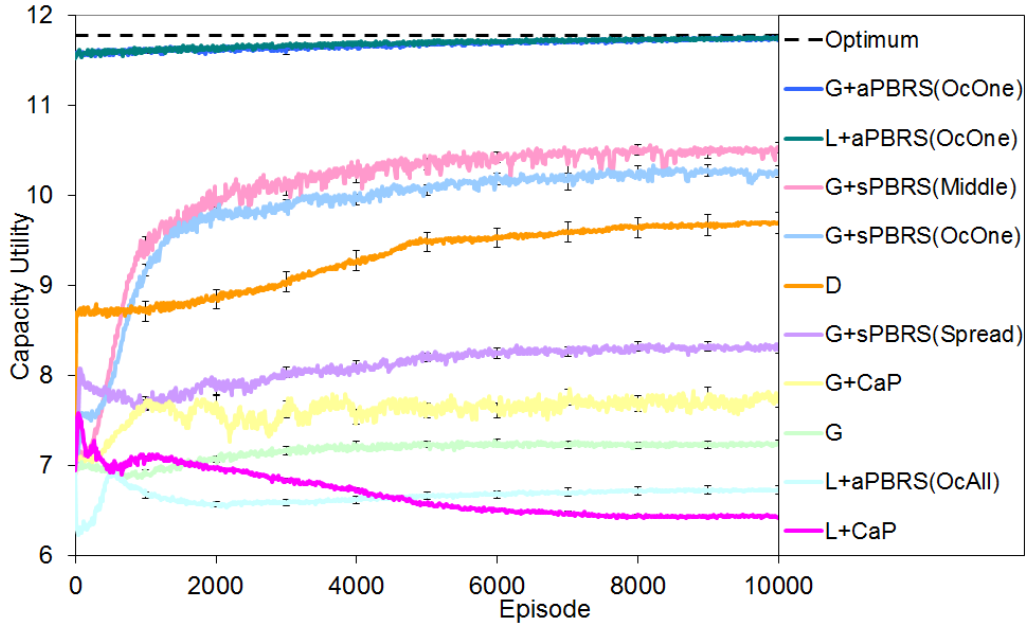


Figure 3.7: Best performing reward functions

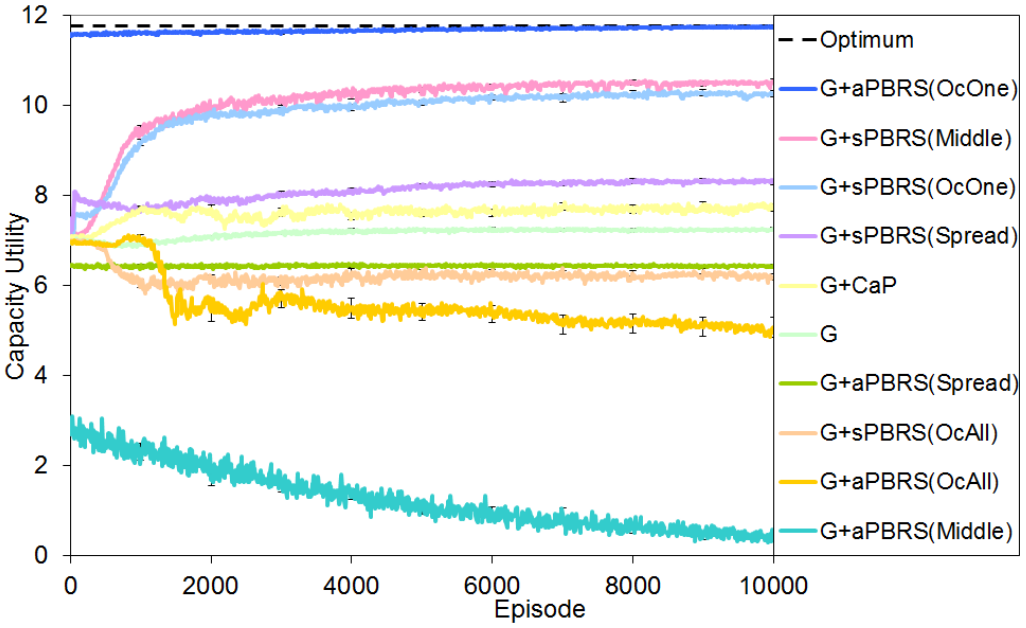


Figure 3.8: G with various heuristics

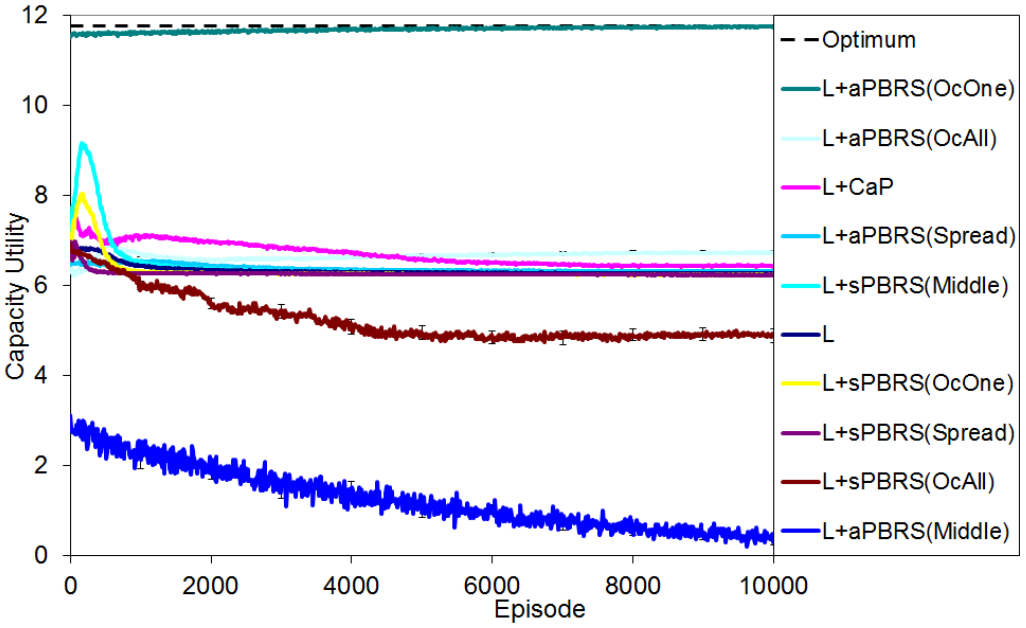


Figure 3.9: L with various heuristics

Table 3.4: Shepherd Problem Domain results (averaged over final 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}

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, these 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, this study also 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 empirical study. Other more complex resource management problems could be analysed using the MAS paradigm, e.g. management of fish stocks, water and other natural resources. In future work, the SPD could be extended 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. I expect that reward shaping will prove to be a useful mechanism to discover good trade-offs between risk and reward in a multi-objective version of the SPD.

It would also be worthwhile to investigate other larger and more complex arrangements of resources in the SPD than the 3×3 grid layout evaluated in this study. 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 & 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. I expect that future studies will demonstrate that scaling can have similar effects in multi-agent domains.

While several multi-agent credit assignment structures were evaluated in this study, there are other recently proposed techniques that may also prove useful for solving resource management problems, e.g. Difference Rewards incorporating Potential-Based Reward Shaping ($DRiP$) (Devlin et al. 2014) and Resource Abstraction (Malialis et al. 2016). I 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; there-

fore I expect that recent work on estimating counterfactuals (Colby et al. 2016) will also have an important role to play, as in these cases it will not be possible to calculate a traditional difference evaluation directly.

3.4 Conclusion

In conclusion, this chapter has explored the question of how to design useful potential functions from a number of different perspectives across three separate empirical studies. The first study demonstrated how a very simple heuristic can be used to improve performance in a realistic Traffic Signal Control application via $aPBRS$. Building upon previous work, the second empirical study evaluated Estimated Counterfactual as Potential, a semi-automated method of generating multi-agent potential functions that aims to match the performance of the analytically computed CaP while requiring much less information. Finally, the third empirical study demonstrated that encoding the same heuristic knowledge in $sPBRS$ and $aPBRS$ formats can have vastly different effects on the behaviour of agents in a MAS, and that the benefical effects of $aPBRS$ with a suitable heuristic do in fact scale up to large MAS. These studies add to the existing body of work that justifies applying principled reward shaping methods to improve Reinforcement Learning, and I hope that they will also inform future research. As I alluded to in Section 3.3.5, reward shaping is also a promising candidate for improving performance in MORL domains. The next chapter will address the question of whether existing reward shaping methods can safely be applied to MORL problems, without the risk of altering the intended goals of those domains.

CHAPTER 4

Reward Shaping in Multi-Objective Reinforcement Learning: Theoretical Considerations

This chapter will address the theoretical implications of applying reward shaping techniques in single- and multi-agent MORL domains. As discussed in Section 2.5.7, reward shaping can alter the optimal policy in MDPs (or the set of Nash equilibria in SGs) when applied carelessly. In multi-objective domains, there exist multiple Pareto optimal solutions, rather than a single preferred solution. Therefore, rather than just preserving a single optimal solution (or a set of Nash equilibria), principled reward shaping methods for MORL should instead preserve the entire set of Pareto optimal solutions of a domain (as well as Nash equilibria in the case of a MOSG).

Section 4.1 discusses the implications of applying $PBRS$ in MORL domains, and theoretical analysis concludes that the Pareto relation between policies is preserved when a potential-based shaping is added to the environment reward function.

Section 4.2 presents a proof which demonstrates that the relative ordering of expected returns for actions in a MOSG is preserved when agents are rewarded with a difference evaluation instead of the system evaluation function. Therefore, any property that depends on the relative ordering of expected returns (including Nash equilibria and the Pareto relation between actions) is invariant when D is used in place of G.

Finally, Section 4.3 concludes with a summary of the contributions of this chapter, and a discussion of their implications for future research.

4.1 Multi-Objective Potential-Based Reward Shaping

This section will extend the existing theoretical guarantees of $PBRS$ with proof that the set of Pareto optimal policies is invariant when $PBRS$ is applied to both infinite- and finite-horizon multi-objective domains. Two ways in which $PBRS$ could be applied to MORL problems are considered: (a) each objective could be shaped separately, or (b) a scalar combination of the objectives could be shaped. Sections 4.1.1 and 4.1.2 will formally analyse case (a), while case (b) will be discused in Section 4.1.3.

4.1.1 Proof for Shaping Each Objective Separately (Infinite-Horizon)

Theorem 1. *Potential-Based Reward Shaping does not alter the Pareto optimal set of policies in an infinite-horizon MOMDP or MOSG.*

Proof. If each objective is shaped independently, the general form of shaping is:

$$\mathbf{R}'_c = \mathbf{R}_c + \mathbf{F}_c \tag{4.1}$$

where $\mathbf{R}_c$ is the reward function component for objective c, $\mathbf{F}_c$ is the shaping applied to objective c, and $\mathbf{R}'_c$ is the shaped reward function component for objective c. If the reward function is modified, it follows that the expected return from the value function is also modified, by adding a term which accounts for all of the expected shaping rewards received during an infinitely long learning trial:

$$\mathbf{V'}_c = \mathbf{V}_c + E\left\{\sum_{t=0}^{\infty}\gamma^t \mathbf{F}_c(s_t, s_{t+1})\right\} \tag{4.2}$$

where $\mathbf{V}_c$ is the value function component for objective c, and $\mathbf{V}'_c$ is the shaped value function component for objective c. Assuming that an agent begins in state s_0 and follows a given policy π, and that the shaping rewards are of the form in Eqn. 2.12, the value of objective c when receiving $sPBRS$ is:

$$\begin{aligned}
\mathbf{V'}^{\pi}_c(s_0) &= \mathbf{V}^{\pi}_c(s_0) + E^{\pi}\left\{\sum_{t=0}^{\infty}\gamma^t \mathbf{F}_c(s_t, s_{t+1})\right\} \\
\mathbf{V'}^{\pi}_c(s_0) &= \mathbf{V}^{\pi}_c(s_0) + E^{\pi}\left\{\sum_{t=0}^{\infty}\gamma^t (\gamma\Phi_c(s_{t+1}) - \Phi_c(s_t))\right\} \\
\mathbf{V'}^{\pi}_c(s_0) &= \mathbf{V}^{\pi}_c(s_0) + E^{\pi}\left\{\sum_{t=0}^{\infty}\gamma^{t+1}\Phi_c(s_{t+1}) - \sum_{t=0}^{\infty}\gamma^t\Phi_c(s_t)\right\} \\
\mathbf{V'}^{\pi}_c(s_0) &= \mathbf{V}^{\pi}_c(s_0) + E^{\pi}\left\{\sum_{t=1}^{\infty}\gamma^t\Phi_c(s_t) - \sum_{t=0}^{\infty}\gamma^t\Phi_c(s_t)\right\} \\
\mathbf{V'}^{\pi}_c(s_0) &= \mathbf{V}^{\pi}_c(s_0) - \Phi_c(s_0)
\end{aligned} \tag{4.3}$$

All π are evaluated from the same starting state s_0, and $\mathbf{V}_c^{\pi}(s_0)$ is modified by the same amount $-\Phi_c(s_0)$ for all $\pi \in \Pi$, therefore the Pareto relation between all $\pi \in \Pi$ is invariant, and the NDS remains consistent:

$$\forall_{c \in C}\{[\mathbf{V'}_c^{\pi}(s_0) = \mathbf{V'}_c^{\pi^*}(s_0)] \iff [\mathbf{V}_c^{\pi}(s_0) = \mathbf{V}_c^{\pi^*}(s_0)]\} \tag{4.4}$$

And:

$$[\mathbf{V'}_c^{\pi}(s_0) < \mathbf{V'}_c^{\pi^*}(s_0)] \iff [\mathbf{V}_c^{\pi}(s_0) < \mathbf{V}_c^{\pi^*}(s_0)] \tag{4.5}$$

□

4.1.2 Proof for Shaping Each Objective Separately (Finite-Horizon)

Theorem 2. *Potential-Based Reward Shaping does not alter the Pareto optimal set of policies in a finite-horizon MOMDP or MOSG.*

Proof. In finite-horizon domains, each learning episode has a certain number of timesteps, referred to as the horizon H. The modified value function component for objective c in this case is:

$$\mathbf{V'}_c = \mathbf{V}_c + E\left\{\sum_{t=0}^{H-1} \gamma^t \mathbf{F}_c(s_t, s_{t+1})\right\} \tag{4.6}$$

Assuming again that an agent starts in state s_0 and follows a given policy π, and that the shaping rewards are of the form in Eqn. 2.12, the value of objective c when receiving $sPBRS$ in a finite-horizon domain is:

$$\begin{aligned}
\mathbf{V'}_c^{\pi}(s_0) &= \mathbf{V}_c^{\pi}(s_0) + E^{\pi}\left\{\sum_{t=0}^{H-1} \gamma^t \mathbf{F_c}(s_t, s_{t+1})\right\} \\
\mathbf{V'}_c^{\pi}(s_0) &= \mathbf{V}_c^{\pi}(s_0) + E^{\pi}\left\{\sum_{t=0}^{H-1} \gamma^t (\gamma\Phi_c(s_{t+1}) - \Phi_c(s_t))\right\} \\
\mathbf{V'}_c^{\pi}(s_0) &= \mathbf{V}_c^{\pi}(s_0) + E^{\pi}\left\{\sum_{t=0}^{H-1} \gamma^{t+1}\Phi_c(s_{t+1}) - \sum_{t=0}^{H-1} \gamma^t \Phi_c(s_t)\right\} \\
\mathbf{V'}_c^{\pi}(s_0) &= \mathbf{V}_c^{\pi}(s_0) + E^{\pi}\left\{\sum_{t=1}^{H} \gamma^t \Phi_c(s_t) - \sum_{t=0}^{H-1} \gamma^t \Phi_c(s_t)\right\} \\
\mathbf{V'}_c^{\pi}(s_0) &= \mathbf{V}_c^{\pi}(s_0) + E^{\pi}\left\{\gamma^H \Phi_c(s_H)\right\} - \Phi_c(s_0) \\
\mathbf{V'}_c^{\pi}(s_0) &= \mathbf{V}_c^{\pi}(s_0) + \gamma^H \sum_{s \in S}\left\{Pr^{\pi}(s_H = s)\Phi_c(s)\right\} - \Phi_c(s_0)
\end{aligned} \tag{4.7}$$

This result is similar to that for the infinite-horizon case, apart from the additional term $\gamma^H \sum_{s \in S}\left\{Pr^{\pi}(s_H = s)\Phi_c(s)\right\}$, which represents the expected shaping reward for the final

state s_H. The probability of being in a certain state s at time H depends on the policy π being evaluated, which in turn determines what shaping reward will be received. This additional term could potentially cause changes in the Pareto relation between policies.

If $\gamma < 1.0$ and the value of H is sufficiently large, the value of the additional term would become insignificant. However, this term can be eliminated completely by ensuring that the final potential equals zero, i.e. $\Phi_c(s_H) = 0$. This is the same condition identified by both Devlin (2013) and Grześ (2017), and discussed in Section 2.6.4 of this thesis.

This requirement can easily be taken into account when designing potential functions for episodic problems where the terminal states are known, by assigning a zero potential to all terminal states. Alternatively, for applications where the terminal states are not known beforehand, the potential of the last state experienced can dynamically be set to zero.

A more robust method to preserve the theoretical guarantees of $PBRS$ in finite-horizon problems is to implement an additional absorbing state s_{abs} with zero potential (i.e. $\Phi(s_{abs}) = 0$). Upon reaching the terminal state, all agents select actions as normal, and are transitioned to the absorbing state. No reward is received from the environment for this transition, but agents do receive the shaping reward as calculated by Eqn. 2.12. As all possible policies now terminate in the absorbing state, $Pr^{\pi}(s_H = s_{abs}) = 1.0$, meaning that the additional term $\gamma^H \sum_{s \in S} \left\{ Pr^{\pi}(s_H = s) \Phi_c(s) \right\}$ is guaranteed to sum to zero.

Assuming that the final potential is set to zero using any of these methods, the form of $\mathbf{V'}_c^{\pi}(s_0)$ is exactly the same as in the infinite-horizon case, and therefore the NDS also remains consistent when $PBRS$ is applied to finite-horizon MORL domains. □

4.1.3 Discussion

Following the same methods as the proofs above, it can be demonstrated that the guarantees of Pareto front invariance also hold when $PBRS$ is used to shape a scalarised combination of objectives, if the objective specific terms $\mathbf{V'}_c^{\pi}(s_0)$ and $\mathbf{V}_c^{\pi}(s_0)$ are replaced with scalarised versions, and $\mathbf{F_c}$ is replaced by a single shaping F.

It may also be shown that these guarantees hold if an action-based potential function is used instead, provided that an agent implementing $aPBRS$ in an MORL domain uses a biased greedy action selection mechanism (Eqn. 2.14), as per the original $aPBRS$ proof provided by Wiewiora et al. (2003).

$PBRS$ has not been applied to any MORL domains thus far, and therefore questions remain as to how best to apply it. Previous work by Brys et al. (2014) used $PBRS$ for multi-objectivisation, i.e. adding additional psuedo objectives to a single-objective problem in order to speed up learning. However, this work is not an evaluation of $PBRS$ in a true multi-objective context, as the objectives considered are fully correlated and are therefore not in conflict. Given this deficit in the literature, the effect of $PBRS$ in MORL domains merits substantial further study.

One remaining question is whether $PBRS$ should be applied by (a) shaping each objective separately or (b) shaping a scalarised combination. Further investigation is required into this topic, although the efficacy of one option versus another is likely to be domain specific, and will also depend on the nature of the heuristic information that is available.

Option (a) may be more useful in domains where there is a low degree of correlation between objectives, or where domain knowledge is available that could improve performance on one specific objective. On the other hand, shaping scalarised combinations may be preferable in domains with strongly correlated objectives, or when applying $PBRS$ to complex domains where designing potential functions to shape each objective individually is impractical or unintuitive.

In Chapter 5, both options (a) and (b) are applied in a single-agent domain (Section 5.1), and option (b) is applied in two multi-agent domains (Sections 5.2 and 5.3). These studies will examine the effects of $PBRS$ in MORL domains, and provide supporting evidence for the proofs in this section. While scalarised Q-learning is used exclusively in the empirical demonstrations in Chapter 5, note that option (a) is also suitable for use in combination with MORL algorithms that do not use scalarisation, as each component of the reward vector is shaped independently.

Following from these proofs, and the proof of policy invariance by Ng et al. (1999), I expect to demonstrate that agents learning with and without $PBRS$ in single-agent MORL domains will learn the exact same set of Pareto optimal policies. In the multi-agent case, I expect that agents learning with and without $PBRS$ may converge to different Pareto optimal joint policies. This follows from the proof of consistent Nash Equilibria in SGs by Devlin & Kudenko (2011).

While applying $PBRS$ does not alter the true Pareto front of a MOSG, it may alter the Nash equilibrium reached by the agents, and therefore different policies could be learned compared to agents learning without $PBRS$. However, the set of possible policies that could be learned and their Pareto relation to one another remains consistent when $PBRS$ is applied. Which specific point(s) of equilibrium will be reached is a feature of the learning algorithm(s) implemented by the agents; recall from Section 2.3 that multiple individual Q-learners do not provide any theoretical guarantees in this respect.

4.2 Difference Rewards Theory

Recent work by Colby & Tumer (2015) considered the effect of applying D in a two-player single-objective matrix game, and showed that the relative ordering of expected returns (and therefore the Nash equilibria) are not altered when agents are rewarded using D instead of G. The analysis in this section will generalise their result to the case of a co-operative Stochastic Game with $|C| \geq 1$ objectives and N agents.

Theorem 3. *For any state $s \in S$ in a co-operative Stochastic Game, any property that depends on the relative ordering of rewards is not altered when difference evaluations are used in place of the system evaluation function.*

Proof. For any system state $s \in S$ in a co-operative Stochastic Game, the agents select some joint action a according to their joint policy π, and are rewarded immediately for the system state transition using the global system evaluation function $\mathbf{G}$. For this analysis it is assumed that system transitions are deterministic, i.e. $\forall s \in S, s' \in S, a \in A | T(s,a,s') = 1$, and that states and actions are represented discretely.

If all agents except agent i follow some joint policy $\pi^{\dagger}_{-i} \in \Pi_{-i}$, and agent i follows some policy $\pi_i \in \Pi_i$, the resulting joint policy is $\pi^{\dagger}_{-i} \cup \pi_i$. Suppose that the reward for a system objective $c \in C$ is greater if agent i follows policy $\pi^1_i \in \Pi_i$ rather than $\pi^2_i \in \Pi_i$ in state s when all other agents follow their respective policies from π_{-i}. Formally:

$$\mathbf{G}_c(s, a^{\dagger}_{-i} \cup a^1_i) > \mathbf{G}_c(s, a^{\dagger}_{-i} \cup a^2_i) \tag{4.8}$$

where $\mathbf{G}_c(s,a)$ is the return from the system evaluation function for objective c when joint action a is selected in system state s, $a^{\dagger}_{-i}$ are the actions selected in state s by all agents except agent i when following their policies from $\pi^{\dagger}_{-i}$, and a^1_i and a^2_i are the actions selected by agent i when following policy π^1_i or π^2_i respectively.

If each objective is to be shaped independently (rather than shaping a scalarised combination) when using difference evaluations, a counterfactual term must be calculated for each objective c in order to apply Eqn. 2.21 to the global reward vector. However, as the counterfactual term $\mathbf{G}_c(s_{-i} \cup s^c_i, a^{\dagger}_{-i} \cup a^c_i)$ for any objective c does not depend on the policy being followed by agent i, for each possible system state s it can be inferred that the counterfactual for objective c for agent i when all other agents follow the joint policy π_{-i} must be a fixed quantity. Therefore, $\mathbf{G}_c(s_{-i} \cup s^c_i, a_{-i} \cup a^c_i)$ may be subtracted from each side of Eqn. 4.8 while preserving the inequality:

$$\begin{aligned} \mathbf{G}_c(s, a^{\dagger}_{-i} \cup a^1_i) - \mathbf{G}_c(s_{-i} \cup s^c_i, a^{\dagger}_{-i} \cup a^c_i) > \\ \mathbf{G}_c(s, a^{\dagger}_{-i} \cup a^2_i) - \mathbf{G}_c(s_{-i} \cup s^c_i, a^{\dagger}_{-i} \cup a^c_i) \end{aligned} \tag{4.9}$$

Therefore, noting that the difference evaluation for objective c for agent i is:

$$\mathbf{D}_{c,i}(s_i, a_i) = \mathbf{G}_c(s,a) - \mathbf{G}_c(s_{-i} \cup s^c_i, a_{-i} \cup a^c_i) \tag{4.10}$$

It can be shown that:

$$\begin{aligned} \forall c \in C, s \in S, i \in \{1, ..., N\} \Big[\mathbf{D}_{c,i}(s_i, a^1_i) > \mathbf{D}_{c,i}(s_i, a^2_i) \\ \iff \mathbf{G}_c(s, a^{\dagger}_{-i} \cup a^1_i) > \mathbf{G}_c(s, a^{\dagger}_{-i} \cup a^2_i) \Big] \end{aligned} \tag{4.11}$$

This means that difference evaluations do not alter the order of rewards for actions in any system state s, although they do alter the absolute values. Any property that relies on the ordering of rewards, and not the absolute value is therefore unaffected for each system state s. For example, if an action a_i in state s leads to a Nash equilibrium reward with respect to $\mathbf{G}$, it also leads to a Nash equilibrium reward with respect to $\mathbf{D}_i$. In the case of a MOSG where $|C| \geq 2$, if an action a_i in state s is Pareto optimal with respect to $\mathbf{G}$, it is also Pareto optimal with respect to $\mathbf{D}_i$. □

4.3 Conclusion

This chapter presented the first reported theoretical analysis of reward shaping in MORL domains. Two popular credit assignment techniques were considered: Potential-Based Reward Shaping and difference rewards. When applying any method of Knowledge-Based Reinforcement Learning, it is important that domain knowledge is provided in a theoretically sound manner, without the risk of altering the original goal(s) of the problem as defined by the environment reward function. Following from the analysis in this chapter, it can be concluded that both D and $PBRS$ possess this desirable property.

Section 4.1 evaluated the effect of $PBRS$ on the expected value of objectives, and demonstrated that the use of a potential-based shaping function does not alter the Pareto relation between an agent's policies in MORL domains. This results holds true for infinite-horizon domains without any additional requirements, and for finite-horizon domains provided that the potential of the last state experienced is equal to zero. These guarantees of Pareto front invariance add to the preexisting guarantees discussed in Section 2.6, and provide theoretical justification for the use of $PBRS$ in MORL applications.

Section 4.2 built upon previous work by Colby & Tumer (2015), generalising their results from the case of a two-player single-objective matrix game, to a MOSG with N players. This analysis demonstrated that the order of expected returns for actions in any system state is invariant when an agent is rewarded using a difference evaluation, rather than the system evaluation function. Therefore, actions which would lead to a Nash equilibrium reward with respect to D will also lead to a Nash equilibrium reward with respect to G, and actions that are Pareto optimal with respect to D are also Pareto optimal with respect to G.

Both $PBRS$ and D preserve the order of expected returns, but alter the absolute values of rewards received. Therefore, actions must be selected using an advantage-based policy to ensure that their respective theoretical guarantees will hold. Wiewiora (2003) defines an advantage-based policy as one that "chooses an action in a given state with a probability that is determined by the differences of the Q-values for that state, not their absolute magnitude".

The most commonly used action selection methods in RL satisfy this condition, as they greedily select actions based on the order of expected returns. Examples of advantage-based action selection methods include greedy, ϵ-greedy and softmax. If agents were to select actions based on some other criteria (e.g. achieving a minimum utility for a certain objective), these guaran-

tees could potentially be broken, which may modify the Pareto optimal solutions and/or Nash equilibria of a domain.

Arguably, the most interesting property of reward shaping is its effect when applied in a cooperative MAS; i.e. its ability to improve the quality of the final joint policies learned. Besides those presented in Sections 3.2 and 3.3 of this thesis, there are many other examples in the literature which demonstrate that the system evaluation function is both too noisy and too uninformative to allow agents to learn good joint policies in single-objective MAS (see e.g. Tumer et al. (2009); Devlin et al. (2014); Malialis et al. (2016)). Reward shaping can address this deficit by providing additional information to agents, thus improving agent coordination and the quality of the final joint policy learned.

As MAS research naturally moves towards more precise models of real world systems which explicitly take multiple objectives into account, the shortcomings inherent in G as a reward structure will only become more pronounced; this issue has already been identified by Yliniemi (2015) and Yliniemi & Tumer (2016) in a MOMARL context. Therefore, reward shaping is a more important technique than ever before, as coordinating agents' actions to achieve a high level of system performance becomes more difficult when multiple criteria must be optimised simultaneously.

Following from the analysis in this chapter, future researchers can now apply reward shaping to a whole new class of problems (MOSGs), in a theoretically sound manner and without fear of distracting agents from their originally intended goals. The next chapter of this thesis will build upon the investigations presented here, demonstrating how the benefits of principled reward shaping methods can be leveraged to improve performance across a range of MORL application domains, both single- and multi-agent.

CHAPTER 5

Reward Shaping in Multi-Objective Reinforcement Learning: Empirical Studies

This chapter presents three different empirical studies on reward shaping in both single- and multi-agent MORL domains. These include the first reported empirical evaluations of reward shaping in MORL problems where the true Pareto optimal solutions are known. Given the theoretical analysis presented in the previous chapter, I expect to demonstrate that agents learning using either D or $sPBRS$ can discover true Pareto optimal solutions in MORL domains, while also benefiting from the increased learning speed that is characteristic of these techniques.

Section 5.1 presents an evaluation of $sPBRS$ in a widely used single-agent MORL benchmark problem, the Deep Sea Treasure domain. Empirical results demonstrate that agents learning with and without $sPBRS$ learn the exact same set of Pareto optimal solutions, and that the quality of the shaping heuristic used has a dramatic impact on learning speed.

In Section 5.2 the first ever MOMARL benchmark with known Pareto optimal solutions is introduced, and is used to evaluate the efficacy of both D and $sPBRS$ at improving agent coordination in MOSGs. Experimental results show that both D and $sPBRS$ can guide agents towards true Pareto optimal solutions in MOSGs.

Finally, Section 5.3 presents an empirical evaluation of D and $sPBRS$ using a realistic electricity generator scheduling problem. Both shaping techniques are found to improve performance in this domain when compared to agents learning using the unshaped system evaluation function, in terms of learning speed and the quality of the non-dominated solutions found.

5.1 Single-Agent Study

The aim of this initial study is to empirically evaluate whether the characteristic effects of $sPBRS$ in MDPs (policy invariance and altered learning rates) still hold in MOMDPs. As stated in Chapter 4, the notion of policy invariance in single-objective domains naturally extends to Pareto front invariance in MORL domains. As there may be multiple non-dominated policies in MORL domains, any reward shaping technique that claims invariance must preserve the true Pareto front of the problem, i.e. the set of policies that is Pareto optimal w.r.t. the original reward function must remain consistent.

An established single-agent MORL benchmark domain was chosen in order to present the simplest possible evaluation of the effects of $PBRS$ in MORL domains. I expect to demonstrate that agents learning with and without $sPBRS$ will learn the exact same set of policies upon convergence, experimentally validating the theoretical analysis presented in Section 4.1. Furthermore, I hypothesise that the effect of $sPBRS$ on learning speed will depend on the quality of the potential function used, as is the case in single-objective domains. Note that D is not included in this study, as it is only applicable to MAS.

5.1.1 Deep Sea Treasure

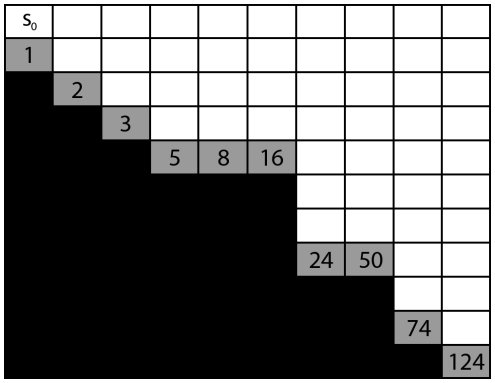
The Deep Sea Treasure (DST) environment was proposed by Vamplew et al. (2010) as a benchmark problem for single-agent MORL algorithms. This is a useful benchmark problem, as the true Pareto front is known. Thus, it can be used to accurately evaluate the effect of $sPBRS$ on the set of Pareto optimal policies that are learned in single-agent MORL domains.

The DST environment, shown in Fig. 5.1a, consists of 10 rows and 11 columns. An agent controls a submarine, which searches for undersea treasures. There are 10 treasure locations in all, and the agent begins each episode in the top left state (labelled s_0 in Fig. 5.1a).

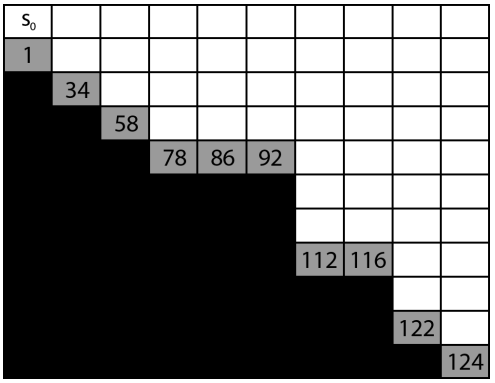
An episode ends after 1000 actions, or when the agent reaches a treasure location. The agent's state is defined as its current position on the grid, and the actions available correspond to moving one square in one of the four cardinal directions. Actions which would cause the agent to leave the grid leave its position unchanged.

There are two objectives in this domain: to minimise the time taken to reach a treasure, and to maximise the reward received when a treasure is reached. After each action selection, the agent receives a reward vector with two elements. The first element is the time reward, which is -1 for all turns. The second element is the treasure reward, which is the value for the corresponding cell in Fig. 5.1a if a treasure is reached, and zero for all other turns.

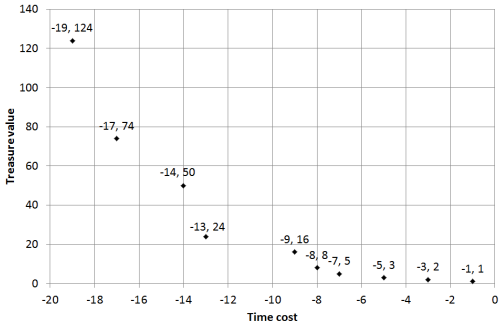
The Pareto front for this problem consists of 10 elements, with a non-dominated policy corresponding to each of the 10 treasure locations. The Pareto front is plotted in Fig. 5.1c and is globally concave, with local concavities at the second, fourth and sixth points from the left (Vamplew et al. 2010).



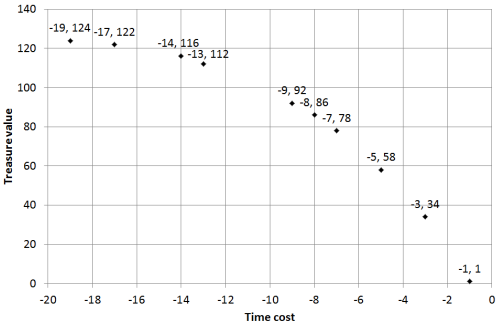
(a) Deep Sea Treasure environment



(b) Convex Deep Sea Treasure environment



(c) Deep Sea Treasure Pareto front



(d) Convex Deep Sea Treasure Pareto front

Figure 5.1: Deep Sea Treasure domains

A modified version of the DST domain called the Convex Deep Sea Treasure (CDST) environment will also be used. In the CDST, the values for the treasure rewards have been altered to create a Pareto front that is globally convex. This modification means that linear scalarised Q-learning can learn all of the Pareto optimal policies in the CDST. The CDST and its Pareto front are shown in Figs. 5.1b and 5.1d respectively.

5.1.2 Experimental Procedure

For scalarised Q-learning, the reward vector received at each timestep is converted into a single scalar value using linear scalarisation (Eqn. 2.7). As was noted earlier in Section 2.4.1, a shortcoming of linear scalarised Q-learning is its inability to learn policies in concave regions of the Pareto front. Thus linear scalarised Q-learning will not learn all Pareto optimal policies in the DST environment, but this domain will nevertheless prove useful when evaluating the effects of $sPBRS$. Linear scalarised Q-learning will learn all policies in the CDST due to the globally convex Pareto front.

This algorithm was run with 100 different weight vectors, where the continuous range $[0, 1]$ is uniformly discretised with step size $\frac{1}{100-1}$, so that a number of different policies would be

Algorithm 3 DST domain with $PBRS(Poor)$

initialise Q-values: $\forall s, a | Q(s,a) = 0 \times w_{time} + 125 \times w_{treasure}$
for $episode = 1 \rightarrow num\_episodes$ **do**
    set initial agent position
    for $timestep = 1 \rightarrow max\_timesteps$ **do**
        sense current position s
        set potential $\Phi(s)$ (Eqn. 5.3)
        choose action a, using ϵ-greedy
        move agent to s'
        set potential $\Phi(s')$ (Eqn. 5.3)
        calculate environment time reward
        calculate environment treasure reward
        set r = scalarised environment reward (Eqn. 2.7)
        set f (Eqn. 2.12)
        set $r' = r + f$ (Eqn. 2.10)
        update $Q(s,a)$ values using r' (Eqn. 2.2)
        if $treasure\_found$ **then**
            break
        end if
    end for
    choose action a, using ϵ-greedy
    move to absorbing state
    set $f = 0 - \Phi(s')$ (Eqn. 2.12)
    set $r' = 0 + f$ (Eqn. 2.10)
    update $Q(s',a)$ values (Eqn. 2.2)
    reduce ϵ using $epsilon\_decay\_rate$
end for

learned. This is a similar method to that used by Vamplew et al. (2010) to learn all Pareto optimal policies in the DST using linear scalarised Q-learning.

At the start of each run, the action values were optimistically initialised to $[0, 125]$ scalarised with the appropriate weight vector for non-terminal states, and to 0 for terminal states. The learning parameters used were as follows: $\alpha = 0.1$, $\gamma = 1$. The action value initialisation method, and values for α and γ are the same as were used by Vamplew et al. (2010) in their empirical study on the DST domain. The exploration rate, ϵ, was set to 0.998^e, where e is the episode number.

To test the effect of $sPBRS$ in this problem domain, two different types of potential functions were used: good and poor. In these potential functions each entry corresponds to a cell in the DST environment. Entries in bold denote treasure locations, and blank entries denote unreachable cells. The process followed in the DST domain when $sPBRS$ is applied is described in detail in Algorithm 3.

The good heuristics for the DST and CDST are shown in Eqns. 5.1 and 5.2 respectively. These heuristics were chosen to illustrate the increased learning speed that is possible when

useful domain knowledge is available, and are used to shape the treasure component of the reward vector that is received. Agents are encouraged to explore along the Pareto front when using the good heuristics, with states close to high valued treasures assigned the highest potentials.

Eqn. 5.3 is an example of a poorly-designed potential function, and is used to shape the scalarised combination of the reward vector that is received in the DST and the CDST. This potential function encourages an agent towards the upper right corner of the domain, far away from any treasure locations. As this heuristic information is misleading, it is expected to reduce the learning speed of any agents that receive it.

$$\Phi_{Good}(s) = \begin{bmatrix} 6 & 12 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ \mathbf{1} & 19 & 25 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ & \mathbf{2} & 31 & 37 & 2 & 2 & 2 & 2 & 2 & 2 \\ & & \mathbf{3} & 43 & 50 & 56 & 62 & 3 & 3 & 3 \\ & & & \mathbf{5} & \mathbf{8} & \mathbf{16} & 78 & 4 & 4 & 4 \\ & & & & & & 74 & 5 & 5 & 5 \\ & & & & & & 81 & 87 & 93 & 6 \\ & & & & & & \mathbf{24} & \mathbf{50} & 99 & 7 \\ & & & & & & & & 105 & 112 \\ & & & & & & & & \mathbf{74} & 118 \\ & & & & & & & & & \mathbf{124} \end{bmatrix} \tag{5.1}$$

$$\Phi_{Good}(s) = \begin{bmatrix} 6 & 12 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ \mathbf{1} & 19 & 25 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ & \mathbf{34} & 31 & 37 & 2 & 2 & 2 & 2 & 2 & 2 \\ & & \mathbf{58} & 43 & 50 & 56 & 62 & 3 & 3 & 3 \\ & & & \mathbf{78} & \mathbf{86} & \mathbf{92} & 78 & 4 & 4 & 4 \\ & & & & & & 74 & 5 & 5 & 5 \\ & & & & & & 81 & 87 & 93 & 6 \\ & & & & & & \mathbf{112} & \mathbf{116} & 99 & 7 \\ & & & & & & & & 105 & 112 \\ & & & & & & & & \mathbf{122} & 118 \\ & & & & & & & & & \mathbf{124} \end{bmatrix} \tag{5.2}$$

$$\Phi_{Poor}(s) = \begin{bmatrix} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 \\ \mathbf{0} & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 \\ & \mathbf{0} & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \\ & & \mathbf{0} & 1 & 2 & 3 & 4 & 5 & 6 & 7 \\ & & & \mathbf{0} & \mathbf{0} & \mathbf{0} & 3 & 4 & 5 & 6 \\ & & & & & & 2 & 3 & 4 & 5 \\ & & & & & & 1 & 2 & 3 & 4 \\ & & & & & & \mathbf{0} & \mathbf{0} & 2 & 3 \\ & & & & & & & & 1 & 2 \\ & & & & & & & & \mathbf{0} & 1 \\ & & & & & & & & & \mathbf{0} \end{bmatrix} \tag{5.3}$$

5.1.3 Experimental Results

All plots include error bars representative of the standard error of the mean based on 30 statistical runs. Specifically, the error is calculated 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 100 episode intervals. The plots show the average performance across the 30 statistical runs that were conducted at 100 episode intervals.

The results from the DST domain are plotted in Figs. 5.2a and 5.2b, which show the online and offline hypervolumes of the non-dominated policies learned by each approach over the training period. The online hypervolume is calculated using the accumulated rewards during learning, while the offline hypervolume is calculated using the accumulated rewards received by greedily evaluating the current policy. In both cases the Pareto-dominated accumulated reward vectors are removed, and the remaining Pareto optimal vectors are then used to calculate the hypervolumes.

All hypervolumes are calculated using a reference point of [-25,0]. The hypervolume of the true Pareto front for the DST domain is 1155 using the reference point specified. A summary of the final hypervolumes and final policies learned in the DST domain is provided in Table 5.1.

As expected, scalarised Q-learning falls far short of the maximum possible hypervolume in the DST, as it only learns two Pareto optimal policies upon convergence, corresponding to the extreme solutions at either end of the Pareto front, $[-1, 1]$ and $[-19, 124]$. This results in a hypervolume of 762, which matches with the experiences of Vamplew et al. (2010) and Van Moffaert & Nowé (2014), who also report that scalarised Q-learning can only learn the policies on the convex portion of the Pareto front in this domain.

Figs. 5.2a and 5.2b show a considerable improvement in learning speed when a good $sPBRS$ heuristic is added to scalarised Q-learning in the DST. Note also that the hypervolume actually increases beyond its final value when $sPBRS$ is added. This shows that the addition of $sPBRS$

Table 5.1: Pareto dominating policies for DST at the end of the training period

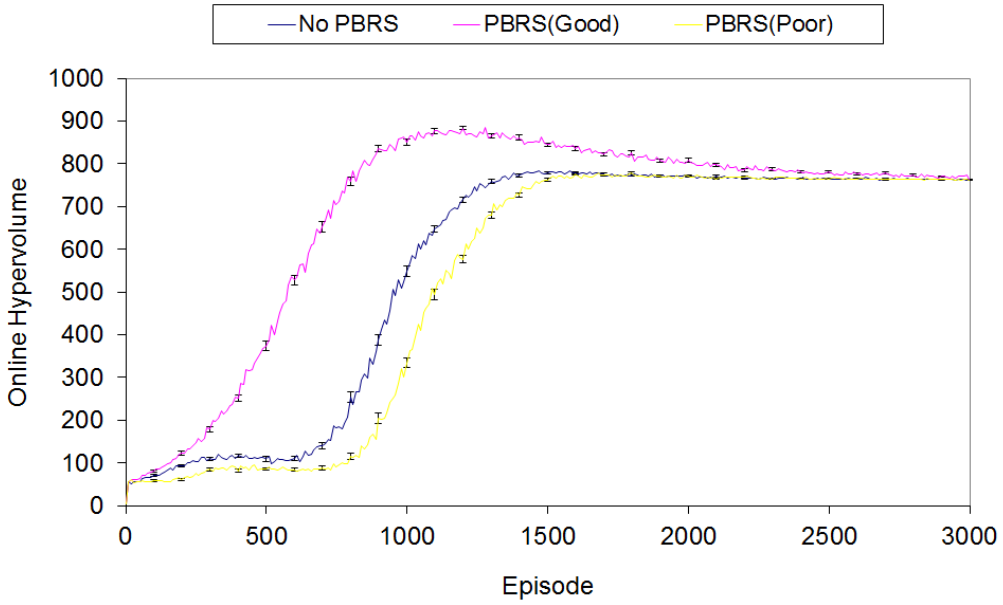
	Hypervolume	Pareto dominating policies
True Pareto front	1155	[-1,1],[-3,2],[-5,3],[-7,5],[-8,8],[-9,16], [-13,24],[-14,50],[-17,74],[-19,124]
No sPBRS	762	[-1,1],[-19,124]
sPBRS (Good)	762	[-1,1],[-19,124]
sPBRS (Poor)	762	[-1,1],[-19,124]

Table 5.2: Pareto dominating policies for CDST at the end of the training period

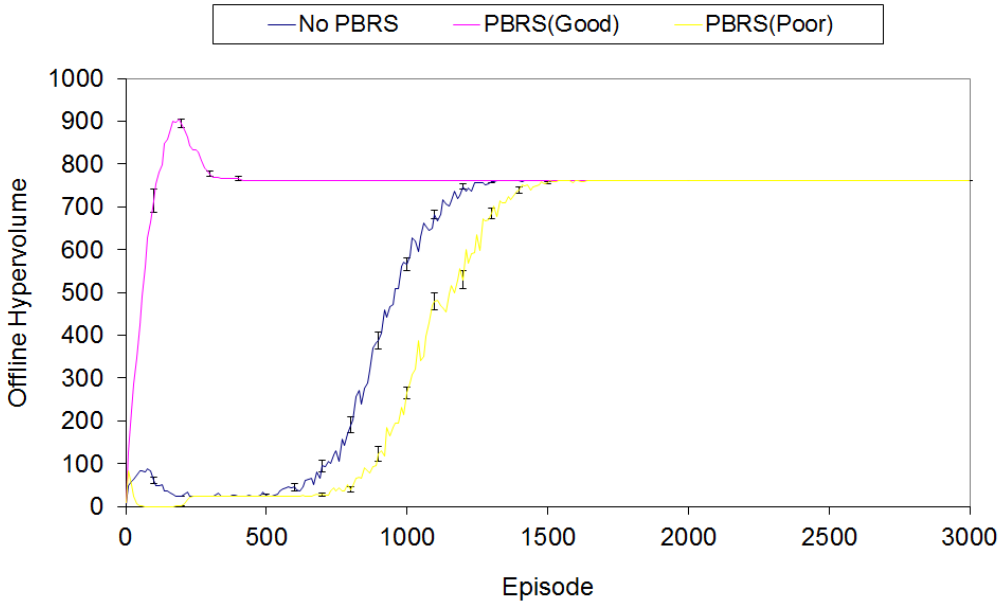
	Hypervolume	Pareto dominating policies
True Pareto front	2166	[-1,1],[-3,34],[-5,58],[-7,78],[-8,86],[-9,92], [-13,112],[-14,116],[-17,122],[-19,124]
No sPBRS	2166	[-1,1],[-3,34],[-5,58],[-7,78],[-8,86],[-9,92], [-13,112],[-14,116],[-17,122],[-19,124]
sPBRS (Good)	2166	[-1,1],[-3,34],[-5,58],[-7,78],[-8,86],[-9,92], [-13,112],[-14,116],[-17,122],[-19,124]
sPBRS (Poor)	2166	[-1,1],[-3,34],[-5,58],[-7,78],[-8,86],[-9,92], [-13,112],[-14,116],[-17,122],[-19,124]

causes more of the Pareto optimal policies to be sampled while learning; however, upon convergence it reaches the exact same hypervolume as that reached without $sPBRS$. The increased learning speed that is a characteristic of $sPBRS$ is displayed here, but the set of policies learned upon convergence has not been altered. The poor heuristic exhibits a reduced learning speed as expected, but it still learns the same final policies as agents learning without $sPBRS$ and with a good $sPBRS$ heuristic.

The results from the CDST domain are plotted in Figs. 5.3a and 5.3b, and the final policies learned and final hypervolumes are presented in Table 5.2. In this domain, the maximum hypervolume is 2166 when all 10 Pareto optimal policies are learned, calculated as before using a reference point of [-25,0]. The basic scalarised Q-learning agent is capable of learning all 10 Pareto optimal policies in the CDST, as evidenced by the final hypervolume, which reached the maximum value of 2166. When a good $sPBRS$ heuristic is added, the maximum hypervolume of 2166 is reached more quickly. Here $sPBRS$ has improved the learning speed, without altering the set of Pareto optimal policies learned for the problem. When using a poor $sPBRS$ heuristic, the learning speed is reduced, but the agent still converges to the maximum hypervolume, and successfully learns all 10 Pareto optimal policies.

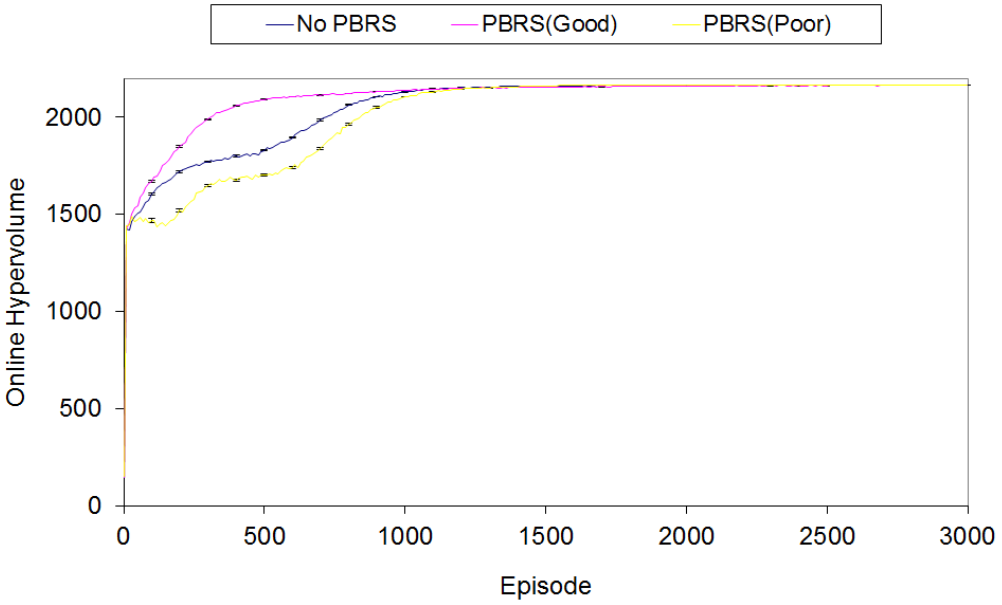


(a) Online hypervolume of non-dominated policies learned in the DST environment

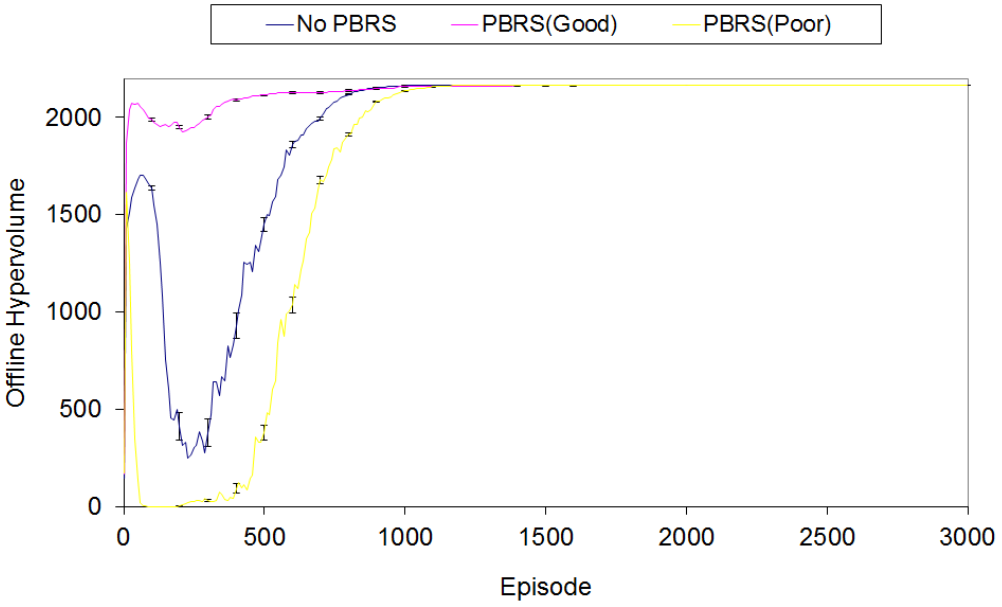


(b) Offline hypervolume of non-dominated policies learned in the DST environment

Figure 5.2: Deep Sea Treasure environment results



(a) Online hypervolume of non-dominated policies learned in the CDST environment



(b) Offline hypervolume of non-dominated policies learned in the CDST environment

Figure 5.3: Convex Deep Sea Treasure environment results

5.1.4 Discussion

The empirical results in this study demonstrate that the increased learning speed that is characteristic of $sPBRS$ can be leveraged in single-agent MORL problem domains, without altering the Pareto optimal policies learned upon convergence. An interesting finding is the effect of $sPBRS$ on an agent's exploration when a suitable heuristic is used. In the early stages of learning in the DST, scalarised Q-learning with a good heuristic achieves a high hypervolume initially, and samples Pareto optimal policies that are in the concave region of the Pareto front. However, upon convergence scalarised Q-learning with a good heuristic reaches the same hypervolume as scalarised Q-learning without $sPBRS$. In both the DST and CDST, it is evident that a well-designed $sPBRS$ heuristic can improve learning speed, and conversely that a poor heuristic can be harmful to an agent's initial performance. However, in all cases the set of Pareto optimal policies learned with and without $sPBRS$ is consistent in both problem domains, regardless of the quality of the heuristic used. Thus, I have demonstrated that the two main strengths of $sPBRS$ for RL in conventional MDPs (i.e. improved learning speed and policy invariance) can also be leveraged successfully with RL in MOMDPs.

5.2 Multi-Agent Benchmark Domain

In this study the Multi-Objective Beach Problem Domain (MOBPD) is introduced, a new Multi-Objective Stochastic Game which will serve as a benchmark problem for MARL algorithms. Up to now, the performance of MARL algorithms in multi-objective problems has been judged purely in relative terms, and there are no MOSGs in the literature to date where the true set of Pareto optimal solutions is known. Therefore the MOBPD will serve as a useful benchmark for future evaluations, as MARL algorithms can now be judged against a known absolute maximum level of performance, by comparing the hypervolume of non-dominated solutions learned with the hypervolume of the true Pareto front. Here the MOBPD will be used to empirically evaluate the effects of D and $sPBRS$ on agent coordination in MOSGs.

5.2.1 Multi-Objective Beach Problem Domain

The MOBPD extends an earlier single-objective version introduced by Devlin et al. (2014), in a similar manner to the multi-objective extension (Yliniemi & Tumer 2016) to the El-Farol bar problem (Arthur 1994). In the MOBPD, each tourist (agent) begins at a hotel on a specific beach section, and then decides at which section of the beach they will spend their day. At each timestep each agent knows which beach section $b \in B$ it is currently attending, and can choose to move to an adjacent section ($move\_left$ or $move\_right$), or to $stay\_still$. Once all agents have completed their selected actions they are rewarded. The agents must coordinate their actions to maximise the social welfare or global utility of the system, which is measured by two conflicting objectives: "capacity" and "mixture".

Each beach section has a certain capacity ψ, and the highest capacity reward for a section is received when the number of tourists (agents) present is equal to the capacity of the section. Sections which are either too crowded or too empty receive lower rewards as they are less desirable to the tourists. The local capacity reward $L_{cap}(b)$ for a particular section is calculated as:

$$L_{cap}(b) = x_b e^{\frac{-x_b}{\psi}} \tag{5.4}$$

where b is the beach section (state), and x_s is the number of agents present at that section. The global capacity utility can then be calculated as the summation of $L_{cap}(b)$ over all sections in the MOBPD:

$$G_{cap} = \sum_{b \in \mathbf{B}} L_{cap}(b) \tag{5.5}$$

Each agent in the MOBPD is assigned one of two static types: m or f. The maximum mixture reward for a section is received when the number of m agents in attendance is equal to the number of f agents, while sections with an unequal mixture of agents receive a lower reward as they are less desirable. The local mixture reward $L_{mix}(b)$ for a particular section is calculated as:

$$L_{mix}(b) = \frac{\min(|M_b|, |F_b|)}{(|M_b| + |F_b|) \times |B|} \tag{5.6}$$

where $|M_b|$ is the number of agents of type m present at that section, $|F_b|$ is the number of agents of type f present at that section, and $|B|$ is the total number of sections in the beach. The global mixture utility can then be calculated as the summation of $L_{mix}(b)$ over all sections in the MOBPD:

$$G_{mix} = \sum_{b \in B} L_{mix}(b) \tag{5.7}$$

The difference reward (D_i) for an agent can be calculated by applying Eqn. 2.21 for each objective. As an agent only influences the capacity or mixture utility of the section it is currently attending at a particular timestep, the utilities of all other states cancel out, and D_i may be calculated as:

$$D_{cap,i}(b) = L_{cap}(b) - (x_b - 1)e^{\frac{-(x_b-1)}{\psi}} \tag{5.8}$$

$$D_{mix,i}(b) = \begin{cases} L_{mix}(b) - \frac{\min(|M_b|-1, |F_b|)}{(|M_b|+|F_b|-1)\times|B|} & i \in m \\ L_{mix}(b) - \frac{\min(|M_b|, |F_b|-1)}{(|M_b|+|F_b|-1)\times|B|} & i \in f \end{cases} \tag{5.9}$$

Similar measures to those taken by Yliniemi & Tumer (2016) are implemented in order to ensure that the objectives are independent and that no trivial solutions exist. L_{mix} is maximised when an equal number of agents attend the same beach section; however odd values for ψ are used in experiments so that L_{cap} and L_{mix} cannot both be maximised at the same time at any one section. It is also ensured that there are many more agents than available capacity in the beach

Table 5.3: Normalisation constants

	Experiment 1	Experiment 2
L_{cap}^{min}	0.000	0.000
L_{cap}^{max}	1.105	1.840
L_{mix}^{min}	0.000	0.000
L_{mix}^{max}	0.101	0.101
G_{cap}^{min}	0.000	0.000
G_{cap}^{max}	4.416	7.359
G_{mix}^{min}	0.000	0.000
G_{mix}^{max}	0.460	0.460
D_{cap}^{min}	-0.134	-0.136
D_{cap}^{max}	0.718	0.820
D_{mix}^{min}	-0.034	-0.034
D_{mix}^{max}	0.101	0.101

sections, and that the proportion of m and f agents is not equal (70% of type m and 30% of type f are used in the experiments). The maximum G_{cap} value is achieved when most of the agents overcrowd one section, and exactly ψ agents attend each of the other sections. This is in conflict with the maximum G_{mix} scenario, where most of the agents overcrowd a section, and exactly 1 agent of type m and 1 agent of type f attend each of the remaining sections.

5.2.2 Applying MARL

Agents using credit assignment structures L, G, $G + sPBRS$ and D were tested on this problem domain, as well as agents that randomly select actions from a uniform distribution as a baseline. In the case of L and G, the components of the reward vector are first normalised (Eqn. 2.9) using the utopia and nadir values given in Table 5.3, and then scalarised using a linear combination (Eqn. 2.7). The normalised and scalarised combinations are then used in the agents' value function updates (Eqn. 2.2).

In the case of D, first each objective is shaped separately using its specific counterfactual value (Eqns. 5.8 and 5.9). The resultant shaped reward vector is then normalised (Eqn. 2.9) and scalarised (Eqn. 2.7) as above.

When applying $sPBRS$, the global reward vector is first normalised (Eqn. 2.9) then scalarised (Eqn. 2.7), before adding the shaping reward F (Eqn. 2.12) to the scalarised combination. Algorithm 4 shows the entire process that is followed when the agents in the MOBPD are rewarded with $G + sPBRS$. The following two heuristics adapted from the work of Devlin et al. (2014) were used to test the effect of $sPBRS$:

- **Middle:** All agents are invited to a party at the middle beach section ($b = 2$). This heuristic incorporates some basic knowledge about the optimal trade-off solutions, i.e. the

Algorithm 4 MOBPD with $G + sPBRS(Middle)$

initialise Q-values: $\forall b, a | Q(b, a) = 0$
for $episode = 1 \rightarrow num\_episodes$ **do**
    set initial agent positions
    for $timestep = 1 \rightarrow num\_timesteps$ **do**
        for $i = 1 \rightarrow num\_agents$ **do**
            sense current beach section b
            set potential $\Phi(b)$ (Eqn. 5.10)
            choose action a, using ϵ-greedy
            move agent to b'
            set potential $\Phi(b')$ (Eqn. 5.10)
        end for
        for all beach sections $b \in B$ **do**
            calculate local capacity reward $L_{cap}(b)$ (Eqn. 5.4)
            calculate local mixture reward $L_{mix}(b)$ (Eqn. 5.6)
        end for
        calculate global capacity reward G_{cap} (Eqn. 5.5)
        calculate global mixture reward G_{mix} (Eqn. 5.7)
        for $i = 1 \rightarrow total\_agents$ **do**
            set r = scalarised global reward (Eqn. 2.7)
            set f (Eqn. 2.12)
            set $r' = r + f$ (Eqn. 2.10)
            update $Q(b, a)$ values using r' (Eqn. 2.2)
        end for
        reduce ϵ using $epsilon\_decay\_rate$
    end for
    for $i = 1 \rightarrow num\_agents$ **do**
        choose action a, using ϵ-greedy
        move to absorbing state
        set $f = 0 - \Phi(b')$ (Eqn. 2.12)
        set $r' = 0 + f$ (Eqn. 2.10)
        update $Q(b', a)$ values (Eqn. 2.2)
    end for
end for

idea that one resource should be "sacrificed" or congested by most of the agents for the greater good of the system. This shaping is expected to improve both the performance and learning speed of agents receiving $sPBRS$, and will demonstrate the effect of $sPBRS$ when useful but incomplete domain knowledge is available.

$$\Phi(b) = \begin{cases} 1 & \text{if } b = 2 \\ 0 & \text{otherwise} \end{cases} \tag{5.10}$$

- **Spread:** The Spread heuristic encourages agents to distribute themselves evenly across the sections in the MOBPD. This is an example of a weak heuristic, and demonstrates the effect of $sPBRS$ in cases where very little useful domain knowledge is available. Therefore, agents receiving this shaping are expected to show modest if any improvements in learning speed and final performance.

$$\Phi(b) = \begin{cases} 1 & \text{if } b = 0, agent\_id \in [0, N/|B| - 1] \\ 1 & \text{if } b = 1, agent\_id \in [N/|B|, 2N/|B| - 1] \\ 1 & \text{if } b = 2, agent\_id \in [2N/|B|, 3N/|B| - 1] \\ 1 & \text{if } b = 3, agent\_id \in [3N/|B|, 4N/|B| - 1] \\ 1 & \text{if } b = 4, agent\_id \in [4N/|B|, 5N/|B| - 1] \\ 0 & \text{otherwise} \end{cases} \tag{5.11}$$

where N is the total number of agents.

5.2.3 Experimental Procedure

Two different empirical studies were conducted in the MOBPD. In the first experiment $\psi = 3$, $num\_agents\_M = 35$ and $num\_agents\_F = 15$, while in the second experiment $\psi = 5$, $num\_agents\_M = 70$ and $num\_agents\_F = 30$ in order to increase the complexity. Changing the parameters in this way produces separate, independent versions of the problem that each have a unique set of Pareto optimal system utilities. The sets of Pareto optimal utilities for both versions of the problem are listed in Table 5.4. These were determined by calculating G_{cap} and G_{mix} for each possible distribution of m and f agents among the beach sections, and then removing all dominated solutions. As the rewards for both objectives are normalised in the range $[0, 1]$, an even weighting of $[0.5, 0.5]$ was used when scalarising objectives.

The number of sections is set to $|B| = 5$, and the first $num\_agents\_M/2$ and $num\_agents\_F/2$ begin each episode at beach section 1, while the rest begin at beach section 3. 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 $\alpha = 0.1$, 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 values were

Table 5.4: MOBPD Pareto optimal system utilities

	Experiment 1		Experiment 2	
Soln. no.	G_{cap}	G_{mix}	G_{cap}	G_{mix}
1	4.107372	0.452381	5.362651	0.456522
2	4.134956	0.450000	5.819238	0.455556
3	4.162565	0.447368	6.275914	0.454545
4	4.190221	0.444444	6.732591	0.453488
5	4.217961	0.441176	7.189267	0.452381
6	4.239412	0.415315	7.199119	0.451220
7	4.267104	0.412381	7.208971	0.450000
8	4.288620	0.385965	7.218824	0.448718
9	4.316275	0.383333	7.228680	0.447368
10	4.337837	0.356410	7.231350	0.433012
11	4.365466	0.354054	7.241201	0.431852
12	4.414673	0.324561	7.251055	0.430633
13			7.260908	0.429351
14			7.273433	0.413659
15			7.283284	0.412500
16			7.293138	0.411282
17			7.315515	0.394321
18			7.325368	0.393165
19			7.357598	0.375000

selected following parameter sweeps to determine the best performing values.

All plots include error bars representative of the standard error of the mean based on 50 statistical runs. Specifically, the error is calculated 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.

5.2.4 Experimental Results

The results for both experiments are summarised in Tables 5.5 and 5.6. These tables list the number of true Pareto optimal solutions found across all runs (PO Solns.), the average hypervolume of the non-dominated solutions found on each statistical run (Avg. HV), and the hypervolume of the best non-dominated solutions found across all runs (Best HV). Best HV gives an indication of how close an approach can get to finding the true Pareto front of the problem, while Avg. HV shows how consistent the performance of an approach is. Figs. 5.4a and 5.4b show the average performance on the normalised scalarised global reward, while Figs. 5.5a and 5.5b show the average hypervolume of the non-dominated solutions found on each run.

Table 5.5: Experiment 1 results

	PO Solns.	Avg. HV	Best HV
True Pareto Front	12		1.980063
D	12	1.974039	1.980063
$G + PBRS(Mid)$	10	1.826471	1.978657
$G + PBRS(Spr)$	1	1.455105	1.856893
G	0	1.427198	1.853276
$Random$	0	1.377096	1.555496
L	0	1.187191	1.426849

Table 5.6: Experiment 2 results

	PO Solns.	Avg. HV	Best HV
True Pareto Front	19		3.347111
D	16	3.322784	3.329418
$G + PBRS(Mid)$	0	2.852388	3.238757
G	0	2.158390	2.474170
$G + PBRS(Spr)$	0	1.966866	2.300028
L	0	1.939231	2.338821
$Random$	0	1.609211	1.849685

The best non-dominated solutions found by each approach over all runs, as well as the true Pareto fronts are shown in Figs. 5.6a and 5.6b. The true Pareto fronts in both problem instances represent the best possible trade-offs between the two system objectives: capacity utility and mixture utility. The hypervolume may be used to measure the spread and diversity of a given solution set; therefore this metric may be used to determine how close a particular learning approach comes to the maximum possible level of performance in each instance of the domain. Learning approaches which find more of the true Pareto optimal set or whose NDS results in a higher hypervolume are deemed to perform better than learning approaches which discover few true Pareto optimal solutions or achieve a lower hypervolume.

D offered the best overall performance in both experiments, sampling all 12 Pareto optimal solutions in the first experiment, and 16 of 19 in the second experiment. $G + PBRS(Middle)$ sampled 10 Pareto optimal solutions in the first experiment, and none in the second. $G + PBRS(Spread)$ sampled a single Pareto optimal solution in the first experiment, and none in the second. Both of the typical MARL credit assignment structures L and G, as well as the random baseline failed to find any true Pareto optimal solutions. This highlights the fact that in even the simplest of multi-objective multi-agent problems, G alone may not be sufficiently informative to allow agents to find solutions that form part of the true Pareto optimal set, therefore justifying the need for more sophisticated credit assignment techniques.

Figs. 5.4a and 5.4b give an indication of the relative learning speed of the different approaches, measured using the return from the normalised scalarised system evaluation function. D again offers the best performance here, although $G + PBRS(Middle)$ almost matches it in the early episodes. As expected, L performs poorly here, as it does not encourage all agents to act in the system's best interest. $G + PBRS(Spread)$ performs poorly compared to $G + PBRS(Middle)$; as is the case in single-objective SGs, poorly designed potential functions with misleading information can damage system performance.

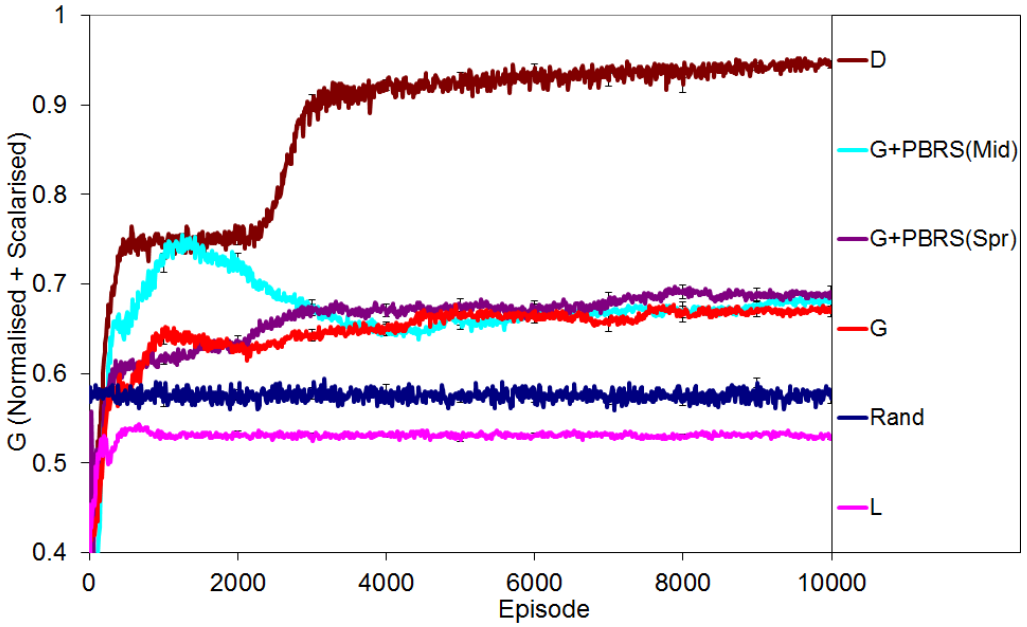
In Figs. 5.5a and 5.5b it is clear that $G + PBRS(Middle)$ samples a lot of solutions that are close to the Pareto front in the early stages of both experiments, resulting in a high hypervolume calculation, and initially beating the performance of D. This demonstrates the beneficial effect that $PBRS$ with a suitable heuristic can have on the agents' exploration. D initially samples promising solutions more slowly, but by the end of each experiment it has reached an average hypervolume very close to that of the true Pareto front (shown with a black dashed line in both plots). In terms of average hypervolume reached, D offers statistically better performance than $G + PBRS(Middle)$ in both experiment 1 ($p = 1.11 \times 10^{-17}$) and experiment 2 ($p = 2.67 \times 10^{-29}$). $G + PBRS(Middle)$ does however offer a statistically significant increase in performance over unshaped G on this metric in the first experiment ($p = 1.61 \times 10^{-26}$) and the second experiment ($p = 6.79 \times 10^{-47}$).

Figs. 5.6a and 5.6b show that the best non-dominated solutions found by D and $G + PBRS(Middle)$ match very closely with those of the true Pareto front in both experiments. The solutions found by L and G are dominated by those found by D and $G + PBRS(Middle)$; these typical MARL credit assignment structures are not informative enough to guide agents towards good solutions in the MOBPD.

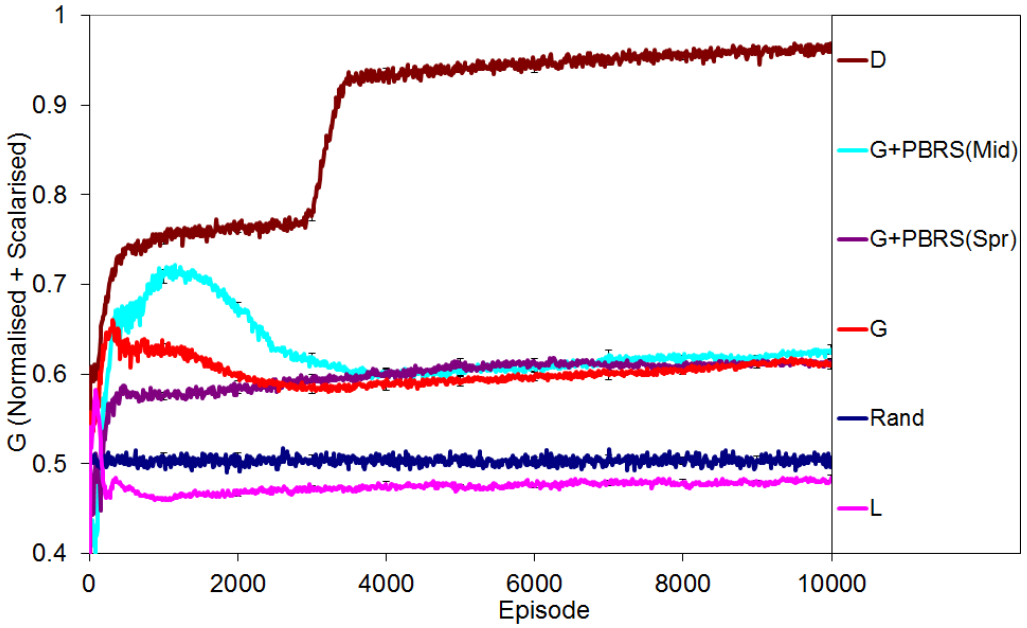
The performances of D and $G + PBRS(Middle)$ demonstrate that well designed reward shaping techniques can guide agents towards the true Pareto optimal solutions in MOSGs by making G more informative. Thus the issue of appropriate credit assignment is just as important in MOSGs as it is in traditional single-objective SGs. Furthermore, the results for D and $G + PBRS(Middle)$ offer the first supporting empirical evidence that both D and $PBRS$ preserve the true Pareto optimal sets of solutions in MOSGs. In the second experiment, both approaches do not perform as well due to the increased complexity of the problem. In the case of $PBRS$, more suitable heuristics could be designed to improve performance, although $G + PBRS(Middle)$ performs extremely well considering the simple nature of the information that is provided.

5.2.5 Discussion

This study evaluated the effectiveness of two widely-used reward shaping methodologies for improving agent coordination in cooperative MOSGs. The MOBPD was introduced, a new MOSG with known sets of Pareto optimal solutions that will serve as a useful benchmark for evaluating future MARL algorithms. The experimental work shows that both $PBRS$ and D can improve

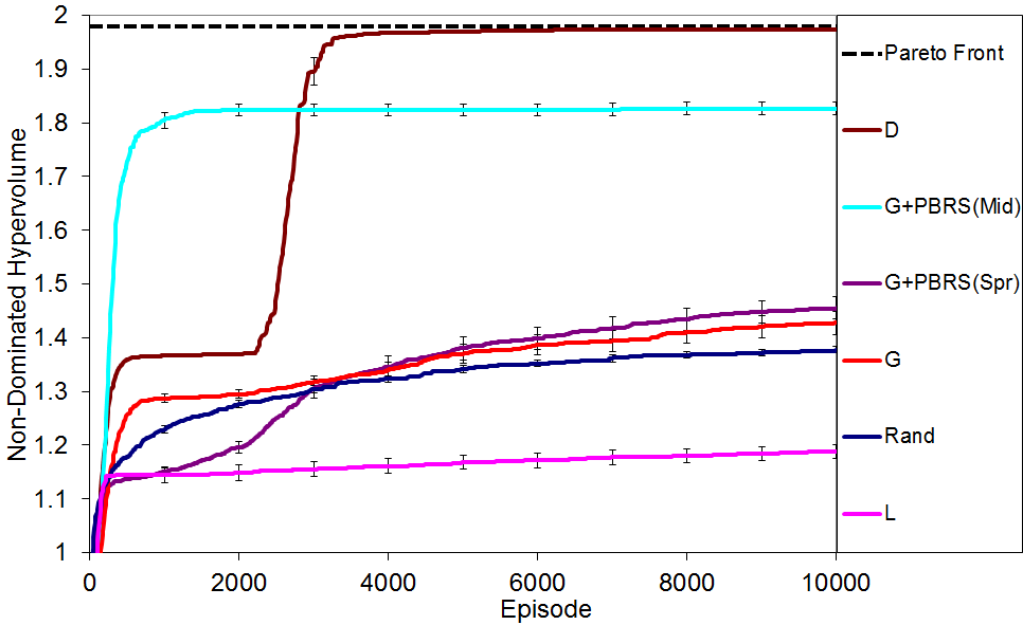


(a) $num\_agents\_M = 35$, $num\_agents\_F = 15$

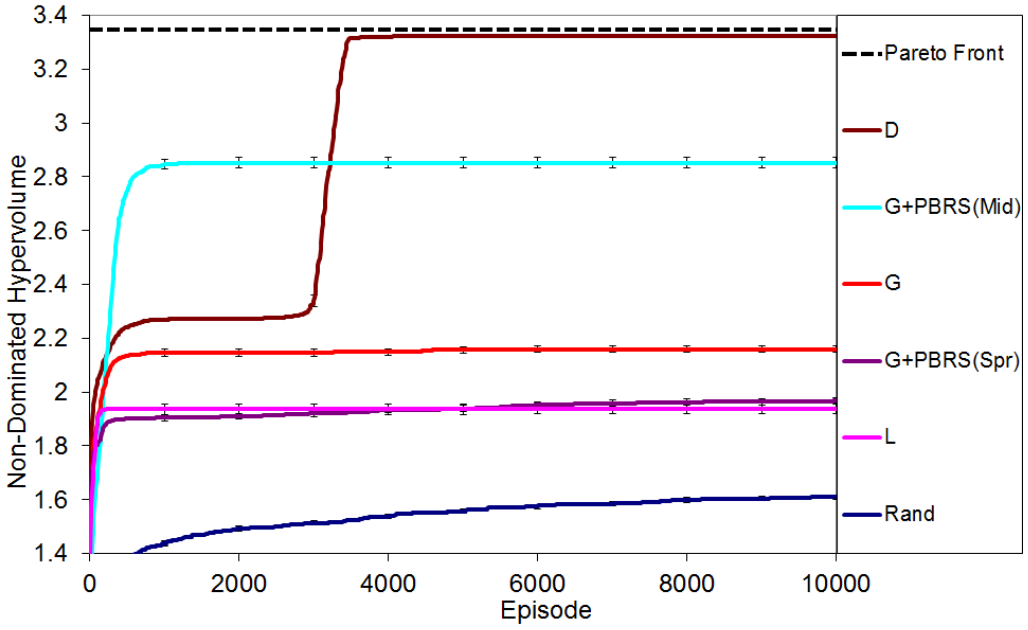


(b) $num\_agents\_M = 70$, $num\_agents\_F = 30$

Figure 5.4: Average performance on normalised scalarised global reward

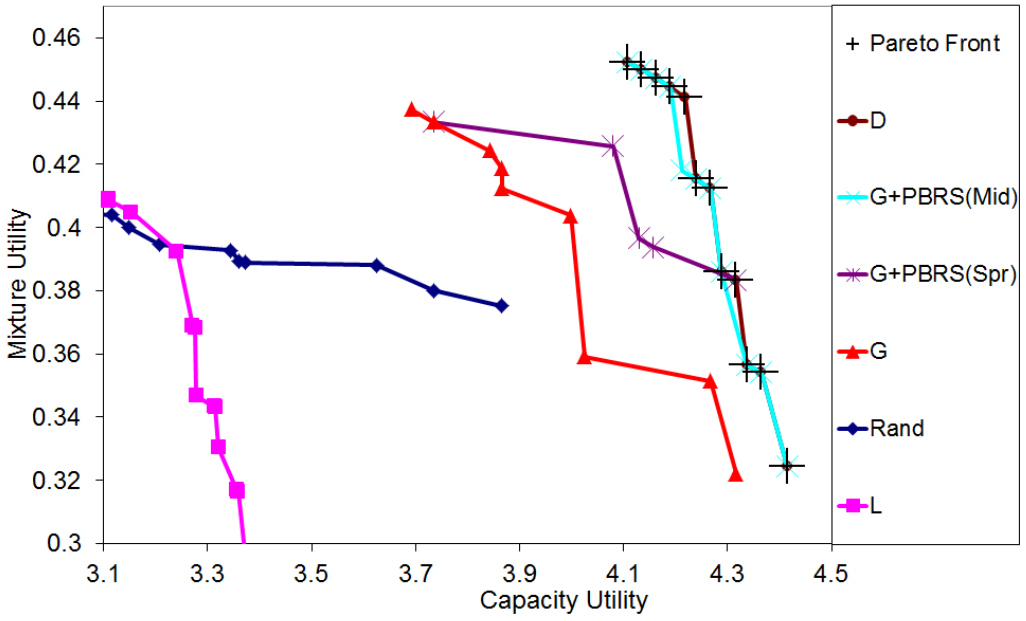


(a) $num\_agents\_M = 35$, $num\_agents\_F = 15$

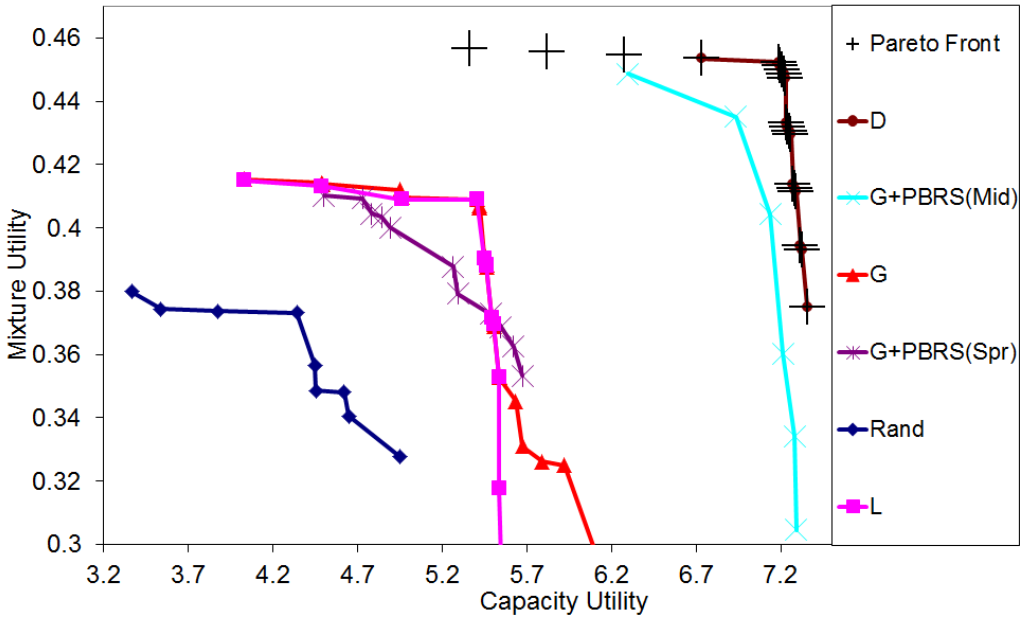


(b) $num\_agents\_M = 70$, $num\_agents\_F = 30$

Figure 5.5: Average hypervolume of non-dominated solutions found



(a) $num\_agents\_M = 35$, $num\_agents\_F = 15$



(b) $num\_agents\_M = 70$, $num\_agents\_F = 30$

Figure 5.6: Best non-dominated episodes over all runs

learning speed and the quality of the non-dominated set of solutions found in MOSGs, when compared to agents learning using G alone. Crucially, this work also demonstrated for the first time that agents learning using these reward shaping techniques can sample true Pareto optimal solutions in MOSGs. This provides further empirical support for the theoretical analysis presented in Chapter 4, showing that both D and $PBRS$ can safely be applied to MOSGs without the risk of altering the original goals of the problem.

5.3 Multi-Agent Application Domain

Now that the benefits of reward shaping in MOMARL have been established, it is important to test whether they will still hold in a real world application domain. In this study, D and $sPBRS$ will be applied to a Dynamic Economic Emissions Dispatch (DEED) problem. DEED is an established problem domain, that has previously been studied using approaches such as GAs (Basu 2008) and Particle Swarm Optimisation (PSO) (Mason 2015).

The problem consists of a series of electricity generators, which must be scheduled appropriately in order to meet a specified customer demand profile. Generator scheduling is a complex task due to many different factors, including: unpredictable fluctuations in demand; power loss within the transmission lines; and varying efficiency levels, power limits and ramp limits among generators in the same plant (Basu 2008).

High and often unpredictable fuel prices mean that efficient generator scheduling is necessary to produce electricity in a cost effective manner. However, it is also desirable to minimise the environmental impact of electricity production due to the emission of harmful atmospheric pollutants such as sulphur dioxide (SO_2) and nitrogen oxide (NO). Thus, the problem may be approached from a multi-objective perspective, with the goal of minimising both fuel cost and emissions.

Minimising both cost and emissions from power stations is a difficult problem, because these goals are in opposition to each other as the optimal solution for each objective is approached. This problem domain will serve as a testbed for evaluating the effectiveness of different MARL credit assignment structures while agents learn to optimise these conflicting objectives.

First, the traditional format of the DEED problem will be introduced in Section 5.3.1, before it is reformulated as a MOSG in Section 5.3.2 to allow the application of MARL. A new variant of the problem with random generator failure will also be tested, with the goal of testing the robustness and adaptability of agents to system disturbances. The empirical results obtained will be compared to those published previously (using e.g. GAs, PSO), to determine whether MARL can develop solutions for the DEED problem which are of comparable quality.

5.3.1 Dynamic Economic Emissions Dispatch

The version of the DEED problem which will be used for this study was originally proposed by Basu (2008). Basu's version is presented as a multi-dimensional optimisation problem, with each dimension in the problem space representing the power output of a generator at a given time. The cost function f_1 which computes the total fuel cost for the generators, including the effect of valve point loading (Walters & Sheble 1993), is defined as:

$$f_1 = \sum_{m=1}^{M} \sum_{n=1}^{N} [a_n + b_n P_{nm} + c_n (P_{nm})^2 + |d_n sin\{e_n (P_n^{min} - P_{nm})\}|] \tag{5.12}$$

where $M = 24$ is the number of hours, $N = 10$ is the number of power generators, a_n, b_n, c_n, d_n and e_n are the cost coefficients associated with each generator n, P_{nm} is the power output from generator n at time m, and P_n^{min} is the minimum permissible power output of generator n.

The total combined emissions of SO_2 and NO from the group of generators is calculated using function f_2 (Basu 2008):

$$f_2 = \sum_{m=1}^{M} \sum_{n=1}^{N} [\alpha_n + \beta_n P_{nm} + \gamma_n (P_{nm})^2 + \eta \exp \delta P_{nm}] \tag{5.13}$$

where α_n, β_n, γ_n, η_n and δ_n are the emission coefficients associated with each generator n.

The total power output in a given hour must be equal to the sum of the power demand and transmission losses:

$$\sum_{n=1}^{N} P_{nm} = P_{Dm} + P_{Lm} \qquad \forall m \in M \tag{5.14}$$

where P_{Dm} is the power demand over hour m and P_{Lm} is the transmission loss over hour m.

There are two inequality constraints which any potential solutions are subject to: the operating limits and the ramp limits for each power generator in the station. The operating limits specify the minimum and maximum possible power output of a generator, while the ramp limits determine the maximum allowed increase or decrease in the power output of a generator from one hour to the next.

$$P_n^{min} \leq P_{nm} \leq P_n^{max} \tag{5.15}$$

$$P_{nm} - P_{n(m-1)} \leq UR_n \tag{5.16a}$$

$$P_{n(m-1)} - P_{nm} \leq DR_n \tag{5.16b}$$

where P_n^{min} and P_n^{max} refer to the minimum and maximum power output of each generator, P_{nm} is the power output for $n \in N$ and $m \in M$, and UR_n and DR_n are the ramp up and ramp down limits for generator n.

In order to satisfy the equality constraint described by Eqn. 5.14, the first generator $n = 1$ is a slack generator. The power outputs of the other 9 generators are set directly, and the slack generator makes up any shortfall in the combined power output. The settings for the slack generator are therefore dependant variables during the optimisation process, while the outputs of the other $N - 1$ generators are independent variables. The power output of the slack generator for a single hour, P_{1m}, may be calculated by rearranging Eqn. 5.14:

$$P_{1m} = P_{Dm} + P_{Lm} - \sum_{n=2}^{N} P_{nm} \tag{5.17}$$

The loss in the transmission lines between generators, P_{Lm}, over hour m is calculated as follows:

$$P_{Lm} = \sum_{n=2}^{N}\sum_{j=2}^{N} P_{nm}B_{nj}P_{jm} + 2P_{1m}(\sum_{n=2}^{N} B_{1n}P_{nm}) + B_{11}(P_{1m})^2 \tag{5.18}$$

where B is the matrix of transmission line loss coefficients Basu (2008).

Combining Eqn. 5.17 with Eqn. 5.18 produces the following quadratic equation:

$$\begin{aligned} 0 = B_{11}(P_{1m})^2 + (2\sum_{n=2}^{N} B_{1n}P_{nm} - 1)P_{1m} + \\ (P_{Dm} + \sum_{n=2}^{N}\sum_{j=2}^{N} P_{nm}B_{nj}P_{nm} - \sum_{n=2}^{N} P_{nm}) \end{aligned} \tag{5.19}$$

Solving this quadratic equation using basic algebra will give the reactive power of the slack generator, P_{1m}, at each hour. All required values for the cost coefficients, emission coefficients, ramp limits, generator capacity limits, power demands and transmission line loss coefficients can be found in the work of Basu (2008).

5.3.2 DEED as a Multi-Objective Stochastic Game

In order to create a version of the Dynamic Economic Emissions Dispatch problem suitable for the application of MARL, it can be reformulated as a Multi-Objective Stochastic Game. The problem is divided into one of sequential decision making, where each hour $m \in M$ is a separate timestep in the SG. Each of the 9 directly controlled generators $n = \{2, ..., 10\}$ are assigned to an agent $i = \{2, ..., 10\}$, where agent i sets the power output P_{nm} of its generator $n = i$ at every timestep m.

It is now necessary to derive new cost and emissions functions, which will measure the system utility at each timestep. From Eqn. 5.12, a function f_c^L may be derived which computes the local cost for generator n over hour m:

$$f_c^L(n,m) = a_n + b_n P_{nm} + c_n(P_{nm})^2 + |d_n sin\{e_n(P_n^{min} - P_{nm})\}| \tag{5.20}$$

Thus the global cost function f_c^G for all generators over hour m is:

$$f_c^G(m) = \sum_{n=1}^{N} f_c^L(n,m) \tag{5.21}$$

Similarly, from Eqn. 5.13 an emissions function f_e^L may be developed for generator n over hour m:

$$f_e^L(n,m) = E(\alpha_n + \beta_n P_{nm} + \gamma_n(P_{nm})^2 + \eta \exp \delta P_{nm}) \tag{5.22}$$

where $E = 10$ is the emissions scaling factor, chosen so that the magnitude of the local emissions function f_e^L matches that of the local cost function f_c^L. It follows that the global emissions function f_e^G for all generators over hour m is:

$$f_e^G(m) = \sum_{n=1}^{N} f_e^L(n,m) \tag{5.23}$$

The next environmental state for each agent i is defined as a vector containing the change in power demand ΔP_D since the previous timestep, and the previous power output of the generator n, P_{nm}. The change in power demand at time m is calculated as:

$$\Delta P_{Dm} = P_{Dm} - P_{D(m-1)} \tag{5.24}$$

Therefore the state vector for agent i (controlling generator n) at time m is:

$$s_{i,m} = [\Delta P_{Dm}, P_{n(m-1)}] \tag{5.25}$$

The action chosen by agent i at each timestep determines the power output of the generator n under its control. However, the power output constraints in Eqn. 5.15 must be satisfied for each generator. Therefore the possible action set for agent i consists of:

$$A_i = \{P_n^{min}, ..., P_n^{max}\} \tag{5.26}$$

At any hour m, when the ramp limits in Eqns. 5.16a and 5.16b are imposed, an agent's action set is constrained to:

$$A_{im} = \{P_{n(m-1)} - UR_n \geq P_n^{min}, ..., P_{n(m-1)} - UR_n \leq P_n^{max}\} \tag{5.27}$$

It is also necessary to consider the power limits and ramp limits of the slack generator, $n = 1$. A global penalty function f_p^G based on the static penalty method (Smith et al. 2000) is used to capture violations of these constraints:

$$f_p^G(m) = \sum_{v=1}^{V} C(|h_v + 1|\delta_v) \tag{5.28}$$

$$h_1 = \begin{cases} P_{1m} - P_1^{max} & \text{if } P_{1m} > P_1^{max} \\ P_1^{min} - P_{1m} & \text{if } P_{1m} < P_1^{min} \\ 0 & \text{otherwise} \end{cases} \tag{5.29}$$

$$h_2 = \begin{cases} (P_{1m} - P_{1(m-1)}) - UR_1 & \text{if } (P_{1m} - P_{1(m-1)}) > UR_1 \\ (P_{1m} - P_{1(m-1)}) + DR_1 & \text{if } (P_{1m} - P_{1(m-1)}) < -DR_1 \\ 0 & \text{otherwise} \end{cases} \tag{5.30}$$

where $|V| = 2$ is the number of constraints handled using this method (one possible violation each for slack generator power and ramp limits over hour m), $C = 10E6$ is the violation constant, h_v is the violation of each constraint, and $\delta_v = 0$ if there is no violation in a given constraint and $\delta_v = 1$ if the constraint is violated. The violation constant $C = 10E6$ was selected so that the output of the penalty function will have a similar magnitude to that of the cost function f_c^G. The penalty function is an additional objective that must be optimised, in addition to cost and emissions.

The counterfactual cost, emissions and violations terms for an agent i are calculated by assuming that the agent did not choose a new power output value in the timestep m (i.e. the power output of generator $n = i$ did not change):

$$f_c^{G(-i)}(m) = \sum_{\substack{n=1 \\ n \neq i}}^{N} f_c^L(n, m) + f_c^L(i, m-1) \tag{5.31}$$

$$f_e^{G(-i)}(m) = \sum_{\substack{n=1 \\ n \neq i}}^{N} f_e^L(n, m) + f_e^L(i, m-1) \tag{5.32}$$

The output of the counterfactual version $f_p^{G(-i)}$ of the penalty function f_p^G is calculated as per Eqn. 5.28, with the term $P_{1m}^{(-i)}$ substituted for P_{1m} in Eqns. 5.29 and 5.30. $P_{1m}^{(-i)}$ is calculated as:

$$P_{1m}^{(-i)} = P_{Dm} + P_{Lm} - \sum_{\substack{n=2 \\ n \neq i}}^{N} P_{nm} - P_{i(m-1)} \tag{5.33}$$

5.3.3 Action Selection

In initial experimental work on the DEED MOSG using the full action definitions in Eqns. 5.26 and 5.27, the quality of the policies learned was highly variable, often resulting in poor performance. This may be attributed to the fact that the action space A_i for each agent is of a different size. For example, using a discretisation level of 1MW, the smallest action space has 46 actions, and the largest has 321 actions when using the generator operating limits specified by Basu (2008). These discrepancies meant the time required for each agent to sample the full state-action space varied widely. To overcome this difficulty, an abstraction A^* of the action space will be used, where each agent has a set of 101 possible actions $A^* = \{0, 1, ..., 99, 100\}$. Each action represents a different percentage value of the operating range of the generator, so generators with wider operating ranges have larger increments. The power output from generator n for action a_i^* is calculated as:

$$P_n = P_n^{min} + a_i^* \left(\frac{P_n^{max} - P_n^{min}}{100} \right) \qquad i = n \tag{5.34}$$

The power output selected by an agent is still subject to the ramp limits, as per Eqns. 5.16a, 5.16b and 5.27, so a^* selections that would violate these limits are not allowed. This action space abstraction is used in all experimental work presented in this study. Agents select actions from A^* using the ϵ-greedy strategy.

5.3.4 Applying MARL

Multiple individual Q-learning agents were tested in the DEED MOSG defined above, learning with credit assignment structures L, G, $G + PBRS$, $G + CaP$ and D. All reward vectors are scalarised using either a linear scalarisation (+) (Eqn. 2.7) or a hypervolume scalarisation (λ) (Eqn. 2.8). The objective weights used for linear scalarisation are: $w_c = 0.225$, $w_e = 0.275$, and $w_p = 0.5$. These objective weights were chosen following parameter sweeps, so as to maintain a good balance between the objectives. The agents receive one of these scalarised reward signals while learning: $L(+)$, $L(\lambda)$, $G(+)$, $G(\lambda)$, $G + PBRS(+)$, $G + CaP(+)$, $G + CaP(\lambda)$, $D(+)$ or $D(\lambda)$.

Note that in the case of a local reward L, the number of objectives $|C| = 2$ as there is no local penalty function. $|C| = 3$ for all other credit assignment schemes, as they all make use of the global versions of the objective functions. Note also that the rewards assigned are negative, as all objectives must be minimised.

In the case of D, each objective is first shaped separately using its specific counterfactual value (Eqns. 5.31, 5.32 and 5.33) before being scalarised. When applying $G + PBRS$ and

$G + CaP$, the global reward vector is first scalarised, before adding the shaping reward F (Eqn. 2.12) to the scalarised combination. Three different heuristics were used to test effectiveness of $sPBRS$ in this problem:

- **High:** All agents are encouraged to select high power values. This heuristic is expected to quickly reduce the cost and emissions values during learning, and lead to good solutions.

$$\Phi_{High}(i,m) = -\left(100 + \left(\frac{P_{n,m-1} - P_n^{min}}{P_n^{max} - P_n^{min}} \times 100\right)\right) \times 10^6 \qquad i = n \tag{5.35}$$

- **Low:** All agents are encouraged to select low power values. This heuristic is also expected to increase learning speed, although encouraging agents to select low power values will increase loading on the slack generator, which may negatively affect the running costs and emissions produced.

$$\Phi_{Low}(i,m) = -\left(100 - \left(\frac{P_{n,m-1} - P_n^{min}}{P_n^{max} - P_n^{min}} \times 100\right)\right) \times 10^6 \qquad i = n \tag{5.36}$$

- **Mixed:** This heuristic encourages a mixture of the two different behaviours above. The agents $i = 2$ to $i = 5$ are encouraged to select low power values using the low heuristic, whereas the agents from $i = 6$ to $i = 10$ are encouraged to select high power values using the high heuristic. The design of this mixed heuristic is based on the intuition that it may be beneficial to keep some generators in a group working at close to full power continuously to satisfy the baseline power demand, while the rest of the generators will only increase their power output during peaks of high demand.

$$\Phi_{Mixed}(i,m) = \begin{cases} \Phi_{Low}(i,m) & \text{if } i \leq 5 \\ \Phi_{High}(i,m) & \text{otherwise} \end{cases} \tag{5.37}$$

5.3.5 Experimental Procedure

Experiments were conducted on two different variations of the DEED MOSG. In the normal version of the problem, the agents learn for $20,000$ episodes, each of which comprises 24 hours. The second version also lasts for 20,000 episodes; after 10,000 episodes a random generator $n \in \{2, ..., 10\}$ fails, and the agents must learn to compensate for the loss of this generator, while still meeting the same electricity demand. The aim of this second experiment is to test the robustness to disturbances and adaptability of agents learning using each MARL credit assignment structure.

The demand profile used in both experiments is shown in Fig. 5.7. This is the same demand

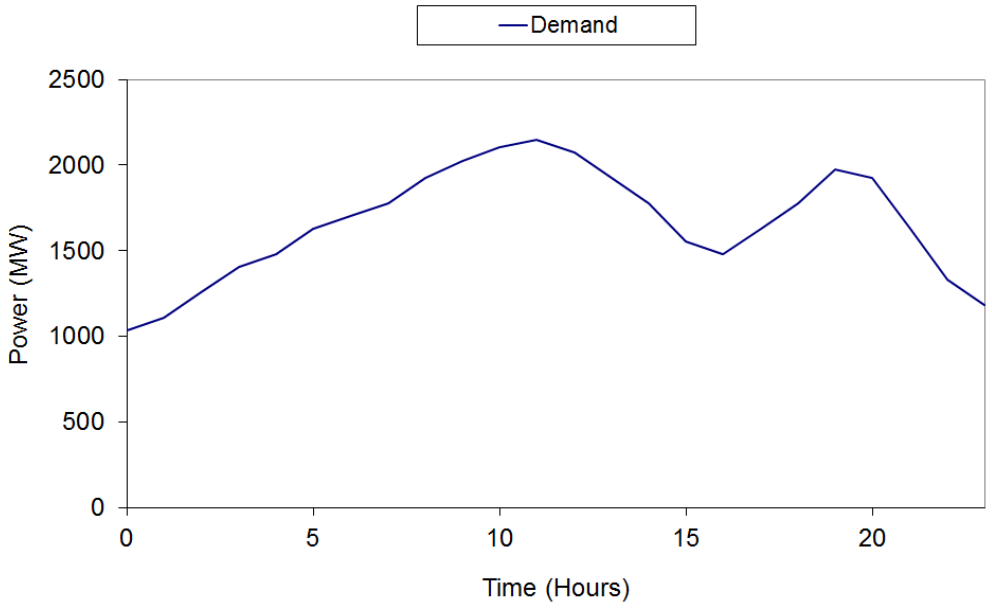


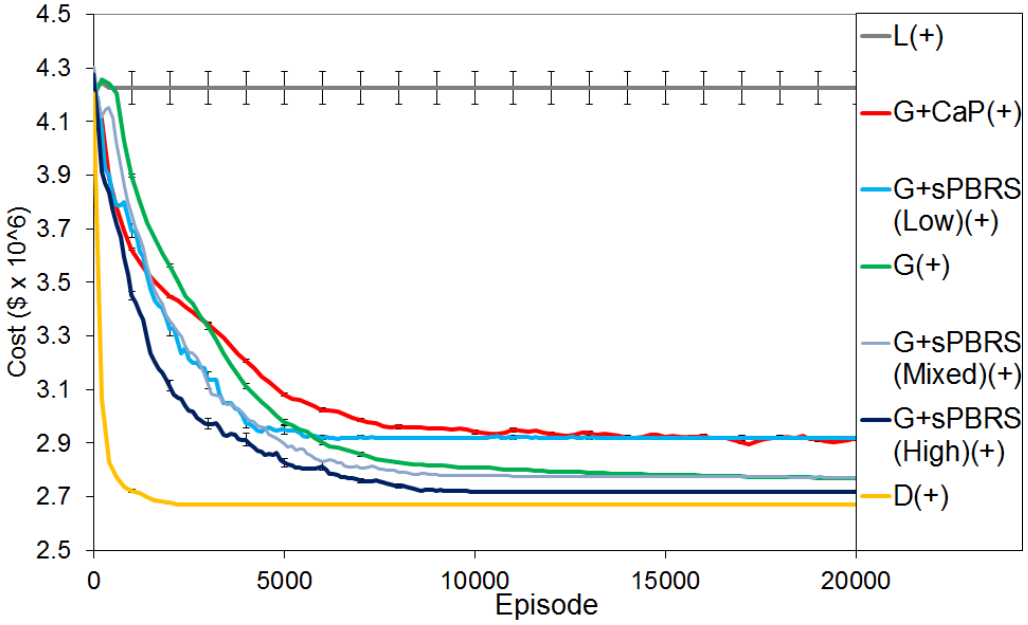
Figure 5.7: 24 hour power demand

profile that was used in used in work by Basu (2008) and Mason (2015), so the DEED MOSG results will be directly comparable to results reported by these authors. The learning parameters for all agents are set as follows: $\alpha = 0.10$, $\gamma = 0.75$, $\epsilon = 0.05$. These values were selected following parameter sweeps to determine the best performing settings.

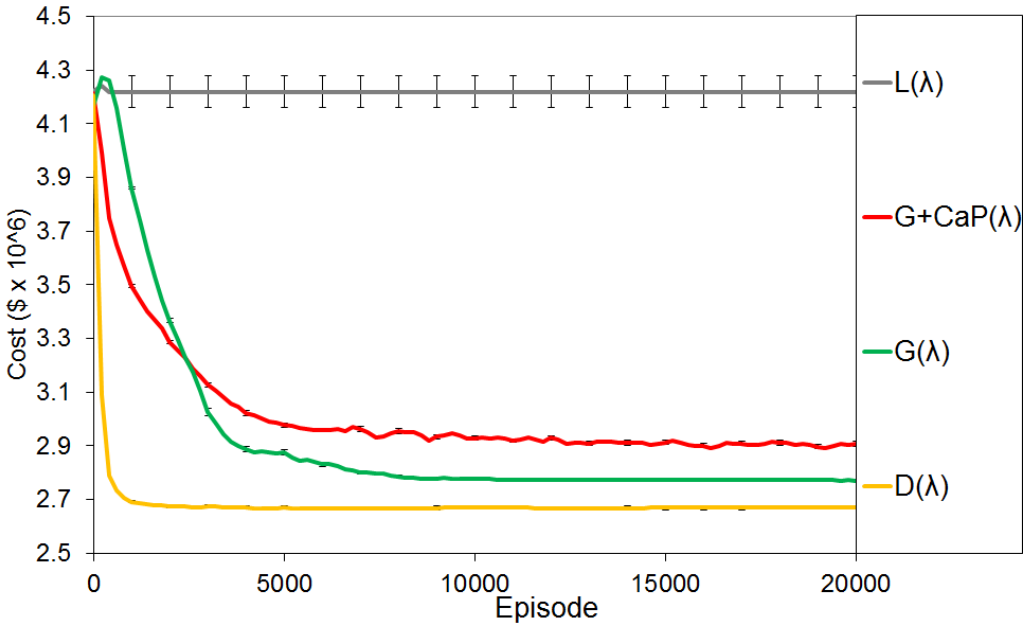
5.3.6 Experimental Results

I will first discuss the results of the standard version of the problem. All plots include error bars at 1000 episode intervals representative of the standard error of the mean based on 50 statistical runs. Specifically, the error is calculated as $\sigma/\sqrt{n}$ where σ is the standard deviation and n is the number of statistical runs. The plots show a 200 episode moving average across the 50 statistical runs that were conducted. 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.

The plots of learning curves for the cost objective in the first experiment (Figs. 5.8a and 5.8b) give an indication of the relative learning speeds and stability of performance for each of the approaches tested. As expected, L performs poorly here, as the local reward encourages agents to greedily minimise their own fuel cost, without considering the utility of the system as a whole. D converges to a stable policy most quickly with both scalarisations, while both variants of G learn good policies, but at a slower rate than D. $G+CaP$ initially learns more quickly than G for both scalarisations; increased learning speed is a typical characteristic of PBRS. However, the final joint policies learned by CaP are not as good as those learned by G or D.



(a) With linear scalarisation



(b) With hypervolume scalarisation

Figure 5.8: Learning curves for the cost objective

Table 5.7: DEED average final performance

	Cost (\$ $\times 10^6$)	Emissions (lb $\times 10^5$)
$L(\lambda)$	4.1149	17.6606
$L(+)$	4.1127	28.8266
$G + sPBRS(Low)(+)$	2.9197	4.5288
$G + CaP(\lambda)$	2.8919	9.6431
$G + CaP(+)$	2.8777	7.4774
$G(\lambda)$	2.7607	3.9788
$G(+)$	2.7647	3.9098
$G + sPBRS(Mixed)(+)$	2.7722	3.7531
$G + sPBRS(High)(+)$	2.7177	3.6828
$D(\lambda)$	2.6748	3.8980
$D(+)$	2.6641	3.3255
PSO-AWL (Mason 2015)	2.5463	2.9455
NSGA-II (Basu 2008)	2.5226	3.0994

$G+sPBRS(High)(+)$ and $G+sPBRS(Mixed)(+)$ both learn more quickly than $G(+)$. $G + sPBRS(Low)(+)$ also initially learns more quickly than $G(+)$; however, it does so at the expense of damaging final performance when compared to unshaped $G(+)$. Here it can be seen again that the characteristic effects of $sPBRS$ in SGs are preserved in MOSGs; well designed heuristics can improve learning speed and final performance in MOSGs, while poorly designed heuristics may damage learning speed and final performance.

Table 5.7 displays the average cost and emissions for the MARL approaches tested, along with NSGA-II results reported by Basu (2008) and PSO-AWL results reported by Mason (2015) for comparison purposes. In this table, the power is presented in MW, the cost is presented in \$ $\times 10^6$ and the emissions are presented in lb $\times 10^5$. All values in the table are rounded to 4 decimal places.

The differences in the mean final performance of $D(+)$ and $G(+)$ were found to be significant for both the cost objective ($p = 5.01 \times 10^{-22}$), and the emissions objective ($p = 3.20 \times 10^{-10}$). $D(+)$ also performed statistically better than $G + sPBRS(High)(+)$ on both the cost objective ($p = 1.06 \times 10^{-8}$) and the emissions objective ($p = 7.01 \times 10^{-16}$). $G + sPBRS(High)(+)$ performed statistically better than $G(+)$ in terms of cost ($p = 2.99 \times 10^{-10}$) and emissions ($p = 2.63 \times 10^{-4}$), while $G + sPBRS(Mixed)(+)$ was statistically the same as $G(+)$ in terms of cost ($p = 0.398$), and statistically better in terms of emissions ($p = 2.61 \times 10^{-3}$). The misleading shaping $G + sPBRS(Low)(+)$ caused a significant drop in average performance when compared with $G(+)$ in terms of both cost ($p = 1.13 \times 10^{-33}$) and emissions ($p = 1.7 \times 10^{-6}$).

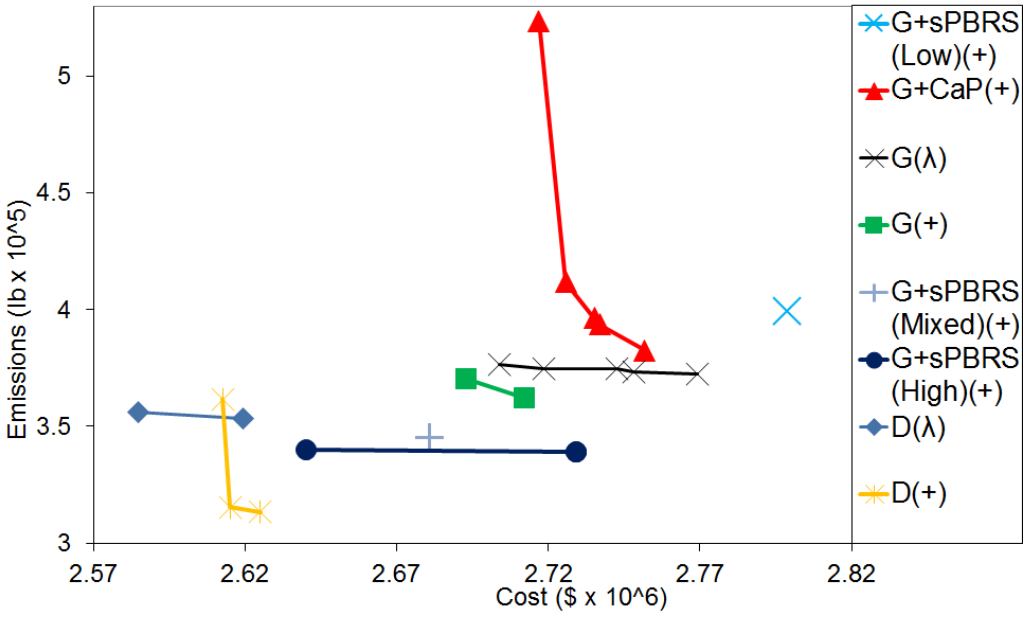
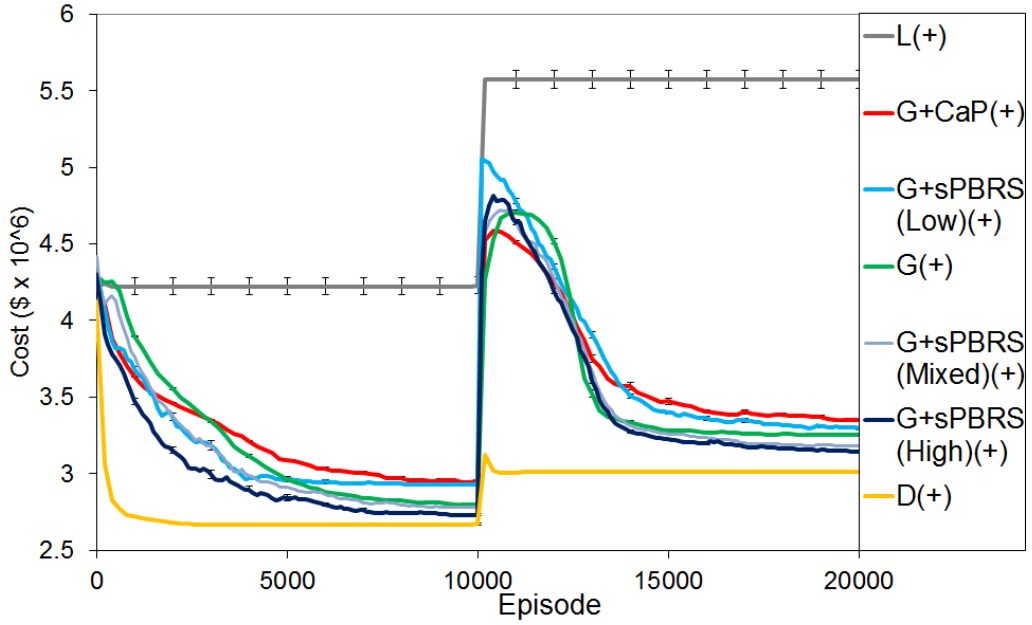


Figure 5.9: Best non-dominated episodes over all runs

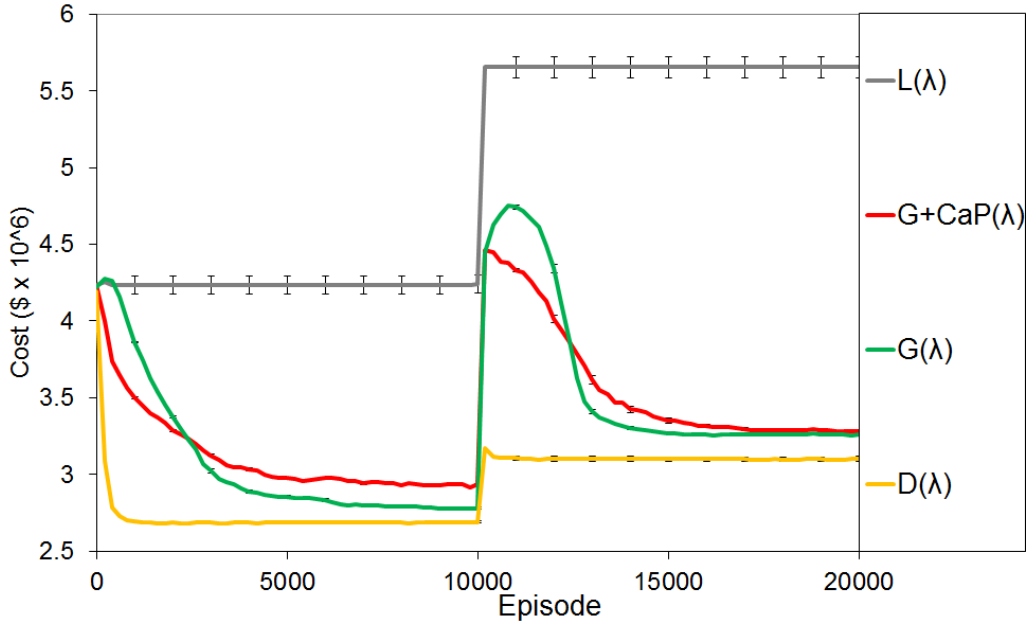
No statistical difference was found between the mean final performance of the scalarisation approaches for $G(+)$ and $G(\lambda)$, or for $CaP(+)$ and $CaP(\lambda)$. The differences in the means between $D(+)$ and $D(\lambda)$ were statistically insignificant for the cost objective, but were significant for the emissions objective ($p = 1.19 \times 10^{-8}$). These results suggest that appropriate credit assignment is far more important than the choice of scalarisation function used in this problem domain, although in general a linear scalarisation seems to provide slightly better results on average than a hypervolume scalarisation.

Analysing the average results presented in Table 5.7, the best MARL approach produces results that are comparable to those of GA and PSO based approaches, although not quite as good. For example, Basu's NSGA-II has 4.2% lower costs, and 6.8% lower emissions than $D(+)$ on average in this problem. However, MAS is arguably a more interesting paradigm to use when studying these types of optimisation problems, due to the ability to modify simulation parameters while learning online, and the possibility of modelling system disturbances (e.g. generator failure). MAS are inherently suited to distributed control and optimisation problems like DEED, and the information learned by the agents can be used to compute new policies online if the parameters of the problem change, unlike the GA and PSO based approaches.

Fig. 5.9 plots the Pareto fronts for the best performing credit assignment structures tested. These fronts are comprised of the best non-dominated episodes produced by each approach over 50 runs conducted in the first experiment. D again offers the best performance here, sampling solutions that Pareto dominate those found by all other approaches. $G + sPBRS(High)(+)$ is



(a) With linear scalarisation



(b) With hypervolume scalarisation

Figure 5.10: Effect of random generator failure

Table 5.8: Sample solution produced by D(+)

Hour	P1	P2	P3	P4	P5	P6	P7	P8	P9	P10
1	137.4166	274.3069	125.8713	159.8020	121.8119	69.2376	62.4752	52.7822	40.1980	13.1188
2	179.3413	244.4554	133.8020	114.6535	145.3762	85.5545	73.3663	80.2475	50.8911	26.0396
3	272.3167	244.4554	136.4455	124.1584	152.1089	94.7327	79.9010	91.0891	63.9604	30.0495
4	287.9734	244.4554	149.6634	164.5545	187.4554	100.8515	94.0594	101.2079	70.4950	43.8614
5	229.8747	251.0891	176.0990	174.0594	219.4356	114.1089	120.1980	110.6040	74.6535	51.4356
6	264.1131	314.1089	197.2475	178.8119	226.1683	131.4455	122.3762	117.8317	75.2475	51.8812
7	282.3970	290.8911	207.8218	214.4554	237.9505	156.9406	122.3762	117.1089	75.2475	52.3267
8	307.7624	294.2079	247.4752	219.2079	241.3168	157.9604	122.3762	117.1089	75.2475	54.1089
9	382.7586	337.3267	250.1188	250.0990	241.3168	157.9604	125.6436	118.5545	78.8119	54.5545
10	351.5082	380.4455	295.0594	295.2475	241.3168	158.9802	126.7327	119.2772	79.4059	54.5545
11	395.0589	406.9802	313.5644	297.6238	241.3168	158.9802	127.8218	119.2772	79.4059	54.5545
12	408.5579	430.1980	324.1386	297.6238	241.3168	158.9802	128.9109	119.2772	79.4059	54.5545
13	358.1410	383.7624	334.7129	297.6238	241.3168	158.9802	128.9109	119.2772	79.4059	54.5545
14	303.8088	353.9109	258.0495	297.6238	241.3168	158.9802	128.9109	119.2772	79.4059	54.5545
15	242.0132	314.1089	199.8911	297.6238	241.3168	158.9802	128.9109	119.2772	79.4059	54.5545
16	164.7548	241.1386	170.8119	292.8713	205.9703	153.8812	128.9109	119.2772	79.4059	41.6337
17	150.3090	217.9208	170.8119	276.2376	205.9703	131.4455	126.7327	109.1584	79.4059	52.3267
18	209.4446	261.0396	221.0396	266.7327	214.3861	136.5446	126.7327	109.1584	79.4059	53.2178
19	282.0803	337.3267	226.3267	250.0990	224.4851	147.7624	126.7327	109.1584	79.4059	53.6634
20	328.8346	413.6139	273.9109	269.1089	236.2673	153.8812	127.8218	112.0495	79.4059	54.1089
21	286.2176	343.9604	326.7822	269.1089	236.2673	155.9208	127.8218	116.3861	79.4059	53.6634
22	161.0515	340.6436	281.8416	238.2178	209.3366	107.9901	124.5545	94.7030	68.1188	52.3267
23	127.0397	264.3564	210.4653	188.3168	199.2376	91.6733	94.0594	83.1386	64.5545	42.5248
24	156.5331	211.2871	170.8119	185.9406	152.1089	115.1287	64.6535	67.9604	52.0792	33.6139
Cost	2.6561	Emissions	3.1674							

the next best performer, with one of its episodes Pareto dominating those of all other approaches except those of D. $G + sPBRS(Mixed)(+)$ also produced an episode that Pareto dominated all episodes produced by unshaped G; these results again confirm that as well as learning more quickly, agents receiving $sPBRS$ with a well designed heuristic can also significantly outperform agents learning using unshaped G in MOSGs. Finally, the misleading information encoded in $G + sPBRS(Low)(+)$ damaged the performance of agents receiving it, evidenced by the fact that all of its episodes are Pareto dominated by unshaped G.

A sample 24 hour generator schedule produced by $D(+)$ is presented in Table 5.8, showing the outputs in MW of the 9 controlled generators (P2 to P10), and the slack generator (P1). Sample generator schedules for GA and PSO approaches are available in the work of Basu (2008) and Mason (2015).

Figs. 5.10a and 5.10b show the learning curves for the cost objective in the second experiment, where a random generator fails. Both variants of L again perform poorly in this experiment. Similar to the first experiment, CaP initially learns more quickly than G, but converges to poorer joint policies on average. D is again the best performing reward structure here, and both variants converge to a stable joint policy after generator failure much more quickly than any other reward structure tested.

The agents learning using D are exceptionally robust to disturbances in this problem domain when compared to agents learning using the other credit assignment structures. Agents learning using $G + sPBRS(High)(+)$ and $G + sPBRS(Mixed)(+)$ again outperform agents using unshaped G in this experiment, learning to adapt to generator failure more quickly, and converging to better joint policies on average.

5.3.7 Discussion

In this study, a multi-objective, real world problem domain was analysed using the MAS paradigm. The DEED domain was reformulated as a sequential decision making problem using the framework of Multi-Objective Stochastic Games, in order to allow the application of Multi-Agent Reinforcement Learning. Furthermore, the effects of applying several different multi-agent credit assignment structures and two different scalarisation techniques were evaluated empirically.

The results show that difference rewards achieved the best overall performance in this problem domain, and that a linear objective scalarisation (+) was somewhat more effective than a hypervolume scalarisation (λ). The best MARL experiment $D(+)$ produced results that are comparable to other previously published attempts at solving this problem domain, including NSGA-II (Basu 2008) and PSO (Mason 2015). Difference rewards were also found to be more robust to disturbances than the other MARL credit assignment structures tested, and they effectively encouraged agents to adapt in the generator failure scenario, and to quickly learn new stable joint policies.

In future work, it would be worthwhile to investigate the use of value function approximation in this domain, as the ability to generalise across states and/or actions would be useful when developing agents that could react quickly to previously unseen changes in power demand, e.g. as would occur in a real world system.

5.4 Conclusion

This chapter presented three studies that evaluate the effect of two widely-used reward shaping techniques in MORL domains. These are the first reported empirical studies to test the effect of $sPBRS$ in MOMDPs and MOSGs. The second and third studies also considered the effect of D in MOSGs, and further strengthen the case for applying D in multi-objective multi-agent domains which was first made by Yliniemi (2015) and Yliniemi & Tumer (2016). Furthermore, these studies in MOSGs confirm that G is not sufficiently informative to lead agents to good solutions in MOMARL domains; appropriate credit assignment is therefore just as important in MOSGs as it is in traditional single-objective SGs.

The results of the first study in the DST and CDST domains confirmed my hypothesis that agents receiving $sPBRS$ will converge to the same policies as agents receiving unshaped rewards from the environment in MOMDPs, regardless of the quality of the heuristic information that is used. Analogously to the case in MDPs with a single objective, experiments demonstrated that well designed $sPBRS$ heuristics can improve learning speed in MOMDPs, whereas misleading heuristics can damage learning speed in MOMDPs.

The second study introduced the MOBPD, the first MOSG in the literature where the true Pareto optimal system utilities are known. This benchmark problem can now be used to evaluate MOMARL techniques against a known absolute maximum level of performance, by comparing the hypervolume achieved by an approach with the hypervolume of the true Pareto front. Results from this domain showed for the first time that agents learning using either $PBRS$ or D can sample true Pareto optimal solutions in MOMARL domains.

The final study in this chapter applied MOMARL to an established multi-objective optimisation problem; Dynamic Economic Emissions Dispatch. The traditional format of the problem was reformulated as a MOSG, which served as an additional benchmark to evaluate the effect of $PBRS$ and D in MOMARL domains. Furthermore, the experimental results of the best MOMARL approach tested were comparable to those achieved by other state-of-the-art optimisation algorithms. This study also elaborated on the theme explored in Chapter 3, offering further insights into how useful potential functions may be developed for complex real world applications.

While difference evaluations offered the best performance across all metrics the second and third studies, they suffer from some notable limitations: global knowledge about the system state and joint action must be available, and the precise mathematical form of the system evaluation function G must be known in order to calculate counterfactuals. Furthermore, D requires an implicit assumption that a centralised mechanism is available to provide tailored feedback to

individual agents (Colby et al. 2016), and the system designer must also select suitable default states and actions that allow a successful implementation.

Potential-Based Reward Shaping does not suffer from the limitations listed above. In MOSGs, agents may each have their own private potential function, and do not need to have any additional knowledge broadcast to them besides the value of the system evaluation function G for a successful implementation. Of course it is also possible to design potential functions that incorporate some level of additional knowledge about the system, but this does not require total observability of the joint states and actions of agents, or the mathematical form of the system evaluation function, as is the case with D.

However, $PBRS$ does have its own specific set of limitations. The process of handcrafting useful potential functions can be very time consuming, and effective potential functions will become increasingly difficult to design in proportion with the complexity of the application domain. In both MOSG studies even the best handcrafted $PBRS$ heuristics failed to match the performance of D; this highlights the effectiveness of difference evaluations in cases where the required constraints are satisfied such that they may be easily applied.

In summary, it is not possible to say that either one of these techniques represents the best all-around candidate for improving cooperative MOMARL performance. Rather, I expect that the preferred technique for a given MOMARL application will depend on the specific constraints present; e.g. whether the form of the system evaluation function is known, the amount of bandwidth available to communicate information to agents in a MAS, the system designer's level of domain knowledge and prior experience with either technique, and the time available to fine tune an implementation.

Given the empirical results presented in this chapter, and the theoretical analysis conducted in Chapter 4, this thesis will inform future MOMARL research. Following the establishment of the MOBPD, researchers may now use this benchmark domain to evaluate the performance of new MOMARL algorithms. It is also now possible to leverage the benefits of reward shaping in MOMARL without the risk of altering an agent's originally intended goals if D or $PBRS$ are applied. Furthermore, this chapter has demonstrated that these reward shaping techniques can address the credit assignment problem in MOSGs, substantially improving agents' performance and learning speed when compared to the unshaped system evaluation function.

CHAPTER 6

Conclusion

Now that the theoretical and empirical results have been presented, this chapter concludes with an overview of the main contributions, limitations, expected impact and future research questions arising from this work.

Section 6.1 provides a summary of the core contributions from the previous chapters, while Section 6.2 discusses their expected impact on future research. Section 6.3 identifies the most significant limitations of this work, and Section 6.4 discusses how they may be addressed, as well as some promising directions which may be worth exploring further in future research. Finally, Section 6.5 concludes this thesis with some final remarks.

As stated in Chapter 1, the core research questions which this thesis is intended to explore are:

1. To what extent can reward shaping be applied to MORL in a theoretically sound manner, without the risk of altering an agent's intended goals? (RQ1)
2. Will existing reward shaping methods improve performance and/or learning speed in MORL domains? (RQ2)
3. Can a benchmark problem be established for MOMARL where the true Pareto optimal solutions are known? (RQ3)

Following from the investigations in this thesis, these questions may be answered as follows:

1. The theoretical analysis in Chapter 4 confirmed that principled reward shaping techniques will not alter agents' originally intended goals in MORL domains. Specifically, $PBRS$ does not alter the Pareto optimal policies in MOMDPs and MOSGs, and D preserves the Pareto relation between actions in MOSGs.

2. The empirical studies in Chapter 5 demonstrated that both $PBRS$ and D can improve converged performance and/or learning speed of agents in MORL domains, and can lead to solutions which Pareto dominate those produced by unshaped agents.

3. The Multi-Objective Beach Problem Domain was introduced in Section 5.2; this is the first example of a MOSG where the true Pareto optimal system utilities are known.

6.1 Summary of Contributions

In summary, the main contributions of this thesis are:

6.1.1 Theoretical Analysis of Reward Shaping for MORL

Chapter 4 presented the first reported theoretical evaluation of the effects of applying reward shaping in MORL, considering both single-agent and multi-agent domains, using the frameworks of MOMDPs and MOSGs respectively. This theoretical analysis was inspired by RQ1 listed above.

My investigations demonstrated that the use of $PBRS$ does not alter the Pareto relation between an agent's policies in MORL domains. These guarantees provide theoretical justification for the use of $PBRS$ in MORL applications, proving that the NDS of a domain remains consistent when a potential-based shaping function is used.

In the case of D, my analysis generalised results provided by Colby & Tumer (2015), demonstrating that the order of expected returns for actions in a MOSG is invariant when an agent is rewarded using a difference evaluation, rather than the system evaluation function. Therefore, actions which would lead to a Nash equilibrium reward with respect to D will also lead to a Nash equilibrium reward with respect to G, and actions which are Pareto optimal with respect to D are also Pareto optimal with respect to G.

6.1.2 Empirical Evaluations of Reward Shaping in MORL domains

Prior to this thesis, very little was understood about the effects of reward shaping in MORL domains. Some experimental evaluations of D in MOMARL domains were conducted previously by Yliniemi & Tumer (2016), whereas there were no reported applications of $PBRS$ to MORL. Chapter 5 contributed three separate empirical studies which evaluated the effects of both D and $PBRS$ across a range of domains, both single- and multi-agent.

In Section 5.1, $sPBRS$ was applied to a widely-used single-agent MORL benchmark, the Deep Sea Treasure domain. Analogously to the case in traditional MDPs, $sPBRS$ was found to have a significant effect on the learning rate of agents receiving shaping; good heuristics were found to improve performance, whereas misleading heuristics were found to damage performance. Significantly, in all cases the agents receiving $sPBRS$ learned the exact same policies as agents that did not receive shaping, regardless of the quality of the heuristic information that was used.

Section 5.2 introduced the Multi-Objective Beach Problem Domain, the first MOSG in the literature where the true Pareto optimal system utilities are known. Both D and $sPBRS$ were evaluated in this domain, and the results showed for the first time that agents learning using either technique can sample true Pareto optimal solutions in MOMARL domains.

Finally, Section 5.3 tested both D and $sPBRS$ in a complex real world application, the Dynamic Economic Emissions Dispatch domain. The traditional DEED problem was reformulated as a MOSG, and the experimental results demonstrated that MOMARL can produce solutions which are comparable to those found by other state-of-the-art optimisation algorithms.

The studies in Chapter 5 provide empirical support for the theory which addresses RQ1, while also answering RQ2 by confirming that both D and $PBRS$ can improve performance in MORL domains. As the first MOSG in the literature where the true NDS is known, the MOBPD addresses RQ3; consequently, MOMARL algorithms can now be evaluated not only in relative terms, but also with respect to a known absolute maximum level of performance.

6.1.3 New Insights into the Design of Potential Functions

Through the empirical studies presented in Chapters 3 and 5, new insights into the process of designing potential functions have emerged. These studies add to the existing body of work that justifies the use of $PBRS$ to improve the performance of RL agents.

Section 3.1 demonstrated how $aPBRS$ can be used to improve performance in a real world application, by applying a simple longest queue first heuristic to a range of Traffic Signal Control problems.

Section 3.2 addressed the question of how to design useful potential functions for cooperative MAS, proposing Estimated Counterfactual as Potential, a semi-automated method of generating multi-agent potential functions. Empirical results demonstrate that this method can match the performance of the analytically computed CaP while requiring much less information.

Section 3.3 investigated the effect of encoding the same heuristic knowledge in $sPBRS$ and $aPBRS$ formats for the first time, finding that the same heuristic can have vastly different effects on the behaviour of agents in a MAS depending on which method is used to encode it. Furthermore, this study also demonstrated that the beneficial effects of $aPBRS$ with a suitable heuristic do in fact scale up to large MAS with up to 100 agents.

Section 5.2 showed that the characteristic effects of $sPBRS$ in MARL domains also apply

in MOMARL scenarios. Specifically, applying $sPBRS$ in MOSGs can alter the learning rate and lead agents in the system to different points of equilibrium, analagously to its effects in SGs.

Finally, Section 5.3 applied $sPBRS$ to a real world problem domain, featuring multiple agents and multiple system objectives. This study again demonstrated how simple heuristics can lead to significant improvements in system performance in MOMARL applications.

6.2 Impact

Following from the contributions outlined above, this thesis will influence future research in a number of significant ways. Firstly, following from the theory in Chapter 4, two commonly used reward shaping techniques can now safely be applied to new classes of problems (MOMDPs and MOSGs), without fear of altering the agents' originally specified goals, as defined by the system evaluation function. This theoretical work is supported by three separate empirical studies across a range of problem domains, both single- and multi-agent, which demonstrate the beneficial effects of principled reward shaping techniques for MORL.

Now that a multi-objective multi-agent benchmark domain with a known NDS has been established (the MOBPD), MOMARL approaches may be evaluated against a known maximum level of system performance, rather than in relative terms. This development will be of great assistance to future researchers who wish to evaluate new algorithms for multi-objective MAS.

Finally, the empirical work in Chapters 3 and 5 has resulted in many new insights into how useful potential functions may be designed for a range of different problem domains. These insights will inform future research on $PBRS$, and will make this method more accessible to all.

6.3 Limitations

Despite the contributions outlined above, this work does have limitations, as outlined below:

6.3.1 Discrete States and Actions

All of the experimental work and theoretical analysis in this thesis assumes that application domains have a finite set of states and actions which are represented discretely. This was a useful assumption for the purposes of this investigation, as it permitted proofs to be developed which support the use of reward shaping in MORL, as well as facilitating empirical validation of the guarantees provided.

However, representing states and actions discretely becomes problematic for larger problems due to the state-action space explosion, as discussed in Section 2.2. Furthermore, many complex real world problem domains feature continuous states and/or actions, therefore limiting the applicability of the techniques considered here unless they are used in conjunction with some form of Q function approximation.

6.3.2 Deterministic Environments

Another limitation of this work is the use of fully deterministic environments (apart from the TSC scenario in Section 3.1), specifically the assumptions that environment state transitions are consistent, and that the rewards for such transitions do not contain any noise. In multi-agent domains, from an individual agent's perspective the environment is stochastic due the the effect of other agents, but determinism may still exist from a system perspective. As with the use of discrete environments, the assumption of determinism was convenient when developing theoretical and empirical results.

Furthermore, it is assumed in this work that actions selected by agents are always successful. This assumption would not hold in many real-world applications (e.g. robotics), as an agent's actuators cannot be fully guaranteed to be immune from failure. Removing the assumption of determinism would present an increased challenge for agents, and I expect that the advantage enjoyed by agents receiving domain knowledge over unshaped agents would be even greater in non-deterministic environments.

6.3.3 Fully Cooperative Domains

The experimental work on multi-agent domains in this thesis considered fully cooperative scenarios only. A core theme of this work is the comparison of D and $PBRS$; D is only applicable to cooperative MAS, and therefore the experiments were limited to fully cooperative domains. However, cooperative domains represent only a subset of all possible multi-objective MAS applications where reward shaping may be useful.

6.3.4 Bi-objective Domains

Finally, all the MORL experiments in this thesis consider systems that have just two objectives, although the theoretical results from Chapter 4 still hold regardless of the number of system objectives present. In practice, real world systems may have many more than just two objectives which must be considered; therefore further evaluations of reward shaping techniques should be conducted using more complex and realistic problem domains.

6.4 Future Work

Following from the investigations in this thesis, there are a number of promising avenues for future research:

6.4.1 Design of Potential Functions

This work has shown that $PBRS$ can improve performance in MOSGs, even when very basic heuristic knowledge is used. The question of how to design useful multi-agent potential functions is an active area of research, and has not been explored comprehensively in a multi-objective context to date.

I expect that certain types of $PBRS$ heuristics could lead agents to discover policies that favour one objective over another. Therefore, in future it may be possible to use $PBRS$ as a mechanism to incorporate user preferences in multi-criteria sequential decision making problems, by designing potential functions that bias an agent's exploration appropriately.

Furthermore, this thesis did not evaluate $aPBRS$ in any MORL domains; given the considerable improvements exhibited when a good $aPBRS$ heuristic is applied in MDPs (Section 3.1) and SGs (Section 3.3), $aPBRS$ is a promising candidate to increase performance in MORL domains in the future.

As suggested by Devlin et al. (2014), the properties of D and $PBRS$ can be combined to leverage the benefits of both techniques. Section 3.2 expanded upon this idea, addressing one of the main limitations of CaP; the need to know the precise form of the system evaluation function and for complete information about the system state and joint action.

This theme could be explored further, by using estimates of the difference evaluation for particular state-action pairs as a potential function with $aPBRS$. Estimated Difference as Potential ($\hat{D}aP$) could incorporate knowledge based on difference rewards while benefiting from the theoretical guarantees of $PBRS$, without suffering from the limitations of D (onerous information requirements) or $\hat{D}$ (lack of theoretical guarantees).

6.4.2 Alternate Credit Assignment Structures for MOMARL

While two commonly used MARL credit assignment techniques were considered in this thesis, numerous other promising methods exist. Recently proposed techniques include: $DRiP$ (Devlin et al. 2014), Resource Abstraction (Malialis et al. 2016), CLEAN Rewards (HolmesParker et al. 2013; Colby et al. 2015), $D++$ (Rahmattalabi et al. 2016), and $\hat{D}$ (Colby et al. 2016). Investigating the theoretical properties and empirical performance of these new reward structures in a MOMARL context is promising direction for future research.

6.4.3 Further MOMARL Benchmark Domains

This thesis introduced the MOBPD, the first MOMARL benchmark domain where the true Pareto optimal solutions are known. Benchmarks of this kind are important, as they allow the performance of a new MORL algorithm to be evaluated with respect to a known absolute maximum level of performance, i.e. the hypervolume of the true Pareto front. Vamplew et al. (2010) proposed a set of benchmark problems for single-agent MORL algorithms, which have proved extremely useful when researchers wish to evaluate different MORL approaches. A similar set of benchmark domains for MOMARL should also be established, where the domains are modelled as MOSGs. A consistent set of benchmarks would be extremely useful to the emerging MOMARL community.

The MOBPD is a good first step towards achieving this goal; however, it does have some shortcomings. Specifically, agents are arranged in a simple 1D environment, where the dynamics

are fully deterministic from a system point of view. Furthermore, the Pareto front of this domain is globally convex. Ideally, MOMARL benchmark domains should be developed which cover all possible NDS shapes: linear, convex and concave, with both discrete and continuous sets of solutions. However, designing the system reward functions so that they satisfy these criteria is more difficult than for single-agent MORL, as the Pareto optimal solutions for a MOSG are derived using joint policies instead of the policy of a single agent. Section 3.3.5 proposed a multi-objective version of the SPD; this extension of the current problem format could potentially satisfy some or all of the criteria above.

6.4.4 Scalarisation Functions for MOMARL

To the best of my knowledge, only linear and hypervolume scalarisation functions have been used with MOMARL to date; these functions are quite basic and may not allow all solutions along the Pareto front to be learned successfully. Therefore, more advanced scalarisation functions such as a weighted hypervolume, Chebyshev scalarisation (Van Moffaert et al. 2013b) or Thresholded Lexicographic Ordering (Gábor et al. 1998; Vamplew et al. 2010) could be used in conjunction with MARL algorithms in future to improve coverage along the Pareto front.

6.4.5 Multi-Policy Algorithms

Recent work in single-agent MORL has led to the development of multi-policy algorithms such as Pareto Q-learning (Van Moffaert & Nowé 2014), which can track multiple non-dominated policies at once; developing such algorithms in a MARL context may prove to be a fruitful direction for future work.

Both $PBRS$ and D were originally designed for use with single-policy algorithms, and to the best of my knowledge reward shaping (or any other KBRL method) has not been tested with multi-policy algorithms to date. Integrating domain knowledge into multi-policy algorithms would reduce the time needed to learn all solutions that form part of the NDS.

6.4.6 Alternate Methods for Knowledge-Based MORL

Although many Knowledge-Based Reinforcement Learning techniques exist, this thesis focused solely on reward shaping. Many of the other techniques surveyed in Section 2.5 maintain policy invariance in MDPs, and could easily be adapted for MORL applications. HARL and PPR could be applied in cases where domain knowledge about Pareto optimal actions is available, as is the case with $aPBRS$. IPM on the other hand would be useful in MORL applications where the environment is stochastic, provided that the system designer has some level of knowledge about the environment dynamics.

Finally, TL is also a promising candidate to improve MORL performance. TL could be applied to MORL by first training an agent on a simplified single-objective version of a domain, then using the knowledge gained to improve learning speed on a more complex version of the same domain with additional objectives.

6.4.7 Reward Shaping for Deep Reinforcement Learning

While this thesis considered domains with discrete state-action spaces only, reward shaping could equally be applied to improve learning speed and performance when Q function approximation is used. Unfortunately, applying Q function approximation breaks the theoretical guarantees provided by many RL algorithms. Whether the theoretical properties of D and $PBRS$ would still hold when Q function approximation is used is currently an open question.

Despite a lack of theoretical guarantees, Deep RL techniques have achieved considerable success in numerous complex problem domains (see e.g. Mnih et al. (2015); Silver et al. (2016)). However, these algorithms have onerous requirements in terms of the number of samples required for training. If reward shaping were applied to Deep RL, I expect that it would have similar effects to those demonstrated in this thesis, improving learning speed, as well as joint performance on system objectives in the case of cooperative MAS.

6.5 Final Remarks

MORL is an emerging research area that will continue to grow in importance, especially considering that many real world problems exhibit conflicting objectives which must be optimised. The issue of appropriate credit assignment will become increasingly prominent as MAS research naturally progresses towards more precise models which explicitly account for multiple system objectives, and reward shaping offers a mechanism to address the credit assignment problem. I hope that by now the case for applying principled reward shaping techniques to improve performance in MORL has been made convincingly, and that this thesis will inspire future work on the design of reward functions.

References

Abdoos, M., Mozayani, N., & Bazzan, A. L. (2014). Hierarchical control of traffic signals using q-learning with tile coding. *Applied Intelligence*, *40*(2), 201–213.

Abdulhai, B., Pringle, R., & Karakoulas, G. (2003). Reinforcement learning for true adaptive traffic signal control. *Journal of Transportation Engineering*, *129*(3), 278–285.

Agogino, A. K. & Tumer, K. (2008). Analyzing and visualizing multiagent rewards in dynamic and stochastic environments. *Autonomous Agents and Multi-Agent Systems*, *17*(2), 320–338.

Arthur, W. B. (1994). Inductive reasoning and bounded rationality. *The American Economic Review*, *84*(2), 406–411.

Asmuth, J., Littman, M. L., & Zinkov, R. (2008). Potential-based shaping in model-based reinforcement learning. In *AAAI*, (pp. 604–609).

Babes, M., Cote, E. M. D., & Littman, M. L. (2008). Social reward shaping in the prisoner's dilemma. In *Proceedings of the 7th International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, (pp. 1389–1392).

Barrett, L. & Narayanan, S. (2008). Learning all optimal policies with multiple criteria. In *Proceedings of the 25th international conference on Machine learning*, (pp. 41–47). ACM.

Basu, M. (2008). Dynamic economic emission dispatch using nondominated sorting genetic algorithm-ii. *International Journal of Electrical Power & Energy Systems*, *30*(2), 140–149.

Bazzan, A. L. C. (2009). Opportunities for multiagent systems and multiagent reinforcement learning in traffic control. *Autonomous Agents and Multi-Agent Systems*, *18*(3), 342–375.

Bazzan, A. L. C. & Klugl, F. (2014). A review on agent-based technology for traffic and transportation. *The Knowledge Engineering Review*, *29*, 375–403.

Bellman, R. (1957). *Dynamic Programming*. Princeton, NJ, USA: Princeton University Press.

Bianchi, R. A. C., Martins, M. F., Ribeiro, C. H. C., & Costa, A. H. R. (2014). Heuristically-accelerated multiagent reinforcement learning. *IEEE Transactions on Cybernetics*, *44*(2), 252–

265.

Bianchi, R. A. C., Ribeiro, C. H. C., & Costa, A. H. R. (2004). Heuristically accelerated q–learning: A new approach to speed up reinforcement learning. In Bazzan, A. L. C. & Labidi, S. (Eds.), *Advances in Artificial Intelligence – SBIA 2004: 17th Brazilian Symposium on Artificial Intelligence, Sao Luis, Maranhao, Brazil, September 29-Ocotber 1, 2004. Proceedings*, (pp. 245–254)., Berlin, Heidelberg. Springer Berlin Heidelberg.

Bianchi, R. A. C., Ribeiro, C. H. C., & Costa, A. H. R. (2007). Heuristic selection of actions in multiagent reinforcement learning. In *Proceedings of the 20th International Joint Conference on Artifical Intelligence*, IJCAI'07, (pp. 690–696)., San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

Bianchi, R. A. C., Ribeiro, C. H. C., & Costa, A. H. R. (2012). Heuristically accelerated reinforcement learning: Theoretical and experimental results. In *Proceedings of the 20th European Conference on Artificial Intelligence*, ECAI'12, (pp. 169–174).

Brafman, R. I. & Tennenholtz, M. (2003). R-max - a general polynomial time algorithm for near-optimal reinforcement learning. *J. Mach. Learn. Res.*, *3*, 213–231.

Brys, T. (2016). *Reinforcement Learning with Heuristic Information*. PhD thesis, Vrije Universitet Brussel.

Brys, T., Harutyunyan, A., Taylor, M. E., & Nowé, A. (2015). Policy transfer using reward shaping. In *Proceedings of the Fourteenth International Conference on Autonomous Agents and Multi-Agent Systems (AAMAS)*.

Brys, T., Harutyunyan, A., Vrancx, P., NowÃ©, A., & Taylor, M. E. (2017). Multi-objectivization and ensembles of shapings in reinforcement learning. *Neurocomputing*, *263*, 48 – 59.

Brys, T., Harutyunyan, A., Vrancx, P., Taylor, M. E., Kudenko, D., & Nowé, A. (2014). Multi-objectivization of reinforcement learning problems by reward shaping. In *Neural Networks (IJCNN), 2014 International Joint Conference on*, (pp. 2315–2322). IEEE.

Brys, T., Pham, T. T., & Taylor, M. E. (2014). Distributed learning and multi-objectivity in traffic light control. *Connection Science*, *26*(1), 65–83.

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

Claus, C. & Boutilier, C. (1998). The dynamics of reinforcement learning in cooperative multiagent systems. In *Proceedings of the Fifteenth National/Tenth Conference on Artificial Intelligence/Innovative Applications of Artificial Intelligence*, AAAI '98/IAAI '98, (pp. 746–752).

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

Colby, M. & Tumer, K. (2015). An evolutionary game theoretic analysis of difference evalu-

ation functions. In *Proceedings of the 2015 Annual Conference on Genetic and Evolutionary Computation*, (pp. 1391–1398). ACM.

Colby, M. K., Kharaghani, S., HolmesParker, C., & Tumer, K. (2015). Counterfactual exploration for improving multiagent learning. In *Proceedings of the 2015 International Conference on Autonomous Agents and Multiagent Systems*, (pp. 171–179).

Devlin, S. (2013). *Potential-Based Reward Shaping for Knowledge-Based, Multi-Agent Reinforcement Learning*. PhD thesis, University of York, UK.

Devlin, S., Grzes, M., & Kudenko, D. (2010). Multi-agent reinforcement learning with reward shaping for keepaway takers. In *Proceedings of the Adaptive and Learning Agents workshop (at AAMAS 2010)*.

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

Devlin, S., Grzes, M., & Kudenko, D. (2011b). 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)*, (pp. 1227–1228).

Devlin, S. & Kudenko, D. (2011). 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)*, (pp. 225–232).

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

Devlin, S. & Kudenko, D. (2016). Plan-based reward shaping for multi-agent reinforcement learning. *The Knowledge Engineering Review*, *31*, 44–58.

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

Dresner, K. & Stone, P. (2006). Multiagent traffic management: Opportunities for multiagent learning. In K. Tuyls, P. Hoen, K. Verbeeck, & S. Sen (Eds.), *Learning and Adaption in Multi-Agent Systems*, volume 3898 of *Lecture Notes in Computer Science* (pp. 129–138). Springer Berlin Heidelberg.

Duggan, J. (2008). Using system dynamics and multiple objective optimization to support policy analysis for complex systems. In H. Qudrat-Ullah, J. Spector, & P. Davidsen (Eds.), *Complex Decision Making: Theory and Practice* (pp. 59–81). Springer Berlin Heidelberg.

Eck, A., Soh, L.-K., Devlin, S., & Kudenko, D. (2013). Potential-based reward shaping for pomdps (extended abstract). In *Proceedings of the Twelfth International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, (pp. 1123–1124).

Eck, A., Soh, L.-K., Devlin, S., & Kudenko, D. (2016). Potential-based reward shaping for

finite horizon online pomdp planning. *Autonomous Agents and Multi-Agent Systems*, *30*(3), 403–445.

Efthymiadis, K., Devlin, S., & Kudenko, D. (2016). Overcoming incorrect knowledge in plan-based reward shaping. *The Knowledge Engineering Review*, *31*, 31–43.

Efthymiadis, K. & Kudenko, D. (2014). A comparison of plan-based and abstract mdp reward shaping. *Connection Science*, *26*(1), 85–99.

Efthymiadis, K. & Kudenko, D. (2015). Knowledge revision for reinforcement learning with abstract mdps. In *Proceedings of the Fourteenth International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, (pp. 763–770).

El-Tantawy, S. & Abdulhai, B. (2010). An agent-based learning towards decentralized and coordinated traffic signal control. In *Intelligent Transportation Systems (ITSC), 2010 13th International IEEE Conference on*, (pp. 665–670).

El-Tantawy, S. & Abdulhai, B. (2012). Multi-agent reinforcement learning for integrated network of adaptive traffic signal controllers (marlin-atsc). In *Intelligent Transportation Systems (ITSC), 2012 15th International IEEE Conference on*, (pp. 319–326).

El-Tantawy, S., Abdulhai, B., & Abdelgawad, H. (2013). Multiagent reinforcement learning for integrated network of adaptive traffic signal controllers (marlin-atsc): Methodology and large-scale application on downtown toronto. *Intelligent Transportation Systems, IEEE Transactions on*, *14*(3), 1140–1150.

Fernández, F. & Veloso, M. (2006). Probabilistic policy reuse in a reinforcement learning agent. In *Proceedings of the fifth international joint conference on Autonomous agents and multiagent systems*, (pp. 720–727). ACM.

Gábor, Z., Kalmár, Z., & Szepesvári, C. (1998). Multi-criteria reinforcement learning. In *Proceedings of the Fifteenth International Conference on Machine Learning*, (pp. 197–205).

Grześ, M. (2017). Reward shaping in episodic reinforcement learning. In *Proceedings of the 16th International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, (pp. 565–573).

Grześ, M. & Kudenko, D. (2008). Plan-based reward shaping for reinforcement learning. In *Intelligent Systems, 4th International IEEE Conference on*.

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

Grześ, M. & Kudenko, D. (2010). Online learning of shaping rewards in reinforcement learning. *Neural Networks*, *23*(4), 541–550.

Harutyunyan, A., Brys, T., Vrancx, P., & Nowé, A. (2015). Shaping mario with human advice (demonstration). In *Proceedings of the Fourteenth International Conference on Autonomous Agents and Multi-Agent Systems (AAMAS)*.

Harutyunyan, A., Devlin, S., Vrancx, P., & Nowé, A. (2015). Expressing arbitrary reward func-

tions as potential-based advice. In *Twenty-Ninth Conference on Artificial Intelligence (AAAI)*.

HolmesParker, C., Agogino, A., & Tumer, K. (2013). Clean rewards for improving multiagent coordination in the presence of exploration. In *Proceedings of the 2013 International Conference on Autonomous Agents and Multi-agent Systems*, (pp. 1113–1114).

Kapetanakis, S. & Kudenko, D. (2002). Reinforcement learning of coordination in cooperative multi-agent systems. In *Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence*.

Khamis, M. A. & Gomaa, W. (2014). Adaptive multi-objective reinforcement learning with hybrid exploration for traffic signal control based on cooperative multi-agent framework. *Eng. Appl. Artif. Intell.*, *29*, 134–151.

Kim, H., Lim, W., Lee, K., Noh, Y.-K., & Kim, K.-E. (2015). Reward shaping for model-based bayesian reinforcement learning. In *Twenty-Ninth AAAI Conference on Artificial Intelligence*.

Knowles, J. D., Watson, R. A., & Corne, D. W. (2001). Reducing local optima in single-objective problems by multi-objectivization. In *Evolutionary Multi-Criterion Optimization: First International Conference, EMO 2001 Zurich, Switzerland, March 7–9, 2001 Proceedings*, (pp. 269–283).

Koenig, S. & Simmons, R. G. (1996). The effect of representation and knowledge on goal-directed exploration with reinforcement-learning algorithms. *Machine Learning*, *22*(1-3), 227–250.

Krajzewicz, D., Erdmann, J., Behrisch, M., & Bieker, L. (2012). Recent development and applications of SUMO - Simulation of Urban MObility. *International Journal On Advances in Systems and Measurements*, *5*(3&4), 128–138.

Laud, A. D. (2004). *Theory and application of reward shaping in reinforcement learning*. PhD thesis, University of Illinois at Urbana-Champaign, Urbana, Illinois, USA.

Lu, X., Schwartz, H. M., & Givigi, S. N. (2011). Policy invariance under reward transformations for general-sum stochastic games. *Journal of Artificial Intelligence Research*, *41*(2), 397–406.

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

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

Mannion, P., Duggan, J., & Howley, E. (2015a). Parallel reinforcement learning for traffic signal control. In *Proceedings of the 4th International Workshop on Agent-based Mobility, Traffic and Transportation Models, Methodologies and Applications (ABMTRANS 2015)*.

Mannion, P., Duggan, J., & Howley, E. (2015b). Parallel reinforcement learning with state action space partitioning. In *12th European Workshop on Reinforcement Learning (at ICML 2015)*.

Marler, R. T. & Arora, J. S. (2004). Survey of multi-objective optimization methods for engineering. *Structural and multidisciplinary optimization*, *26*(6), 369–395.

Marthi, B. (2007). Automatic shaping and decomposition of reward functions. In *Proceedings of the 24th International Conference on Machine learning*, (pp. 601–608). ACM.

Mason, K. (2015). Avoidance techniques & neighbourhood topologies in particle swarm optimisation. Master's thesis, National University of Ireland Galway.

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

Mitchell, T. M. (1997). *Machine learning*. McGraw-Hill series in computer science. Boston (Mass.), Burr Ridge (Ill.), Dubuque (Iowa): McGraw-Hill.

Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., Graves, A., Riedmiller, M., Fidjeland, A. K., Ostrovski, G., et al. (2015). Human-level control through deep reinforcement learning. *Nature*, *518*(7540), 529–533.

Nash, J. (1951). Non-cooperative games. *Annals of Mathematics*, *54*(2), 286–295.

Ng, A. Y., Harada, D., & Russell, S. J. (1999). Policy invariance under reward transformations: Theory and application to reward shaping. In *Proceedings of the Sixteenth International Conference on Machine Learning*, ICML '99, (pp. 278–287).

Pareto, V. (1971). *Manual of political economy*. Macmillan.

Pham, T., Brys, T., & Taylor, M. E. (2013). Learning coordinated traffic light control. In *Proceedings of the Adaptive and Learning Agents workshop (at AAMAS 2013)*.

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

Rahmattalabi, A., Chung, J. J., & Tumer, K. (2016). D++: Structural credit assignment in tightly coupled multiagent domains. In *Proceedings of the workshop on On-line decision-making in multi-robot coordination (at RSS 2016)*.

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

Roijers, D. M., Vamplew, P., Whiteson, S., & Dazeley, R. (2013). A survey of multi-objective sequential decision-making. *Journal of Artificial Intelligence Research*, *48*, 67–113.

Roijers, D. M., Whiteson, S., & Oliehoek, F. A. (2013). Computing convex coverage sets for multi-objective coordination graphs. In *International Conference on Algorithmic Decision Theory*, (pp. 309–323).

Roijers, D. M., Whiteson, S., & Oliehoek, F. A. (2014). Linear support for multi-objective coordination graphs. In *Proceedings of the 2014 international conference on Autonomous agents and multi-agent systems*, (pp. 1297–1304). International Foundation for Autonomous Agents and Multiagent Systems.

Roijers, D. M., Whiteson, S., & Oliehoek, F. A. (2015). Computing convex coverage sets for faster multi-objective coordination. *Journal of Artificial Intelligence Research*, *52*, 399–443.

Russell, S. J. & Norvig, P. (2009). Artificial intelligence: a modern approach (3rd edition).

Shelton, C. R. (2001). *Importance Sampling for Reinforcement Learning with Multiple Objectives*. PhD thesis, Massachusetts Institute of Technology, Cambridge, MA, USA.

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

Silver, D., Huang, A., Maddison, C. J., Guez, A., Sifre, L., Van Den Driessche, G., Schrittwieser, J., Antonoglou, I., Panneershelvam, V., Lanctot, M., et al. (2016). Mastering the game of go with deep neural networks and tree search. *Nature*, *529*(7587), 484–489.

Skinner, B. F. (1938). *The behavior of organisms: An experimental analysis.* Appleton-Century.

Smith, A. E., Coit, D. W., Baeck, T., Fogel, D., & Michalewicz, Z. (2000). Penalty functions. *Evolutionary computation*, *2*, 41–48.

Sutton, R. S. & Barto, A. G. (1998). *Introduction to Reinforcement Learning* (1st ed.). Cambridge, MA, USA: MIT Press.

Tamar, A., Di Castro, D., & Meir, R. (2012). Integrating a partial model into model free reinforcement learning. *J. Mach. Learn. Res.*, *13*(1), 1927–1966.

Taylor, A., Dusparic, I., Galván-López, E., Clarke, S., & Cahill, V. (2014). Accelerating learning in multi-objective systems through transfer learning. In *Neural Networks (IJCNN), 2014 International Joint Conference on*, (pp. 2298–2305). IEEE.

Taylor, M. E. & Stone, P. (2007). Cross-domain transfer for reinforcement learning. In *Proceedings of the 24th international conference on Machine learning*, (pp. 879–886). ACM.

Taylor, M. E. & Stone, P. (2009). Transfer learning for reinforcement learning domains: A survey. *Journal of Machine Learning Research*, *10*(Jul), 1633–1685.

Tesauro, G. (1994). Td-gammon, a self-teaching backgammon program, achieves master-level play. *Neural Computing*, *6*(2), 215–219.

Thorpe, T. L. & Anderson, C. W. (1996). Traffic light control using sarsa with three state representations. Technical report, IBM Corporation.

Torrey, L. & Shavlik, J. (2010). Transfer learning. In *Handbook of Research on Machine Learning Applications and Trends: Algorithms, Methods, and Techniques* (pp. 242–264). IGI Global.

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

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

Vamplew, P., Dazeley, R., Berry, A., Issabekov, R., & Dekker, E. (2010). Empirical evaluation methods for multiobjective reinforcement learning algorithms. *Machine Learning*, *84*(1), 51–80.

Vamplew, P., Yearwood, J., Dazeley, R., & Berry, A. (2008). On the limitations of scalarisation

for multi-objective reinforcement learning of pareto fronts. In Wobcke, W. & Zhang, M. (Eds.), *AI 2008: Advances in Artificial Intelligence: 21st Australasian Joint Conference on Artificial Intelligence*, (pp. 372–378)., Berlin, Heidelberg. Springer Berlin Heidelberg.

van der Pol, E. & Oliehoek, F. A. (2016). Coordinated deep reinforcement learners for traffic light control. In *Proceedings of Learning, Inference and Control of Multi-Agent Systems (at NIPS 2016)*.

Van Moffaert, K., Brys, T., Chandra, A., Esterle, L., Lewis, P. R., & Nowé, A. (2014). A novel adaptive weight selection algorithm for multi-objective multi-agent reinforcement learning. In *Neural Networks (IJCNN), 2014 International Joint Conference on*, (pp. 2306–2314).

Van Moffaert, K., Drugan, M. M., & Nowé, A. (2013a). Hypervolume-based multi-objective reinforcement learning. In Purshouse, R. C., Fleming, P. J., Fonseca, C. M., Greco, S., & Shaw, J. (Eds.), *Evolutionary Multi-Criterion Optimization: 7th International Conference, EMO 2013, Sheffield, UK, March 19-22, 2013. Proceedings*, (pp. 352–366)., Berlin, Heidelberg. Springer Berlin Heidelberg.

Van Moffaert, K., Drugan, M. M., & Nowé, A. (2013b). Scalarized multi-objective reinforcement learning: Novel design techniques. In *2013 IEEE Symposium on Adaptive Dynamic Programming and Reinforcement Learning (ADPRL)*, (pp. 191–199). IEEE.

Van Moffaert, K. & Nowé, A. (2014). Multi-objective reinforcement learning using sets of pareto dominating policies. *The Journal of Machine Learning Research*, *15*(1), 3483–3512.

Walters, D. C. & Sheble, G. B. (1993). Genetic algorithm solution of economic dispatch with valve point loading. *Power Systems, IEEE Transactions on*, *8*(3), 1325–1332.

Wang, W. & Sebag, M. (2013). Hypervolume indicator and dominance reward based multi-objective monte-carlo tree search. *Machine Learning*, *92*(2), 403–429.

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

Wiering, M. (2000). Multi-agent reinforcement learning for traffic light control. In *Proceedings of the Seventeenth International Conference on Machine Learning*, ICML '00, (pp. 1151–1158)., San Francisco. Morgan Kaufmann Publishers Inc.

Wiering, M. & van Otterlo, M. (Eds.). (2012). *Reinforcement Learning: State-of-the-Art*. Springer.

Wiewiora, E. (2003). Potential-based shaping and q-value initialization are equivalent. *Journal of Artificial Intelligence Research*, *19*(1), 205–208.

Wiewiora, E. (2017). Reward shaping. In C. Sammut & G. I. Webb (Eds.), *Encyclopedia of Machine Learning and Data Mining* (pp. 1104–1106). Boston, MA: Springer US.

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

Wolpert, D. H. & Tumer, K. (2002). Collective intelligence, data routing and braess' paradox.

Journal of Artificial Intelligence Research, *16*, 359–387.

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

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

Yliniemi, L. & Tumer, K. (2016). Multi-objective multiagent credit assignment in reinforcement learning and nsga-ii. *Soft Computing*, *20*(10), 3869–3887.

Yliniemi, L. M. (2015). *Multi-Objective Optimization in Multiagent Systems*. PhD thesis, Oregon State University, Corvallis, OR.