Potential-Based Reward Shaping for Knowledge-Based, Multi-Agent Reinforcement Learning

SAM DEVLIN

Submitted for the degree of Doctor of Philosophy.

DEPARTMENT OF COMPUTER SCIENCE
THE UNIVERSITY OF YORK

July 2013

Dedicated to my Dad,
who sadly will not get to see it completed
but who gave so much to make sure it could be.

Abstract

Reinforcement learning is a robust artificial intelligence solution for agents required to act in an environment, making their own decisions on how to behave. Typically an agent is deployed alone with no prior knowledge, but if given sufficient time, a suitable state representation and an informative reward function is guaranteed to learn how to maximise its long term reward.

Incorporating domain knowledge, typically known by the system designer, can minimise the number of suboptimal behaviours tried and, therefore, speed up the rate of learning. Potential-based reward shaping is a method of providing this knowledge to an agent by additional rewards. Furthermore, if the agent is alone in the environment, it is guaranteed to learn the same behaviour both with and without potential-based reward shaping.

Meanwhile, there has also been a growing interest in deploying not just one agent but many into the same environment. This application can benefit from the potential of both multi-agent systems and reinforcement learning. However, practical use is often limited by the non-stationary environment, exponential increase in state features with every agent added and partial observability.

This thesis documents work combining knowledge-based reinforcement learning and multi-agent reinforcement learning so that the latter can be achieved quicker and, therefore, feasibly applied to complex problem domains.

Experience gained from many empirical studies is gathered to support novel theoretical contributions proving that the pre-existing guarantees of potential-based reward shaping do not apply when used in multi-agent problem domains. Instead multi-agent potential-based reward shaping may cause agents to learn a different behaviour, but this behaviour is guaranteed to be from the same set of behaviours that the agents could have learned without the additional rewards. Therefore, knowledge-based multi-agent reinforcement learning can both reduce the time a group of agents need to learn a suitable behaviour and increase their final performance.

Contents

List of Figures . . . 7
List of Tables . . . 7
Acknowledgements . . . 8
Declaration . . . 9

1 Introduction and Motivation **11**
1.1 Hypothesis . . . 12
1.2 Scope . . . 12
1.3 Thesis Overview . . . 13

2 Background and Literature Review **14**
2.1 Reinforcement Learning . . . 14
2.1.1 Markov Decision Processes . . . 16
2.1.2 Algorithms . . . 17
2.1.3 Eligibility Traces . . . 19
2.1.4 Function Approximation . . . 20
2.2 Multi-Agent Systems . . . 21
2.3 Multi-Agent Reinforcement Learning . . . 22
2.3.1 Stochastic Games . . . 23
2.3.2 Nash Equilibrium . . . 24
2.3.3 Pareto Optimality . . . 24
2.3.4 Algorithms . . . 25
2.4 Knowledge-Based Reinforcement Learning . . . 27
2.4.1 Reward Shaping . . . 27
2.4.2 Multi-Agent Reward Shaping . . . 31
2.4.3 Alternative Methods . . . 33
2.4.4 Summary . . . 37

3 Multi-Agent, Potential-Based Reward Shaping: Empirical Studies **38**
3.1 Plausibility Study . . . 38
3.2 RoboCup Soccer Study . . . 42
3.2.1 Multi-Agent Learning in RoboCup Soccer . . . 43
3.2.2 KeepAway . . . 44
3.2.3 TakeAway . . . 48
3.3 Conclusion . . . 58

4 Multi-Agent, Potential-Based Reward Shaping: In Theory **60**
4.1 Equivalence to Q-Table Initialisation . . . 61
4.2 Consistent Nash Equilibria . . . 63
4.3 Convergence Guarantees . . . 65
4.4 Dynamic Potential Functions . . . 66
4.4.1 Policy Invariance and Consistent Nash Equilibria . . . 67
4.4.2 Non-Equivalence To Q-Table Initialisation . . . 68
4.5 Empirical Demonstration . . . 71
4.5.1 Results . . . 72
4.6 Properties Invariant to Changes in Absolute Value . . . 74
4.7 Application to Other Algorithms . . . 75
4.8 Finite Potential-Based Reward Shaping . . . 76
4.9 Conclusion . . . 77

5 Designing Multi-Agent Potential Functions **79**
5.1 Multi-Agent Planning . . . 79
5.1.1 Centralised Planning for Decentralised Plans . . . 80
5.1.2 Decentralised Planning for Decentralised Plans . . . 80
5.1.3 Summary . . . 81
5.2 Multi-Agent, Plan-Based Reward Shaping . . . 81
5.3 Empirical Study . . . 82
5.3.1 Results . . . 85
5.4 Scaling Up . . . 86
5.4.1 Extra Flags . . . 87
5.4.2 Extra Agent . . . 88
5.5 Overcoming Conflicted Knowledge . . . 89
5.5.1 Increasing Exploration . . . 91
5.5.2 Improving Knowledge . . . 92
5.5.3 Scaling Up . . . 94
5.6 Conclusion . . . 96

6 Conclusion and Future Work **97**
6.1 Summary of Contributions . . . 98
6.2 Limitations . . . 99
6.3 Future Work . . . 100
6.4 Closing Remarks . . . 101

References **102**

Index **110**

List of Symbols and Acronyms **112**

List of Figures

2.1 Agent-Environment Interaction. 15
2.2 Example Function Approximation Techniques. 20
2.3 Typical Effect of PBRS on Single-Agent Reinforcement Learning. 28
2.4 Plan-Based Reward Shaping. 30
2.5 An Example of Braess Paradox. 32
2.6 An Example Hierarchical Reinforcement Learning Solution. 34
2.7 Coordination Guided Reinforcement Learning's Two Level Learning System. . . 35

3.1 Problem Domain for Plausibility Study 39
3.2 Multiple Independent Q-Learners 39
3.3 ASFQ-learning 40
3.4 Joint Action Q-Learners 41
3.5 Distributed Q-Learning 41
3.6 WoLFQ-PHC 42
3.7 Snapshot of a 3 vs. 2 KeepAway game. 43
3.8 The 25 Possible Locations of Keepers when Off-The-Ball and 5 Example Actions Given a Keeper at K. 45
3.9 3 Learning Keepers vs. 2 Hand-Coded Takers. 47
3.10 State Representation for Learning Takers 48
3.11 2 Learning Takers vs. 3 Hand-Coded Keepers. 52
3.12 2 Learning Takers vs. 3 Hand-Coded Keepers at 40x40. 53
3.13 2 Learning Takers vs. 3 Hand-Coded Keepers at 50x50. 55
3.14 3 Learning Takers vs. 4 Hand-Coded Keepers. 56
3.15 4 Learning Takers vs. 5 Hand-Coded Keepers. 57

4.1 Boutilier's Coordination Game 71
4.2 Boutilier's Coordination Game without Reward Shaping or with the Optimal Nash Equilibrium Encouraged 73
4.3 Boutilier's Coordination Game with Miscoordination Encouraged or the Safety Nash Equilibrium 74
4.4 Boutilier's Coordination Game with Uniform or Negative Bias, Random, Dynamic PBRS 74

5.1 Multi-Agent, Flag-Collecting Problem Domain 82

5.2 Initial Results . . . 86
5.3 Scaled Up Problem Domain . . . 87
5.4 Pessimistic Initialisation in the Scaled Up Problem Domain . . . 87
5.5 Scaled Up Problem Domain with 3 Agents . . . 88
5.6 Pessimistic Initialisation in the Scaled Up Problem Domain with 3 Agents . . . 88
5.7 Example Behaviour of Joint-Plan-Based Agents . . . 89
5.8 Example Behaviour of Individual-Plan-Based Agents . . . 90
5.9 Competitive Reward . . . 91
5.10 Optimistic Initialisation . . . 92
5.11 Optimistic Partial Plans . . . 93
5.12 Pessimistic Partial Plans . . . 94
5.13 Pessimistic Partial Plans in the Scaled Up Problem Domain . . . 95
5.14 Optimistic Initialisation in the Scaled Up Problem Domain . . . 95

6.1 Plan-Based Reward Shaping with Belief Revision . . . 100

List of Tables

3.1 State Representations for Learning Keepers . . . 45
3.2 Shaping Values of a Tackling Taker given $\gamma = 1$ . . . 49

Acknowledgements

I would like to begin with a large thank you to my supervisor Daniel Kudenko. Your advice, guidance and support throughout my PhD has been critical to my success. From encouraging me to do a PhD all the way through to helping me find a suitable and exciting post-doc, Daniel has been a great supervisor, mentor and friend.

I would also like to thank my examiners; Karl Tuyls, Enda Howley and James Cussens. I greatly appreciate the time you all put into reading, assessing and giving feedback on this thesis. Each of you has had a unique and significant impact on my continual development as a researcher and academic.

After submitting my thesis, but before the final examination and corrections, I was very fortunate to visit Oregon State University. My time there was very enjoyable and highly productive. For this fantastic experience, I would like to add thanks to Kagan Tumer for welcoming me into his lab, to Logan Yliniemi and his family for hosting me, and to Santander and the Artificial Intelligence research group at York for their generous funding.

Finally, and perhaps most significantly, I would like to thank Dani for being there every day throughout all the ups and downs, my family for their love and support and everyone in York who knew when I needed a pint.

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 University of York. This thesis is the result of my own investigations, except where otherwise stated. Other sources are acknowledged by explicit references.

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

1. Sam Devlin, Marek Grześ and Daniel Kudenko. An Empirical Study of Potential-Based Reward Shaping and Advice in Complex, Multi-Agent Systems In *Advances in Complex Systems (ACS)*, 2011. World Scientific Publishing Co. Pte. Ltd.

2. Sam Devlin, Marek Grześ and Daniel Kudenko. 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)*, 2011. ACM Press.

3. Sam Devlin and Daniel Kudenko. 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)*, 2011. ACM Press.

4. Sam Devlin and Daniel Kudenko. Dynamic Potential-Based Reward Shaping. In *Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, 2012. ACM Press.

5. Adam Eck, Leen-Kiat Soh, Sam Devlin and Daniel Kudenko. Potential-Based Reward Shaping for POMDPs (Extended Abstract) In *Proceedings of the 12th International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, 2013. ACM Press.

6. Kyriakos Efthymiadis, Sam Devlin and Daniel Kudenko. Overcoming Erroneous Domain Knowledge in Plan-Based Reward Shaping (Extended Abstract). In *Proceedings of the 12th International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, 2013. ACM Press.

7. Kyriakos Efthymiadis, Sam Devlin and Daniel Kudenko. Knowledge Revision for Reinforcement Learning with Abstract MDPs (Extended Abstract). In *Proceedings of the 13th International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, 2014 (In Press).

8. Sam Devlin, Logan Yliniemi, Kagan Tumer and Daniel Kudenko. Potential-Based Difference Rewards for Multiagent Reinforcement Learning. In *Proceedings of the 13th International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, 2014 (In Press).

9. Kyriakos Efthymiadis, Sam Devlin and Daniel Kudenko. Overcoming Incorrect Knowledge in Plan-Based Reward Shaping. In *Knowledge Engineering Review (KER)*, Cambridge Journals (In Press).

10. Yann-Michaël De Hauwere, Sam Devlin, Daniel Kudenko and Ann Nowé. Context Sensitive Reward Shaping for Sparse Interaction Multi-Agent Systems. In *Knowledge Engineering Review (KER)*, Cambridge Journals (In Press).

11. Sam Devlin and Daniel Kudenko. Plan-Based Reward Shaping for Multi-Agent Reinforcement Learning. In *Knowledge Engineering Review (KER)*, Cambridge Journals (In Press).

CHAPTER 1

Introduction and Motivation

Machine learning is the process of automatically improving a process through experience. My chosen field of research from within this broad capability is reinforcement learning (RL). Processes, or agents as they are commonly known in the field, are deployed into an environment they must adapt to and perform in. To do so agents typically receive no prior knowledge on how to behave, nor any explicit labeling of a behaviour as right or wrong. Instead agents must explore new states to experience a scalar reward provided by the environment. Gradually the agent can exploit the knowledge learnt of which state-action pairs are expected to maximise the long term reward received. [Mitchell, 1997]

In complex domains, learning the optimal behaviour of an agent with no prior knowledge can be too slow for many practical applications. However, the assumption, that those deploying agents cannot impart any advice to the agent before it begins learning is often unnecessary. Typically, the designer will have some heuristic knowledge regarding what would be a suitable way to behave. Methods are available, in an approach referred to as knowledge-based RL, to incorporate the designer's knowledge into learning agents. One method popular in this approach, potential-based reward shaping (PBRS), has been proven to learn equivalent policies to agents learning without domain knowledge and demonstrated to significantly reduce the time taken for performance to converge [Ng et al., 1999].

RL research is also currently active in the direction of application to multi agent systems. Whilst the classic RL applications have focussed on a single agent learning alone in an environment, considerable interest has grown in the benefits and implications of deploying multiple learning agents into a common environment. However, despite the availability of a

vast number of multi-agent specific RL algorithms the application to complex problem domains is limited. [Shoham et al., 2007]

Multiple agents acting in one system can share workloads, be robust to individual failure and scale well with the addition of extra agents [Wooldridge, 2002]. If these agents can then learn they can also enjoy the benefits of adapting to new environments and, potentially, behave optimally in problem domains where the correct way to behave is not previously known [Buşoniu et al., 2008]. In addition to the characteristic benefits of the fields of multi-agent systems (MAS) and RL, in combination unique benefits arise. In particular the deployment of multiple agents introduces the possibility of sharing sensations, experiences and/or knowledge improving the rate of learning [Tan, 1993].

Multi-agent RL (MARL) applications to complex domains is understandably exciting, but yet severely limited due to the computational complexity of algorithms and the combined size of the state space and joint-action space. Similar to how heuristic knowledge speeds up search, giving us the A* algorithm, it has been shown again to be beneficial in single agent RL. Therefore, the inclusion of domain knowledge in MARL may yield similar reductions in suboptimal action choices that could be sufficient to make the application to complex problem domains feasible.

1.1 Hypothesis

The overall aim of this thesis is to demonstrate:

> Given sufficient domain knowledge, multi-agent potential-based reward shaping can reduce the time a group of reinforcement learning agents need to learn a suitable behaviour and direct the agents towards convergence on a different joint policy whilst also guaranteed not to modify the agents' original intended goal.

1.2 Scope

For this thesis, I have chosen to only explore PBRS despite there being many other methods of knowledge based RL that may be applicable to multi-agent problem domains. For completion, these other methods of knowledge-based RL are reviewed in Chapter 2 with further deliberation regarding why they were not explored further.

However, unlike many of the other approaches, prior to work on this thesis beginning there were very few published studies using PBRS in MARL. Therefore, a number of interesting and significant questions remained open. In particular, the previous studies experimented with a very limited subset of MARL algorithms and none considered whether the theoretical guarantees of PBRS were still valid when multiple agents were in the same environment.

1.3 Thesis Overview

The next chapter provides a comprehensive review of the existing literature that this thesis builds upon, covering all material required to make the later chapters accessible for all readers.

Afterwards, in Chapter 3, I begin my empirical study of PBRS in MARL by documenting experiments with a wide variety of MARL algorithms representative of many types of algorithm never previously studied with PBRS. These studies demonstrate that PBRS has a different effect on learning in multi-agent systems than the characteristic effect guaranteed when it is applied to single-agent problem domains.

Chapter 4 then explores the theoretical explanation for this change in effect. Concluding that in MARL, PBRS is guaranteed to not alter the set of behaviours the agents may learn but may change which one they learn. This chapter also expands the definition of the additional rewards given by PBRS to allow the potential function to change over time. This more general definition significantly increases the space of reward functions guaranteed not to alter the original intended goal of the agents.

Finally, given both the increased space of reward functions and the theoretical justification of PBRS in MARL, Chapter 5 introduces a method for designing the required potential function by translating an abstract multi-agent plan that may or may not include conflicts amongst the agents.

The thesis concludes in Chapter 6 with a summary of all contributions documented with this thesis, some comments on the limits of these results and how they may be extended upon with further work in the future.

CHAPTER 2

Background and Literature Review

This chapter covers the fundamental and current state of research required to understand the topics of this thesis. Section 2.1 introduces RL covering the basic concepts used throughout my research. Section 2.2 defines the concept of MAS and then Section 2.3 discusses applying RL to MAS. The chapter closes with coverage of existing methods of knowledge-based RL.

2.1 Reinforcement Learning (RL)

RL is a type of machine learning. Machine learning is the process whereby computer programs improve through experience. Two other types of machine learning commonly known are supervised and unsupervised learning. [Mitchell, 1997]

Supervised learning requires a domain expert to label example types from which the algorithms can identify patterns and learn how to identify new examples. Unsupervised learning identifies patterns in the data and clusters similar pieces of data with no input from an expert. RL lies somewhere between the two. It does not require an expert to explicitly state if it has done wrong or right but it does receive some quantifiable input to suggest it has and uses this to reinforce what it believes to be the correct way to behave. [Mitchell, 1997]

RL is focused on goal directed learning through interaction with an environment. The environment, in single-agent RL, is everything outside of the agent [Sutton and Barto, 1998]. The agent is the learning and decision making entity in the environment. It is a program capable of independent action that makes decisions, based on its own motivations, about how to behave [Wooldridge, 2002]. Therefore, an RL agent learns how to satisfy its own motivations through experiences had in an environment.

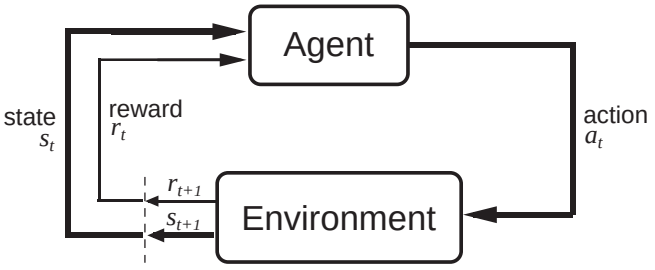


Figure 2.1: Agent-Environment Interaction [Sutton and Barto, 1998].

To gain these experiences the agent repeatedly interacts with the environment as illustrated in Figure 2.1. The interaction is a cycle beginning with the environment presenting the current situation to the agent in the form of a state representation. The agent then chooses, from a set of given actions, what to do in this state. The action the agent has taken will have some effect on the environment and, therefore, may change the current state. The environment then returns the new state and a numerical reward for the agent's decision based on the state-action-state tuple. This reward is the quantifiable input, alluded to before, that reinforces what the agent believes to be the correct way to behave. The cycle then repeats by the agent making another decision on how to act. [Sutton and Barto, 1998]

The agent uses the rewards from the environment to generate its policy. An agent's policy is a mapping from states to actions perceived favourable in an attempt to maximise the reward it will receive. In RL the policy is represented as a value function. The value function maps state s and action a to the reward that will be received over time given that the agent is in state s, performs action a and continues to follow the same policy throughout the remaining interactions. [Sutton and Barto, 1998]

Value functions can be initialised pessimistically, optimistically or randomly. Pessimistic initialisation sets the value of all state-action pairs to the minimum possible value (i.e. if all rewards given are greater than or equal to 0, the initial value of 0 is pessimistic). Alternatively, optimistic initialisation sets the value of all state-action pairs to the maximum possible value. Optimistic initialisation ensures all actions are tried in all states before convergence to a fixed behaviour occurs. Pessimistic initialisation allows promising policies to become favoured by the agent quicker, but will not discover the optimal policy until it is found by exploration. Random initialisation balances the benefits of both methods.

To maximise the reward received throughout all interactions the RL agent must balance carefully the need to explore with the desire to exploit. Specifically, in each state it must choose whether to exploit an action already known to be worthwhile or to explore new options and potentially discover a more beneficial state-action pairing. With exploration the agent must occasionally take random actions to learn their reward and to discover if they lead to states of higher reward. This prevents the agent from performing optimally, but without exploration

the agent will never have the knowledge to perform optimally. The balancing of exploration and exploitation is a key design decision when implementing a RL solution. [Sutton and Barto, 1998]

Common methods of action selection include greedy, ϵ-greedy and Boltzmann/soft-max. Greedy action selection will always pick the action perceived at that time to be of the highest value. This method is often combined with optimistic initialisation of the value function to ensure ample exploration. ϵ-greedy picks the highest valued action with probability ϵ and a random action with probability $1 - \epsilon$. This method is often implemented with ϵ gradually declining to 0[1]. Boltzmann (or soft-max) action selection gives each action a probability of being chosen based on their current relative values (i.e. actions of higher value are more likely to be picked).

2.1.1 Markov Decision Processes (MDP)

RL is used to solve problem domains modelled mathematically as a MDP [Puterman, 1994]. A MDP is a 4-tuple $< S, A, T, R >$, where:

- $\mathbf{S}$ is the state space,
  a set of all possible states;
- $\mathbf{A}$ is the action space,
  a set of all possible actions;
- $\mathbf{T}$ is the transition probability function: $T(s, a, s') = Pr(s'|s, a)$,
  the probability that action a in state s will lead to state s';
- $\mathbf{R}$ is the reward function: $R(s, a, s') \in \mathbb{R}$,
  the reward received when action a transitions an agent from state s to state s'.

In an MDP the outcome of a state-action pair depends solely on the current state. All previous actions and states have no effect on the outcome. Formally this condition is known as the Markov property and all systems where the optimal next action can be chosen by just knowing the current state are said to hold it. [Puterman, 1994]

Semi-Markov Decision Processes (SMDP)

Every action in an MDP is assumed to take the same length of time. However, in many practical applications this is often not true. For example, a robot learning to move through a building may have actions to step forward and to turn. To maintain stability it is likely that the turning action will need to be slower than taking a step forward.

SMDPs are a generalisation of MDPs that allow abstract actions; actions that can take multiple time-steps and may have different durations to other actions available to the agent within the same state. SMDPs are commonly used with hierarchical RL, an approach to knowledge-based RL that will be discussed further in Section 2.4.3. [Hengst, 2012]

[1] Practical experience suggests that reducing ϵ to values very close to, but not equal to, 0 is better.

Partially-Observable Markov Decision Processes (POMDP)

MDPs presume that the agent can observe the whole environment accurately at all times. When we consider applications to real world problems this will often not be true. For example, robots receive noisey information from sensors that can only sense their local environment.

To model these applications we can extend MDPs to POMDPs [Kaelbling et al., 1998]. A POMDP is a 6-tuple $< S, A, T, R, \Omega, O >$, where:

- $\mathbf{S}$ Is the state space,
  a set of all possible states;
- $\mathbf{A}$ is the action space,
  a set of all possible actions;
- $\mathbf{T}$ is the transition probability function: $T(s, a, s') = Pr(s'|s, a)$,
  the probability that action a in state s will lead to state s';
- $\mathbf{R}$ is the reward function: $R(s, a, s') \in \mathbb{R}$,
  the reward received when action a transitions an agent from state s to state s';
- $\mathbf{\Omega}$ is the observation space,
  a set of all possible observations;
- $\mathbf{O}$ is the observation probability function: $O(s', a, o) = Pr(o|s', a)$,
  the probability of receiving observation o when action a caused transition to the state s'.

2.1.2 Algorithms

Most RL algorithms to solve an MDP can be grouped into three general types: dynamic programming, temporal difference learning and Monte Carlo methods. In this section each will be discussed in turn promoting its own merits. To overview, dynamic programming can be used when the reward function and transition probability function are known. If they are not, but the Markov property holds for the given problem domain, temporal difference learning can be used. If either the reward function or the transition probability function are unknown and the problem domain does not hold the Markov property, Monte Carlo methods are more appropriate.

Dynamic Programming

When the MDP is entirely known there is no need to simulate interactions with the environment, instead the optimal policy of an agent can be calculated. Dynamic programming constitutes a collection of algorithms that solve known MDP's finding an exact mapping of state-action to maximise reward received. [Sutton and Barto, 1998]

An example of dynamic programming is policy iteration. Policy iteration starts with a random policy, and computes for one state at a time if a better action can be performed. By only changing

one state-action pair at a time for a higher rewarded state-action pair the algorithm can guarantee to monotonically improve the overall policy. [Puterman, 1994]

Temporal Difference Learning

Temporal difference learning algorithms are iterative methods used online during interactions with the environment. On each interaction with the environment the algorithms gradually reduce discrepancies between the expected reward and the reward received by updating the value function so it converges towards an optimal policy. [Mitchell, 1997]

The two most common algorithms of this type are: Q-Learning [Watkins and Dayan, 1992] and SARSA [Rummery and Niranjan, 1994]. Q-Learning is an off-policy learning algorithm, meaning the Q-Learning agent updates its value function whilst following an independent policy. Specifically, after every state-action-reward-state tuple experienced, a Q-learning agent updates its value function using the update rule:

$$Q(s,a) \leftarrow Q(s,a) + \alpha[r + \gamma \max_{a'} Q(s',a') - Q(s,a)] \tag{2.1}$$

where s is the initial state, a the action taken, α the learning rate, γ the discount factor and s' the resultant state.

The learning rate and discount factors are parameters set for each experiment. The learning rate affects how big the change in estimated Q-value is. The discount factor affects the agent's preference over immediate rewards and rewards it may receive later.

Alternatively, SARSA[2] is an on-policy learning algorithm and so, in direct contrast to Q-Learning, follows the policy currently represented by the value function that is simultaneously being updated. SARSA agents update their value function after every state-action-reward-state-action tuple experienced with the update rule:

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

where s is the source state, a the action taken, α the learning rate, γ the discount factor, s' the resultant state and a' the action taken in the resultant state.

These update rules are applicable to MDPs, but can be modified for semi-MDPs by using the update rule:

$$Q(s,a) \leftarrow Q(s,a) + \alpha[r + \gamma^{\Delta t} Q(s',a') - Q(s,a)] \tag{2.3}$$

where s is the source state, a the action taken, α the learning rate, γ the discount factor, s' the resultant state, $a' = \max_{a'} Q(s',a')$ if using Q-Learning or a' is the action taken in the resultant state if using SARSA and Δt is the change in time between states s and s'. [Sutton et al., 1999]

[2] Or Modified Connectionist Q-Learning as it was originally named.[Rummery and Niranjan, 1994]

Or for POMDPs:

$$Q(o,a) \leftarrow Q(o,a) + \alpha[r + \gamma Q(o',a') - Q(o,a)] \tag{2.4}$$

where o is the source observation, a the action taken, α the learning rate, γ the discount factor, o' the resultant observation and $a' = \max_{a'} Q(o',a')$ if using Q-Learning or a' is the action taken in the resultant state if using SARSA

Both Q-Learning and SARSA have been proven to converge to the optimal policy in an MDP provided the following specific requirements are met [Sutton and Barto, 1998]:

1. All state-action pairs are experienced an infinite number of times;
2. Exploration reduces to zero;
3. The learning rate (α) reduces to zero;
4. The Markov property holds.

Monte Carlo Methods

Learning by pure temporal difference methods only updates the value of the last state-action pair. However, if the Markov property does not hold, the entire history of states and actions may be responsible for the reward received. Monte Carlo methods consider all state-action pairs experienced during an interaction with the environment and only update once the interaction stops. Therefore, for problem domains where the Markov property does not hold, Monte Carlo methods are more suitable to learning. [Sutton and Barto, 1998]

To find the optimal policy with Monte Carlo methods, multiple complete sets of interactions, known as episodes, must occur. For problem domains with long or continuous episodes this is impractical and so in practice some balance between Monte Carlo methods and temporal difference learning is often required.

2.1.3 Eligibility Traces

Pure temporal difference learning and Monte Carlo methods represent two extreme cases, the former considering only the current state-action pair and the latter considering all state-action pairs. To balance the benefits of both approaches, eligibility traces can be used to breach the gap between these two extremes.

Eligibility traces are a temporary store of previously experienced state-action pairs and associated eligibility values. When a state-action pair is experienced it is added to the trace with an eligibility of one. The eligibility of the state-action pair is decayed by multiplication with the decay rate parameter (λ), where $0 \geq \lambda \leq 1$, after each subsequent experience of other state-action pairs. If a state-action pair's eligibility falls below a set threshold they are removed from the trace. All state-action pairs in the trace are updated after a reward is received, typically

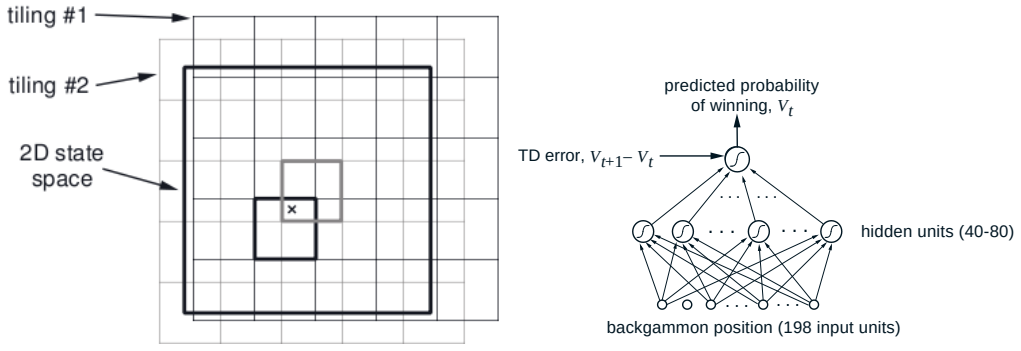
receiving a discounted amount of the reward dependent on their current eligibility. If λ is set to zero, only the current state-action pair is considered and so the resultant algorithm remains a temporal difference learning algorithm. If it is set to one, then the algorithm is effectively a Monte Carlo method as all state-actions experienced will remain in the trace. However, if set to a value between the two, a blended approach is being used. Practical applications in problem domains where the Markov property does not hold tend to benefit from such approaches. [Sutton and Barto, 1998]

2.1.4 Function Approximation

Theoretically, for RL algorithms to converge, every state-action pair must be visited an infinite number of times. This is feasible in small environments. In such applications, tabular state representations, which store the expected reward of every state-action pair, can be used. However, in real applications this is often not feasible, either due to memory constraints or computation time, and so other methods must be used to overcome this requirement.

This difficulty with handling large state and/or action spaces is known commonly in the field as the state-space explosion. Specifically, with every feature added to a state representation there is an exponential growth in the number of states.

The typical method of scaling RL to handle the state-space explosion is to generalise across states and/or actions. By approximating state-action pairs an agent can learn a reasonable behaviour quicker than a tabular implementation that must specifically visit every state-action pair. This form of generalisation is known as function approximation. [Sutton and Barto, 1998]



(a) Tile Coding (b) Neural Network

Figure 2.2: Example Function Approximation Techniques [Sutton and Barto, 1998].

Common examples of function approximation used in RL are tile coding, illustrated in Figure 2.2a, and neural networks, illustrated in Figure 2.2b.

Tile coding discretises the input space into an exhaustive partioning where one tile represents multiple states or state-action pairs. Expected values of reward are then stored per tile instead of per state-action pair thus reducing the number of features to learn from. To increase sensitivity,

multiple tilings can be overlaid one another at slight displacements as illustrated in the example Figure 2.2a. In the example the number of features is still reduced with two tilings, but a finer level of generalisation is now possible than with simply one tiling. [Sutton and Barto, 1998]

Pre-existing work on RoboCup KeepAway and TakeAway demonstrate successful examples of using tile coding to make RL solutions feasible in complex problem domains [Iscen and Erogul, 2008; Stone and Sutton, 2001].

Alternatively, Tesauro's application of RL to backgammon was approximated by a neural network [Tesauro, 1994]. The example neural network illustration, Figure 2.2b, is representative of the implementation used in this classic application.

Neural networks and tile coding are only two popular methods of function approximation used in RL. Many others are occasionally used and many further still exist as this is a developing field in its own right. The curious reader is directed towards [Busoniu et al., 2010; Ripley, 2008; Sutton and Barto, 1998] for more in-depth coverage of function approximation techniques applied to RL and of the field as a whole respectively.

Furthermore, function approximation is only one approach to handling the state-space explosion . An alternative approach is batch RL [Lange et al., 2012], a term used for algorithms that store state-action-reward tuples and process them in batches often reusing the same tuple multiple times. Some examples of this include the algorithms Fitted Q-Iteration (FQI) [Ernst et al., 2005] and Least-Squares Policy Iteration (LSPI) [Lagoudakis and Parr, 2003].

2.2 Multi-Agent Systems (MAS)

Although single-agent RL has been successfully applied to a number of problem domains, it is becoming less common in computer systems to have only one entity acting in an environment. Instead the benefits of having multiple entities deployed in a common environment has begun to shift how systems are designed. A new field dedicated to the study of such systems has arisen, known as MAS [Weiss, 2013; Wooldridge, 2002].

Previously introduced in Section 2.1, an agent is a program capable of independent action. Therefore, intuitively a MAS is any system containing multiple programs capable of independent action. Simple enough to grasp but the implication of multiple, independent agents are huge. Some examples of MAS include (amongst many others) electronic marketplaces [Fasli, 2006], cognitive radios [Akyildiz et al., 2006; Haykin, 2005] and RoboCup Soccer and Rescue simulators[3]. MAS benefit from being robust to individual failure and scale well to larger domains as each entity is typically cheaper than a single agent controlling the entire environment alone. Being inherently distributed, MAS can benefit from parallel computation speeding up the time taken to complete a task. [Buşoniu et al., 2008; Wooldridge, 2002]

No one formal definition for MAS is currently agreed upon but a number of interesting features unique to MAS can be extracted from observing a handful of definitions. Starting with

[3] See http://www.robocup.org/ for more details

Wooldridge [2002], an agent is defined as above but with the interesting note of performing actions on behalf of a user or owner. It is important to realise in many MAS each agent may be controlled by a different organisation, potentially competing with others in the system and so for many applications assuming homogeneity is not sufficient.

Shoham and Leyton-Brown [2008] continue this trend by adding that each of these autonomous entities will have either diverging information, interests or both. Diverging interest again highlights the potential for conflict and competitiveness, whilst diverging information highlights another feature of many MAS; partial observability. In MAS it is very common that no one agent will have complete knowledge of the entire environment, instead they will only be able to observe their local environment. Both in co-operative systems and competitive systems methods to handle this are required.

Fasli [2006] notes that MAS tend to be loosely coupled, the actions of one agent are affected by but not dependent on those of another agent, and that agents can interact in their common environment through a set of rules. These rules will be specific to the problem domain, in some systems communication is allowed continuously throughout whilst in others none is possible.

Finally, Fasli [2006] also reduced the most significant differences between MAS and single-agent systems to four simple dimensions; common environment, interaction, control and knowledge. The common environment implies the loose coupling, every agent is changing the same state. The interaction dimension is the added complication of how agents interact with one another, be it competitively, co-operatively or sometimes a mix of both. Control is with regard to the change from using centralised control algorithms to a decentralised approach and knowledge tackles the issue that no one agent has complete knowledge.

Although no one definition has been agreed upon, the above summary is hoped to give the reader a firm understanding of the type of systems this thesis will be focused on. The concept of MAS is relatively simple, but the implications of multiple agents behaving independently in a common environment are complex and a rich area for potential research.

2.3 Multi-Agent Reinforcement Learning (MARL)

MARL is the deployment of multiple RL agents in a common environment [Nowé et al., 2012; Tuyls and Weiss, 2012]. By combining the techniques of RL with the concept of MAS, MARL inherits benefits from both fields. From RL, MARL agents can improve their performance online whilst acting in their intended domain and react to changes in the environment [Sutton and Barto, 1998]. Whilst from MAS, MARL agents can benefit from distributed computation allowing agents to share workloads, be robust to the failure of one and scalable with the addition of more [Weiss, 2013; Wooldridge, 2002]. The combination of two fields also introduces unique benefits, MARL agents can share experiences, an expert agent can teach a struggling agent or the struggling agent can mimic an expert [Buşoniu et al., 2008; Tan, 1993]. These possibilities could lead to many exciting implementations of MARL, ranging from applications as varied as

managing air traffic flow [Agogino and Tumer, 2012] all the way to robotic soccer [Iscen and Erogul, 2008; Min et al., 2008; Stone et al., 2005].

However, the great potential of MARL comes not without complications. The existing difficulty of single-agent RL, the state-space explosion, is more prominent in MARL as each agent adds its own variables to the joint state-action space. Therefore, each time an agent is added to the environment there is an exponential increase in the number of features observable [Buşoniu et al., 2008; Stone and Veloso, 2000]. Furthermore, with multiple agents learning in a MAS, the environment is no longer static as is often the case in single-agent RL. Therefore, as the transition probability function is now dependent on the joint action, if an agent can only observe its own action the Markov property does not hold. With these difficulties in mind it may be beneficial to co-ordinate but doing so when each agent is independently motivated can be challenging [Buşoniu et al., 2008]. Finally, given that reinforcement rewards are often sparse and control is now decentralised, when a reward is received from the environment which agent(s) should receive it? What reinforcement should an agent receive when its actions help another agent but not itself? These questions form the structural credit assignment problem and provide a unique challenge caused by the combination of MAS and RL [Stone and Veloso, 2000; Tumer and Khani, 2009].

Few, if any, of the complications of MARL are considered solved and so the topic of learning in MAS remains a rewarding and open research area [Shoham et al., 2007; Stone, 2007]. In particular, Tuyls and Weiss [2012] identifies three main challenges; classification limitations, extending the scope and multi-agent learning in complex systems. This thesis contributes to making the last of these key open problems feasible.

2.3.1 Stochastic Games (SG)

RL algorithms solve MDPs, but MDPs do not intuitively model MAS. Instead, a generalisation of MDPs to the multi-agent case, SGs [Shapley, 1953; Myerson, 1990], is commonly the underlying mathematical model of MARL [Buşoniu et al., 2008].

A SG of n agents is a $2n + 2$-tuple $< S, A_1, ..., A_n, T, R_1, ..., R_n >$ where:

- $\mathbf{S}$ is the state space,
  a set of all possible states;
- $\mathbf{A}_i$ is the action space of agent i,
  a set of all actions possible by agent i;
- $\mathbf{T}$ is the transition probability function: $T(s, \mathbf{a}, s') = Pr(s'|s, \mathbf{a})$,
  the probability that joint-action $\mathbf{a}$ (the product of all actions chosen by the set of agents) in state s will lead to state s';
- $\mathbf{R}_i$ is the reward function of agent i: $R_i(s, \mathbf{a}, s') \in \mathbb{R}$,
  the reward received when joint-action $\mathbf{a}$ transitions an agent from state s to state s'.

SGs can be fully cooperative, fully competitive or a mix of both. In fully cooperative SGs, also known as team games, the reward function for all agents is the same. At the other extreme, games with two players where the sum of rewards received for each pair of states and joint-actions is zero are fully competitive. Games with a mixture of both competitive and cooperative elements are known as general-sum games.

Unlike in MDPs, there is no clear concept of an optimal policy in SGs (except for the special case of fully cooperative games) as some trade off between each of the agents' goals must occur. However, from the related field of game theory, many alternative solution concepts can be used. Game theory is closely tied with MARL as many algorithms combine aspects of dynamic programming and/or temporal difference learning with theories from the field [Buşoniu et al., 2008; Nowé et al., 2012].

Typically, MARL agents learn a joint policy representative of a Nash equilibrium. However, often it would be preferable for them to learn a Pareto optimal joint policy. These solution concepts will be detailed further in the following subsections. For alternative solution concepts or more information on game theory, the interested reader is recommended either Fudenberg and Tirole [1991] or Dixit et al. [2004].

2.3.2 Nash Equilibrium

John Nash's famous concept of equilibrium in non-cooperative games [Nash, 1951] is highly prominent in MARL [Akchurina, 2009; Hu and Wellman, 2003; Littman, 2001; Wang and Sandholm, 2003].

A Nash equilibrium is a joint-strategy where no agent would benefit from changing their own strategy assuming all other agents will stick to their current strategy. It can be pure, in that each agent always plays the same actions, or mixed, where each action is assigned a probability of being chosen. Furthermore, in games of finite agents with finite actions there will always be at least one present. [Nash, 1951]

Formally a joint policy π^{NE} is a Nash equilibrium provided:

$$\forall i \in 1 \dots n, \pi_i \in \Pi_i | R_i(\pi_i^{NE} \cup \pi_{-i}^{NE}) \geq R_i(\pi_i \cup \pi_{-i}^{NE}) \tag{2.5}$$

where n is the number of agents, Π_i is the set of all possible policies of agent i, R_i is the reward function for agent i, π_i^{NE} is a specific policy of agent i and π_{-i}^{NE} is the joint policy of all agents except agent i following their own fixed specific policy. If the inequality holds for all agents, the joint policy π^{NE} of each agent following its policy π_i^{NE} is a Nash equilibrium.

2.3.3 Pareto Optimality

A joint policy a is said to Pareto dominate another joint policy b if one or more agents receive a higher reward and all other agents receive the same reward. Any joint policy that is not Pareto dominated by another is Pareto optimal. Therefore, if a joint policy is Pareto optimal, there is no

joint-policy that would increase any agents reward without also reducing the reward of another agent. [Fudenberg and Tirole, 1991]

2.3.4 Algorithms

This section will give a brief insight into the history of MARL algorithms attempting to cover all the most influential papers with a focus on those that have been used during this body of work.

Multiple Independent Learners

The simplest MARL solution is to deploy multiple RL agents using standard algorithms from the single-agent literature. These agents may include features regarding the other agents in their state representation or ignore them entirely. This approach, often referred to as multiple independent learners [Claus and Boutilier, 1998], was popular in early MARL research [Tan, 1993] and remains a common solution [Iscen and Erogul, 2008] despite subsequent algorithm development.

Joint-Action Learners

The alternative simple solution is for agents to observe and learn a value function for the joint-actions taken, as opposed to their own action alone. This approach benefits from full observation as the apparent stochasticity of the environment reduces but can suffer as the required space for the value function and number of experiences needed to learn it grows exponentially with each agent added [Claus and Boutilier, 1998].

Competitive Games

Littman [1994] introduced the mini-max Q-learning algorithm; the first example of combining temporal-difference learning with game theory to solve a subset of MARL problems. Mini-max Q-learning is a modified joint-action learner for fully competitive (i.e. two player, zero-sum) games. On each action selection, a mini-max Q-learning agent will choose the highest valued action assuming the opposing agent will attempt to minimise the reward the learning agent can receive. This algorithm is guaranteed to converge to a fixed policy that receives the highest reward possible against the worst opponent possible (i.e. the rational opponent who attempts to minimise the agent's reward), formally termed the minimax return. If the opponent is not the worst possible, the mini-max Q-learning agent will receive a higher reward.

More recent work by Brafman and Tennenholtz [2003] provides an approximate alternative, the RMax algorithm, which guarantees covergence to the probabilistic minimax return in polynomial time. Research into zero-sum games is largely stagnant with these algorithms appearing to be accepted solutions, however, some recent work has improved upon RMax by adding targeted optimality against memory-bounded agents [Chakraborty and Stone, 2010].

Cooperative Games

Lauer and Riedmiller [2000] introduced an extension to multiple independent learners for fully

cooperative games; Distributed Q-Learning. In deterministic, fully cooperative environments, if a lower reward is received for a state-action pair than for the same pair in a previous experience it can be assumed that another agent is at fault. For this reason, distributed Q-learning agents never reduce a value in their Q-table. Therefore, if all agents in a deterministic, fully cooperative enviroment use distributed Q-learning they will converge to the optimal policy.

Alternatively, for fully cooperative but stochastic environments Wang and Sandholm [2003] introduced a modified joint-action learner, Optimal Adaptive Learning, guaranteed to converge to the optimal Nash equilibrium.

General-Sum Games

In general-sum games the classic algorithm, Hu and Wellman's Nash Q-Learning, was not introduced until 2003; indicative of the greater challenge in and immaturity of research into MARL algorithms for these types of game. Before this time research was largely focussed on the two subsets of this type of game already discussed in the preceding sections.

Nash Q-Learning was proven to converge, under strict conditions, to a Nash equilibrium [Hu and Wellman, 2003]. In practice it has been shown to sometimes converge without the conditions met and is more likely to do so than multiple independent learners in general-sum games [Buşoniu et al., 2008].

However, work continued to reduce the requirements of Nash Q-learning and a recent algorithm, Nash-DE, also provably converges to Nash equilibrium but with weaker conditions than the former algorithm. [Akchurina, 2009]

Alternative Approaches

Whilst all approaches described so far have focussed on extensions of either multiple independent learners or joint-action learners, there is also some mid-ground.

For example, Future Coordinating Q-learning (FCQ-learning) agents begin as multiple independent learners and then use statistical tests to decide, for each state, whether it would be beneficial to expand the state space to include the joint-action. After learning, the resultant Q-table stores values for joint-actions in some states and individual actions alone in others. [De Hauwere et al., 2011; De Hauwere, 2011]

Similarly, Adaptive State Focus Q-learning (ASFQ-learning) provides a mid-ground between multiple indendent learners that ignore all other agents and those that include features regarding the other agents in their state representation. This algorithm starts agents learning with just their own state but, if convergence is not reached, their state representation is expanded to include the state of the other agents. [Buşoniu et al., 2005]

Finally, whilst all approaches mentioned so far are temporal difference learning algorithms, some MARL algorithms have been derived from dynamic programming and monte carlo methods too. For example, Win or Learn Fast Q - Policy Hill Climb (WoLFQ-PHC) is a policy iteration MARL algorithm that speeds up learning if it is not receiving sufficient reward presuming that it

needs to adjust quickly to other agents in the environment. [Bowling and Veloso, 2002]

2.4 Knowledge-Based Reinforcement Learning

RL algorithms typically start with value functions initialised with either a random, pessimistic or optimistic expectation of the reward to be received for each state-action pair. An often overlooked point is that when implementing RL, the designer typically has some domain knowledge specific to the problem that could guide the agent. This is a necessary requirement of the designer as, for the agent to learn, suitable features must be chosen to be part of the state representation.

Knowledge-based RL is the study of incorporating domain knowledge into an RL agent to guide exploration. By providing worthwhile information, it is possible to reduce the number of sub-optimal decisions made by an agent and so reduce the impact of the state-space explosion.

These methods can be compared to that of A* search. By using heuristic knowledge to guide search, performance can be significantly improved over uninformed search algorithms. It is intuitive that a similar approach is beneficial in RL.

2.4.1 Reward Shaping

One promising approach to incorporate knowledge into RL is reward shaping. Reward shaping is the addition of a reward by the designer to that of the reward naturally received from the environment. Rewards provide an intuitive representation of domain knowledge, especially to a designer who may have already designed the environment's reward function. Furthermore, reward shaping requires no modification of the agent or the environment making implementation relatively simple.

However, early work showed, if used poorly, reward shaping can be detrimental to learning. In an application of learning to ride a bicycle [Randløv and Alstrom, 1998], the RL agent discovered that it could benefit more from the additional reward encouraging it to stay balanced by cycling in circles than it could for cycling to the target destination from the environment's reward function. With the poorly designed shaping function the agent converged to a policy that never reached the goal.

Potential-Based Reward Shaping (PBRS)

To avoid such problems, PBRS was proposed [Ng et al., 1999]. PBRS defines the additional reward given as the difference in potential of the source and resultant state. Formally:

$$F(s, s') = \gamma\Phi(s') - \Phi(s) \tag{2.6}$$

where γ must be the same discount factor as used in the agent's update rule and Φ is the potential function mapping states to potentials.

A state's potential is intended to represent the designers preference for the agent to be in that state. For example, it is typical to set potentials close to a goal state high and then linearly

decrease the potential of states as they get further from the goal. PBRS will then encourage the agent to move towards the goal.

Ng et al. [1999] proved that PBRS, defined according to Equation 2.6, does not alter the optimal policy of a single agent in both infinite- and finite- state MDPs.

Wiewiora [2003] proved that an agent learning with PBRS and zero Q-table initialisation will behave identically to an agent without reward shaping when the latter agent's value function is initialised with the same potential function.

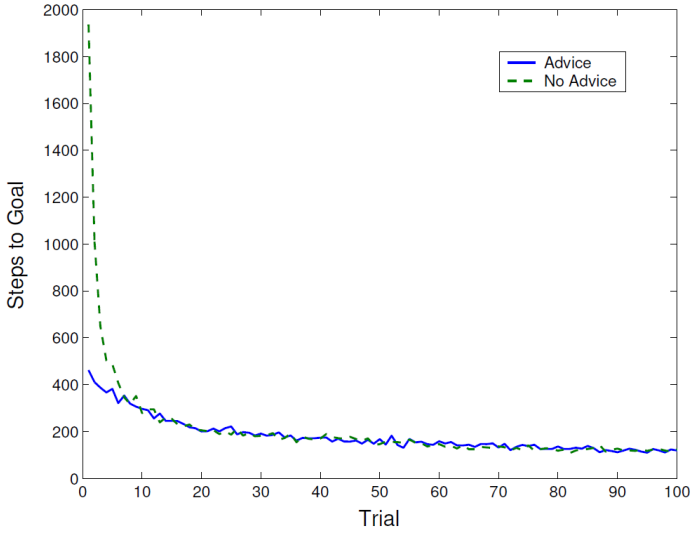


Figure 2.3: Typical Effect of PBRS on Single-Agent RL from Wiewiora et al. [2003].

Figure 2.3 illustrates the typical learning curve of an RL agent with and without PBRS, presuming a good heuristic is used for the potential function. This example is taken from Wiewiora et al. [2003], and shows an agent learning the classic RL problem domain of mountain car both with and without PBRS (or "advice").

Note that immediately upon the start of learning, the agent with PBRS starts with a better performing policy than the agent without. This occurs due to the equivalence between PBRS and Q-table initialisation. At convergence, regardless of whether the agent received PBRS or not, the agent has learnt the optimal policy. In between, the agent with PBRS has a period where it significantly outperforms the agent without. This decreased time to convergence is a large benefit of PBRS, especially in complex problem domains.

Potential-Based Advice

PBRS, as defined by Ng et al. [1999], can only incorporate knowledge regarding preference of states. To include background knowledge regarding favourable actions in reward shaping whilst still maintaining the guarantees of policy invariance, further conditions must be met [Wiewiora

et al., 2003].

Specifically, Wiewiora et al. [2003] identified two methods; look-ahead advice, formally defined in Equation 2.7, and look-back advice, formally defined in Equation 2.10. Please note that in both methods, the potential is now defined as a function of both state and action, rather than just state alone.

Look-ahead advice shapes an agent's reward when moving from state s to s' by action a based on the difference in potential between state-action pairs (s, a) and (s', a'), where a' is defined as in the agent's update rule. Therefore, if using SARSA, a' will be the next action the agent will take or, if using Q-learning, the highest valued action in state s'. Formally:

$$F(s, a, s', a') = \gamma\Phi(s', a') - \Phi(s, a) \tag{2.7}$$

To guarantee look-ahead advice maintains policy invariance, the agent's policy must choose the action with the maximum sum of both Q-value and potential. Formally:

$$\pi(s) = argmax_a\{Q(s, a) + \Phi(s, a)\} \tag{2.8}$$

where $\pi(s)$ is the policy (action the agent will choose) in state s, $Q(s, a)$ is the current estimate of the value of taking action a in state s and $\Phi(s, a)$ is the potential of the state-action pair (s, a).

This is neccesary to maintain the guarantee of policy invariance, because given additional rewards of this form the true value of all state-action pairs becomes:

$$Q^*_\Phi(s, a) = Q^*(s, a) - \Phi(s, a) \tag{2.9}$$

where $Q^*_\Phi(s, a)$ is the value of taking action a in state s when receiving PBRS, $Q^*(s, a)$ is the true value of taking action a in state s when receiving only the original rewards from the environment and $\Phi(s, a)$ is, as before, the potential of the state-action pair (s, a).

Given that the value of different actions within the same state may be modified by different amounts, the ordering of preference over actions within that state may change. However, if the agent chooses actions by Equation 2.8, the ordering is maintained once convergence is reached.

If using look-ahead advice, action a' has not yet been performed when the additional reward is received. Alternatively, look-back advice shapes an agent's reward when moving on to state s'' after action a' is used in state s' based on the difference in potential between state-action pairs (s, a) and (s', a') which have now both already occurred. Formally:

$$F(s, a, s', a') = \Phi(s', a') - \gamma^{-1}\Phi(s, a) \tag{2.10}$$

With look-back advice, Wiewiora et al. [2003] recommend using an on-policy learning algorithm (e.g. SARSA) and a method of action selection invariant to a constant addition to all actions in a state (e.g. greedy, ϵ-greedy or Boltzmann/soft-max).

An agent using look-back advice is not guaranteed, but has been empirically demonstrated, to converge to the same Q-values as the agent would have without advice. Furthermore, no counter example has been published that illustrates a case where look-back advice does alter the optimal policy.

Wiewiora et al. [2003] recommend look-ahead advice for when the prior knowledge predominately identified which states are preferred whilst look-back advice is recommended for when the prior knowledge predominately recommended actions. If the knowledge given is entirely state-based then PBRS alone suffices.

Plan-Based Reward Shaping

Reward shaping is typically implemented bespoke for each new environment using domain-specific heuristic knowledge [Babes et al., 2008; Devlin et al., 2011; Randløv and Alstrom, 1998] but some attempts have been made to automate [Grześ and Kudenko, 2008; Marthi, 2007] and semi-automate [Grześ and Kudenko, 2008] the encoding of knowledge into a reward signal. Automating the process requires no previous knowledge and can be applied generally to any problem domain. The results are typically better than without shaping but less than agents shaped by prior knowledge. Semi-automated methods require prior knowledge to be put in but then automate the transformation of this knowledge into a potential function.

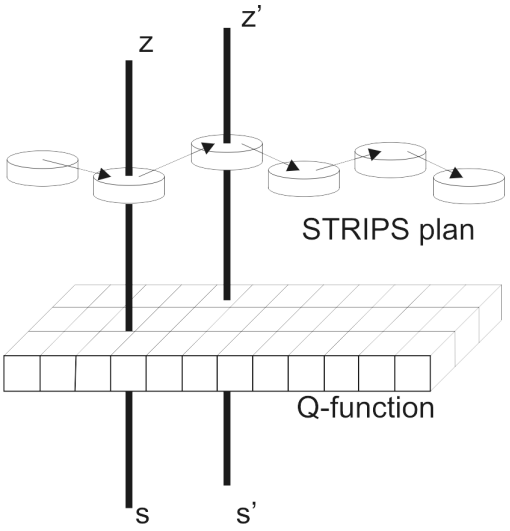


Figure 2.4: Plan-Based Reward Shaping.

Plan-based reward shaping, an established semi-automated method, uses a STRIPS planner to generate high-level plans. The STRIPS plan is then converted to a state-based representation, as illustrated in Figure 2.4, where each state in the high level plan maps to one or more in the low level environment. This representation is encoded into a potential function where states later in the plan receive a higher potential than those lower or not in the plan. Formally:

$$\Phi(s) = CurrentStepInPlan * \omega \tag{2.11}$$

where ω is a scaling factor and $CurrentStepInPlan$ is the number of states before the corresponding high-level state in the state-based representation of the agent's plan.

This potential function is then used by PBRS to encourage the agent to follow the plan without altering the agent's goal. The process of learning the low-level actions necessary to execute a high-level plan is significantly easier than learning the low-level actions to maximise reward in an unknown environment and so with this knowledge agents tend to learn the optimal policy quicker. Furthermore, as many developers are already familiar with STRIPS planners, the process of implementing PBRS is now more accessible and less domain specific. [Grześ and Kudenko, 2008]

2.4.2 Multi-Agent Reward Shaping

The application of reward shaping to MARL was an underdeveloped topic when this work began. This section covers the sole exception to this, difference rewards, and all pre-existing applications of PBRS for knowledge-based MARL.

Arbitrary reward shaping has also been applied succesfully to MARL [Matarić, 1994; 1997; Stone and Veloso, 1999] but may still modify the intended goal of the original reward function or evaluation criteria and, therefore, will not be covered further in this thesis.

Difference Rewards

Difference rewards are a multi-agent specific form of reward shaping for fully cooperative stochastic games that stemmed originally from earlier work under the term "collective intelligence". Many researchers when implementing MARL design private reward functions for each agent and look to observe the emerging behaviour of the MAS. Collective intelligence research explored how to reverse this process. Instead focusing on the world utility (or team reward) and considering how to design individual reward functions that combined improve the global performance. Specifically they aim to ensure the agents do not work against the global task [Wolpert and Tumer, 1999]. A large number of application papers have been published illustrating how the approach of difference rewards can overcome problems typical of individual agents behaving greedily to improve their own individual reward [Tumer and Wolpert, 2000; Tumer and Khani, 2009; Agogino and Tumer, 2012; Agogino et al., 2012].

One specific example [Tumer and Wolpert, 2000] of this prevents the occurrence of the Braess paradox in a network routing problem. The specific networks, illustrated in Figure 2.5, both show routes between the source S and the destination D. With n agents travelling from S to D, passing through towns V_1 costs $10n$, V_2 costs $50 + n$ and V_3 costs $10 + n$. The sole difference between the two networks is the addition of a significantly cheaper route. Intuitively, the overall cost of all agents travelling across Net B should be less than that of Net A. However, agents following

an ideal shortest path algorithm suffer from not considering the world utility and can potentially increase the cost of every agent raising the global cost and incurring the Braess paradox.

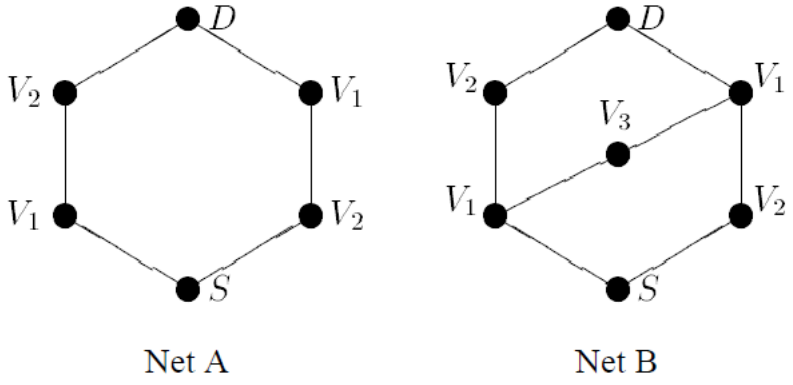


Figure 2.5: An Example of Braess Paradox [Tumer and Wolpert, 2000].

To overcome this the reward function was aligned with the world utility. Instead of all receiving the global reward, each agent was rewarded the difference between the world utility and what it would have been if the agent had not acted. This shaped reward function results in an increased value for the individual agent if and only if a corresponding increase occurs in the world utility. This method of multi-agent reward shaping is now known as difference rewards [Agogino and Tumer, 2012; Agogino et al., 2012], the formal definition of which is commonly:

$$D_i = G(z) - G(z - z_i) \tag{2.12}$$

where D_i is the reward received by agent i, $G(z)$ is the global reward all agents would have received from the environment and $G(z - z_i)$ is the global reward all agents would have received from the environment without agent i (often referred to as the counterfactual).

Unlike PBRS, difference rewards have only been applied to fully cooperative stochastic games. Difference rewards were also a more strongly established concept in MARL with rapid development on-going at Oregon State University. For these reasons they are not the focus of this thesis. Furthermore, as the counterfactual term does not depend on previous states, difference rewards and PBRS are not equivalent. They are both methods of reward shaping, but the additional rewards they give $G(z - z_i)$ and $F(s, s')$ differ greatly. However, although they are not equivalent, the two can be used together [Devlin et al., 2014]; a concept I will discuss further in Chapter 6.

Early Work on PBRS for MARL

Despite the benefits of PBRS demonstrated in single-agent RL being mutually beneficial to learning in MAS, little work had been attempted to apply PBRS to MARL prior to this thesis.

The first published application [Marthi, 2007] involved the automatic decomposition of a learnt shaping function to distribute amongst multiple effectors learning by the partially decentralised algorithm; decomposed SARSA [Russell and Zimdars, 2003]. The application was successful and the results empirically demonstrate the characteristic benefits of increased rate of learning and equivalent performance at convergence common to single-agent PBRS. However, given that decomposed SARSA uses a single centralised agent to make action choices, this was only a partial step towards PBRS being applied to MARL.

More recently, a study applied PBRS to Q-learning in the two player, general sum game; iterated prisoner's dilemma [Babes et al., 2008]. Amongst other experiments, this study documents experiments with one agent in the game learning by Q-learning alone and the other agent either also learning by Q-learning alone or learning by Q-learning with PBRS. In an illustrated typical run, the agent learning by Q-learning with PBRS is seen to learn the same behaviour as Q-learning alone but at a much quicker rate. This behaviour is again typical of single-agent PBRS. More interestingly, in the summary of all results, a difference in the average performance of the agent when learning by Q-learning alone compared to Q-learning with PBRS is documented but not discussed thoroughly.

At this time, only these applications of PBRS to MARL had been published, leaving a large number of unanswered questions. In particular, neither paper considered whether the theoretical proofs they used as motivation to implement PBRS still held within a MAS. Furthermore, PBRS had only been explored with a very limited subset of algorithms that did not represent the many types of MARL specific algorithms. This thesis broaches many of these topics, contributing significantly to the current understanding of multi-agent reward shaping.

2.4.3 Alternative Methods

Reward shaping is not, however, the only method of knowledge-based RL. Many other approaches have been tried and many more still may be plausible. This section covers the most prominent and the most relevant alternative methods, with a focus on highlighting the differences and similarities between these approaches and the approaches of PBRS.

Value Function Initialisation

One obvious method of incorporating domain knowledge is to initialise the value function. This approach has been proven (as noted earlier) to be equivalent to PBRS [Wiewiora, 2003]. Therefore, by studying PBRS in MARL, this thesis will also provide insight into value function initialisation in MARL.

Feature Selection

Another simple method of knowledge-based RL, is to select features for the state representation based on domain knowledge. By excluding features that the designer knows are not relevant to the agent's decisions, the state space is reduced and the agent(s) can learn quicker. The risk of

this method, however, is that an important feature may be removed limiting how much an agent can learn. Furthermore, unlike PBRS, modifying features can change the optimal policy of an agent.

Hierarchical Reinforcement Learning

Selecting features modifies the underlying MDP or SG, another method of doing so to include domain knowledge is hierarchical RL. This method splits an initially flat MDP into a hierarchy of MDPs. Relying heavily on the theory of SMDPs, Hierarchical RL introduces subtasks as action choices. Each subtask is represented by its own separate MDP with its own reward function, has an action set consisting of either a set of primitive actions, subtasks or a combination of both and, therefore, can take a varying length of time to complete. [Barto and Mahadevan, 2003]

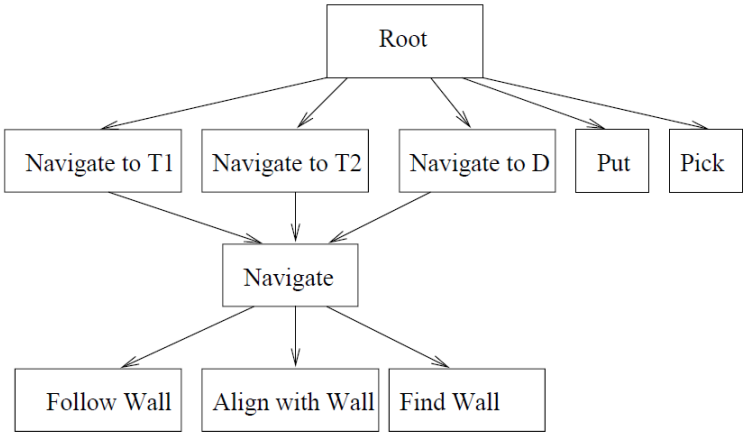


Figure 2.6: An Example Hierarchical Reinforcement Learning Solution [Makar et al., 2001].

Hierarchical RL can either augment or simplify an MDP/SG. If the top level MDP/SG includes all primitive actions for all states (as well as any subtasks), the agent's optimal policy remains the same but its state-action space will have grown. In this instance, the benefit is in the sharing of subtasks across multiple states. Alternatively, if the MDP/SG is simplified by not allowing the primitive actions in all states, the agent also benefits from a reduced state-action space but may change the optimal policy of an agent.

Figure 2.6 illustrates an example hierarchical RL solution. Note the reuse of the subtask "Navigate" and the reduced state-action space resultant of not including primitive actions at all levels. This decomposition was used in a MAS, succesfully demonstrating the benefits of increased final performance and rate of learning when using hierarchical RL in MARL [Makar et al., 2001].

This method is often compared to plan-based reward shaping. However, the two can be significantly different because agents receiving plan-based reward shaping are still learning a value function for the original flat MDP and not multiple MDPs as in hierarchical RL.

Coordination Guided Reinforcement Learning

A final alternative method of modifying the underlying MDP or SG is Coordination Guided RL [Lau et al., 2011; 2012]. All published applications of this method were in MAS but it could, in theory, be applied to single-agent problem domains too.

This approach represents domain knowledge as constraints on actions in certain states. Directly applying these to the MDP could alter the agent's intended behaviour. Therefore, Coordination Guided RL learns which constraints (if any) to apply instead.

To do so, Coordination Guided RL uses the two level learning system illustrated in Figure 2.7. The top level learns which constraints to activate, and then the low level chooses with respect to the activated constraints what actions to take in the environment.

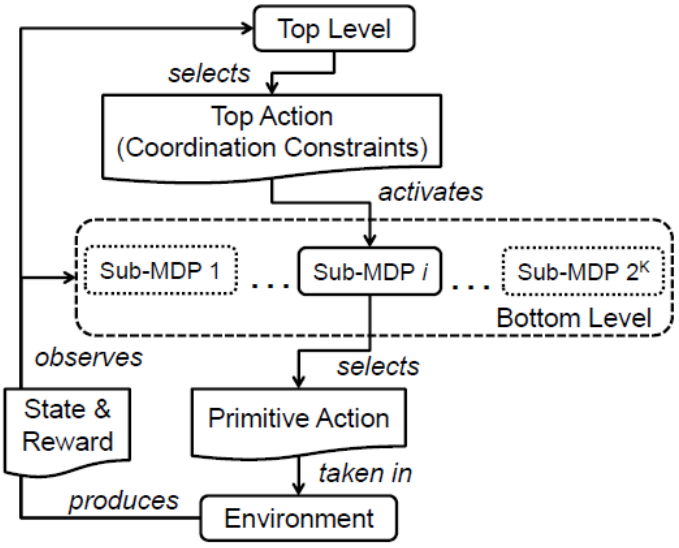


Figure 2.7: Coordination Guided RL's Two Level Learning System [Lau et al., 2012].

By allowing the top level to not apply any constraints, an agent is guaranteed to learn the same policy as an agent using the original MDP or SG. Activating a constraint theoretically means the agent is acting in a different Sub-MDP but, as only one value function is maintained for the lower level, the same number of experiences are needed to learn the true value function. When a constraint is activated, the effect on the agent is of targeted exploration. Therefore, if the domain knowledge provided is good, the agent will learn quicker

Coordination Guided RL could be compared to potential-based advice for recommending actions. The difference in knowledge representation (i.e. constraints for this method and a potential function for potential-based advice) may introduce a preference for some users. However, Coordination Guided RL cannot recommend states whilst PBRS can.

Heuristic Selection of Actions

Alternatively, domain knowledge can be used to alter an agent's action selection directly. Agents learning by Heuristically Accelerated Q-Learning [Bianchi et al., 2008] select actions by:

$$\pi(s) = \begin{cases} argmax_a[Q(s,a) + H(s,a)] & \text{if } random(0,1) < \epsilon \\ random(0, |A|) & \text{otherwise} \end{cases} \tag{2.13}$$

where $\pi(s)$ is the action the agent will take in state s, $Q(s,a)$ is the current value learnt for taking action a in state s, $random(x,y)$ is a function that returns a random number between x and y, ϵ is a probability value between 0 and 1, A is the set of all actions the agent could take and $H(s,a)$ is the heuristic value given to action a in state s and is formally defined as:

$$H(s,a) = \begin{cases} max_{a^*} Q(s,a^*) - Q(s,a) + \eta & \text{if } a = \pi^H(s) \\ 0 & \text{otherwise} \end{cases} \tag{2.14}$$

where η is a small positive value (typically 1) and $\pi^H(s)$ is the action the heuristic policy encourages in state s.

As Heuristically Accelerated Q-Learning does not modify the update rule, rewards or MDP and will still eventually allow the necessary condition of visiting every state an infinite number of times, it does not modify the optimal policy [Bianchi et al., 2004]. However, unlike PBRS, it can only provide knowledge of preferred actions and not preferred states.

This method, more generally referred to as the heuristic selection of actions, has also been applied to MARL by modifying the action selection method of mini-max Q-learning [Bianchi et al., 2007].

Integrated Partial Model

Finally, domain knowledge can also be included by modifying the update rule. This approach has been explored less but recent work [Tamar et al., 2012] has successfully applied it to develop the method Integrated Partial Model. Unlike PBRS, this method requires knowledge of the transition function for some states. It exploits this knowledge to reduce the noise in updates to the value function involving those states. The results show significant improvement in the rate of learning when compared to an agent learning without prior knowledge of the transition function.

This is a very recent development and will, therefore, not be covered further in this thesis. It may, however, become more relevant in the future if development continues.

Transfer Learning

Transfer learning is the concept of improving an agent's learning behaviour in a new (or "target") task by using previous experience in a previous (or "source") task. The source task is typically smaller and/or simpler than the target. Therefore, an agent can quickly learn the source task then

use the knowledge gained to inform initial attempts at the target task. It is a popular method in the reinforcement learning community [Taylor and Stone, 2009; Lazaric, 2012] but has also been studied extensively in psychology, originally using the term "transfer of practice" [Thorndike and Woodworth, 1901].

The concept of transfer learning does not define a method of incorporating knowledge. It is instead a source of knowledge that can be used with most of the methods of knowledge-based RL described earlier in this section. For example, PBRS has been used to implement transfer learning [Fachantidis et al., 2012]. Therefore, the contributions of this thesis towards PBRS may also be useful to future applications of transfer learning especially given recent work on multi-agent transfer learning [Boutsioukis et al., 2012; Vrancx et al., 2011]

Other similar sources of knowledge include imitation learning and apprenticeship learning. Imitation learning [Price and Boutilier, 2003] involves an agent learning to recreate the behaviour of another expert agent already capable of performing the task. Apprenticeship learning [Abbeel and Ng, 2004] also involves imitating an expert. However, this approach assumes there are no rewards from the environment and the agent must instead first learn the reward function by observing the expert's demonstration. This method has been succesfully applied to teach an agent to perform acrobatic movements with a remote control helicopter [Abbeel et al., 2010].

2.4.4 Summary

Despite differences in approach, all cited methods of knowledge-based RL can increase an agent's rate of learning. Therefore, incorporating domain knowledge remains a promising area for ongoing research into scaling RL up to complex MAS.

In particular, this thesis focuses on PBRS because of the existing theoretical proofs in single-agent problem domains and the open questions when applying it to MARL. Furthermore, PBRS can represent more types of knowledge than the Integrated Partial Model method or by the heuristic selection of actions and is simpler to implement and transfer between problem domains than modifying the underlying MDP or SG.

The next chapter will begin my study of whether PBRS can be used with MARL specific algorithms.

CHAPTER 3

Multi-Agent, Potential-Based Reward Shaping: Empirical Studies

This chapter uses three problem domains of differing complexities and scale to capture empirically the characteristic effect of PBRS on MARL. The first problem domain is small enough to quickly evaluate a collection of MARL algorithms representative of a wide range of approaches to multi-agent learning. The second and third problem domains, both within the framework of simulated robotic soccer, increase the complexity and exaggerate the typical learning behaviour of multiple agents using PBRS.

As I will discuss in Section 4.7, the theoretical results presented in the next chapter assume the use of multiple independent Q-learners. Therefore, the studies in this chapter are intended to demonstrate the wider applicability of PBRS in MARL and to introduce the reader to the effect of multi-agent PBRS before explaining it in theory in the next chapter.

3.1 Plausibility Study

The first problem domain chosen, illustrated in Figure 3.1, is a deterministic gridworld in which two agents (the red and green circles) attempt to get to two separate goals (the red and green diamonds) whilst avoiding two obstacles (the grey circles). The agents can choose at each timestep to move up, down, left or right by one square or stay still. Agents that attempt moves which would take them off of the grid, onto an obstacle, through another agent or onto another agent receive a reward of -2 for that choice. When an agent reaches its goal, it is rewarded $+10$. All other action choices are rewarded -1. If an episode reaches 1000 steps, the episode ends regardless of whether agents have reached their goals or not.

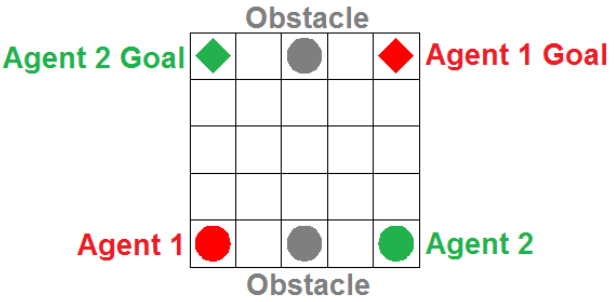


Figure 3.1: Problem Domain for Plausibility Study

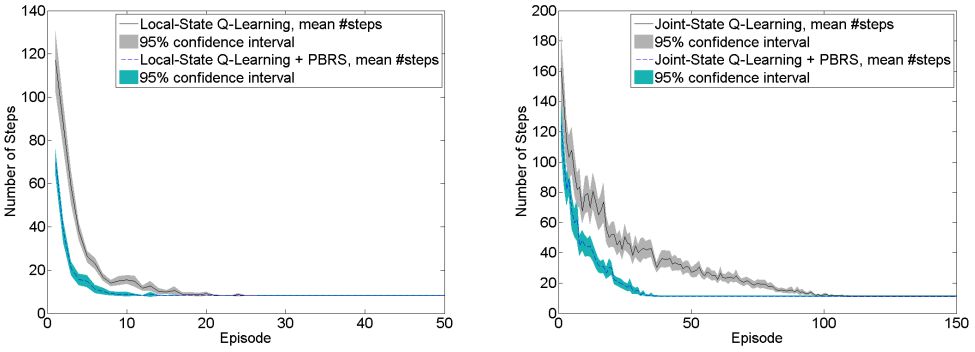
For each algorithm tested, I ran experiments both with and without PBRS to illustrate the effect it has on learning behaviour in a MAS. Agents receiving PBRS used the potential function:

$$\Phi(s) = \max_{d}(DistanceToGoal(d)) - DistanceToGoal(s) \tag{3.1}$$

This potential function increases linearly as the agent gets closer to the goal, and so encourages actions that move the agent towards its goal.

All experiments were repeated 50 times and all agents, except where noted otherwise, used the parameter settings; $\alpha = 0.3, \gamma = 0.95, \lambda = 0.5$ and $\epsilon = 0.2/episode$. These settings are all within the typical ranges used in existing literature. Furthermore, they were chosen from a large set of values tested as they worked with the largest number of algorithms tested.

Results



(a) with Local State

(b) with Joint State

Figure 3.2: Multiple Independent Q-Learners

The first approach tested was multiple independent Q-learners with a state representation

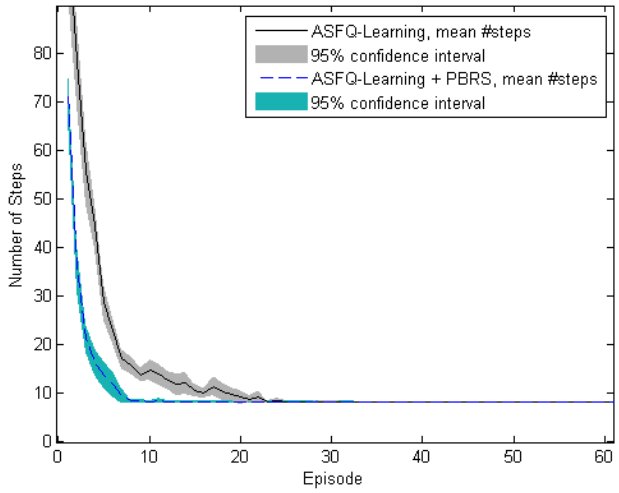


Figure 3.3: ASFQ-learning

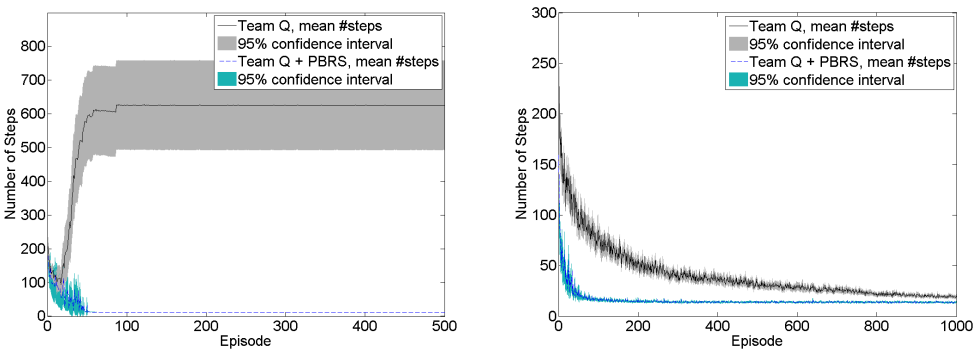
including just their location. The results, illustrated in Figure 3.2a, show the same learning behaviour when comparing agents with and without PBRS as is typical in single-agent RL. Specifically, the agents with PBRS start with a significantly better behaviour, converge earlier and to a policy of equal performance as the same agents without PBRS.

When the agents state representation is expanded to also include the location of the other agent, the pattern of learning behaviour (illustrated in Figure 3.2b) is similar and PBRS again shows similar benefits to previous single-agent applications.

The next algorithm tested was ASFQ-learning, chosen as representative of methods between multiple independent learners and joint-action learners. This algorithm required a number of additional settings all of which were held at the default values provided by one of the algorithm's original designers. Specifically, they were analysis window length $= 256$, 16 analysis stops per window and a zero margin $= 0.05$ or 5%. These settings were used by ASFQ-learning to decide whether to switch from using the local state to the joint state. The results, illustrated in Figure 3.3, again show the same benefits that are thus far characteristic of PBRS.

In other collaborative work, with Dr. Yann-Michaël De Hauwere and Professor Ann Nowé, we have also tested their algorithm FCQ-learning with and without PBRS in a similar setting. The results yet again showed the same improvement in learning when using PBRS and further support the argument that PBRS can be used with approaches between multiple independent learners and joint action learners.[De Hauwere et al., 2012; 2013]

Moving on to an example of joint-action learning, Figure 3.4a illustrates the learning behaviour of Q-Learning agents with and without PBRS under the default parameters given earlier. In these experiments, the agents without PBRS were unable to learn a suitable policy



(a) with Standard Parameters (b) with Alternative Parameters

Figure 3.4: Joint Action Q-Learners

but with PBRS they converged quickly to a good policy. This is the first example of agents learning significantly different behaviours because of PBRS, which is directly in contrast to the proven result for PBRS in single-agent problem domains.

Furthermore, even when the parameters for the joint-action Q-learning agents are adjusted so that the agents converge to a suitable policy without PBRS, agents with PBRS still learned a significantly better policy. Figure 3.4b illustrates an example of this with the settings $\alpha = 0.1, \gamma = 0.98, \lambda = 0$ and $\epsilon = 0.2/episode$.

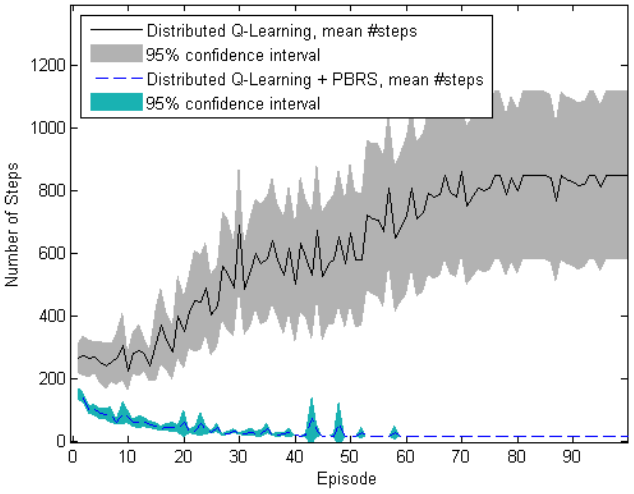
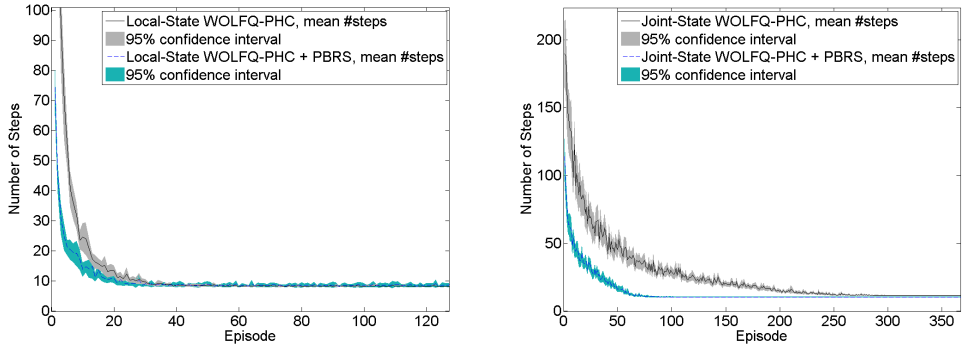


Figure 3.5: Distributed Q-Learning

Given that the SG used in these tests is cooperative, Distributed Q-Learning was also tested. This algorithm requires pessimistic initialisation of the value function and, therefore, all state-



(a) with Local State

(b) with Joint State

Figure 3.6: WoLFQ-PHC

action pairs were initially valued -10. Despite extensive parameter tuning I was unable to get Distributed Q-Learning to converge in this problem domain without PBRS. The results presented in Figure 3.5 are for the settings $\gamma = 0.95, \lambda = 0$ and $\epsilon = 0.8/timestep$. Distributed Q-Learning does not use the learning rate α. These results again show PBRS causing MARL to learn a different joint policy than the same agents learning without PBRS

Finally, to test a thoroughly alternative approach, experiments were run with WoLFQ-PHC using the local state of the agent alone (illustrated in Figure 3.6a) or the joint state of both agents (illustrated in Figure 3.6b). To do so the following additional parameter settings were used; deltaw $= 0.1$ and delta-ratio $= 4$.

In both sets of results, the more typical behaviour of quicker convergence to policies of equivalent performance when using PBRS is seen again. However, the common occurence of this pattern throughout most of the plausibility study is caused by the simplicity of the problem domain. If the complexity of the problem domain increases, as I will show in the next section, other algorithms will exhibit a similar pattern to the learning behaviour of Distributed Q-Learning and the joint-action learners in this study.

3.2 RoboCup Soccer Study

This section presents a study with multiple independent learners in a more complex and competitive MAS. This study is intended to exaggerate the effect of PBRS, helping to illustrate that the convergence to a different joint policy seen occasionally in the plausibility study is typical when multiple RL agents receive reward shaping. RoboCup is an international endeavor[1] which aims at providing an experimental framework in which various technologies can be integrated and evaluated. The overall research challenge is to create humanoid robots which would play and win against world champion humans. Since, the full game of soccer is complex, researchers

[1] See http://www.robocup.org/ for more information

developed several simulated environments which can be used to evaluate techniques for specific sub-problems. One such sub-problem is the KeepAway[2] task [Stone et al., 2006; 2005]. In this task (see Figure 3.7), N players (keepers) learn how to keep the ball when attacked by $N-1$ takers within a small, fixed area of the football pitch.

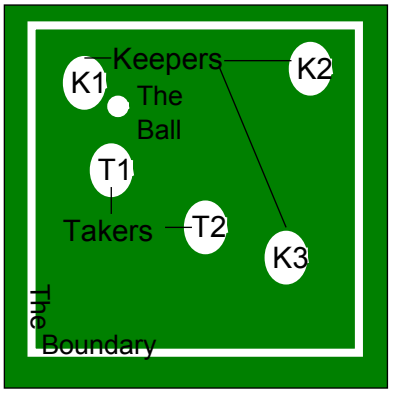


Figure 3.7: Snapshot of a 3 vs. 2 KeepAway game.

This task is multi-agent [Wooldridge, 2002] in its nature, with elements of both cooperation and competition. Overall, there are three types of high level behaviour in this task. First consider the agents trying to maintain possession of the ball; the keepers.

For keepers there are two distinct situations, either the keeper has possession of the ball or it does not. If not in possession of the ball, a keeper needs to move to a position convenient to receive the ball from the keeper that does have possession. The second behaviour is that of the keeper in possession of the ball, who must decide which other keeper to pass to or whether to maintain possession and wait for an appropriate time to pass.

The third and final behaviour is, that of the opposing team of agents trying to win possession of the ball; the takers. The takers must decide whether to close down the keeper in possession of the ball and attempt a tackle or to instead mark one of the keepers off-the-ball and attempt to intercept an incoming pass from the keeper on-the-ball.

3.2.1 Multi-Agent Learning in RoboCup Soccer

Previous work has attempted to learn the keepers' behaviour whilst in possession of the ball using RL whilst the takers and keepers off-the-ball (i.e. not in possession of it) adhere to a hand-coded policy [Stone et al., 2005; Devlin et al., 2009]. To make the problem of learning the keepers' behaviour more complex, both the behaviour of the keeper with the ball and the keepers without the ball can be updated simultaneously. This has been previously studied with a combined temporal difference and policy search solution [Kalyanakrishnan and Stone, 2010]. Alternatively, learning just the behaviour of keepers without the ball would also be a multi-agent

[2] See http://userweb.cs.utexas.edu/∼AustinVilla/sim/Keepaway/ for more information

learning problem, provided games of 3v2 or more players, as at all times at least two keepers would be off-the-ball.

Another multi-agent learning task possible using the KeepAway simulator is learning the behaviour of the takers. This task has previously [Min et al., 2008], and will be throughout this thesis, referred to as *TakeAway* to differentiate between experiments learning the takers' behaviour and those learning the keepers' behaviour. When learning the behaviour of the takers, the behaviour of the keepers is fixed to a hand-coded policy [Stone et al., 2005].

Previous attempts to learn the behaviour of takers proved relatively successful [Iscen and Erogul, 2008; Min et al., 2008] and were a useful resource when attempting to develop novel approaches. The basic learning taker uses SARSA with tile coding to decide the action of a taker every 15 cycles. This work emphasised that allowing a taker to decide an action on every cycle caused indecisiveness in the agent because the short time elapsed between decisions did not allow adequate time for the true benefit or cost of an action to be realised. In experiments allowing decisions to be made every cycle, takers oscillate between decisions causing poor performance.

There still remains large room for improvement in the development of a learning taker as the more challenging a taker can become, the more it will challenge researchers interested in learning the behaviours of keepers. The work presented here resulted in takers performing significantly better than all previous takers against the same opposing keepers in games with the same set up and in games more challenging to the takers.

As they contain elements of both competition and cooperation and are significantly more complex than the gridworld problem domain used in the previous section, these problem domains provide a suitable test bed for further testing the effect of PBRS on MARL.

3.2.2 KeepAway

This section provides more detail on the learning keepers and reward shaping techniques used. This investigation will again compare the performance of RL agents without reward shaping (the baseline learner) to agents that are using PBRS.

Baseline Learner

The baseline learning keeper for these experiments uses an existing hand-coded policy [Stone et al., 2005] when in possession of the ball and learns how to behave when not. More specifically, when not in possession of the ball the keeper must choose to move up, down, left, right or stay still based on the two dimensional pitch being divided into 25 equidistant points as illustrated in Figure 3.8. To learn when to perform these actions they use the SARSA algorithm with tile coding and ϵ-greedy action selection method, as in the original work on learning keepers in KeepAway [Stone et al., 2005].

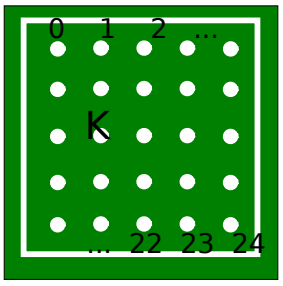


Figure 3.8: The 25 Possible Locations of Keepers when Off-The-Ball and 5 Example Actions Given a Keeper at K.

After each completed action the agent is rewarded according to how much time has elapsed since the action began. This way the keepers are encouraged to maximise the time they, as a team, maintain possession. It is important in these experiments to reward proportional to the time taken, as the actions in both KeepAway and TakeAway take differing lengths of time to complete. In effect, the true model of both problem domains is a SMDP. If the agents were instead rewarded proportional to the number of completed actions, the team would instead learn to perform lots of short actions and so would not learn the desired behaviour of keeping or winning possession.

Keeper	GetOpen
$dist(K_1, K_2)$	$dist(K_1, K_2)$
$dist(K_1, K_3)$	$dist(K_1, K_3)$
$dist(K_1, T_1)$	$dist(K_1, T_1)$
$dist(K_1, T_2)$	$dist(K_1, T_2)$
$argmin_{j \in 1,2}\{dist(K_2, T_j)\}$	$argmin_{j \in 1,2}\{dist(K_2, T_j)\}$
$argmin_{j \in 1,2}\{ang(K_2, K_1, T_j)\}$	$argmin_{j \in 1,2}\{ang(K_2, K_1, T_j)\}$
$argmin_{j \in 1,2}\{dist(K_3, T_j)\}$	$argmin_{j \in 1,2}\{dist(K_3, T_j)\}$
$argmin_{j \in 1,2}\{ang(K_3, K_1, T_j)\}$	$argmin_{j \in 1,2}\{ang(K_3, K_1, T_j)\}$
$dist(K_1, C)$	$dist(K_1, K)$
$dist(K_2, C)$	$argmin_{i,j \in \{2,3\} X \{1,2\}}\{ang(K_i, K_1, T_j)\}$
$dist(K_3, C)$	
$dist(T_1, C)$	
$dist(T_2, C)$	
$positionIndex(K)$	$positionIndex(K)$

Table 3.1: State Representations for Learning Keepers

To increase the number of learning problems evaluated, two different state representations were implemented. Both state representations are documented in Table 3.1. To clarify, K is the

agent itself, K_i is the i-th closest keeper to the ball, T_j is the j-th closest taker to the ball and C is the centre of the pitch. The method $ang(x, y, z)$ returns the angle with vertex y and edges yx and yz, $dist(x, y)$ returns the distance between x and y, $argmin_{j \in 1,2}$ returns 1 or 2 dependent on which returns the smallest value when input to the next method and $positionIndex(K)$ returns the index of the point the agent is closest to out of the 25 equally distributed points keepers can move to. The keeper state representation is based on the early work by Stone et al. [2005] and the GetOpen state representation is similar to the approach of Kalyanakrishnan and Stone [2010].

Separation-Based Reward Shaping

The separation-based reward shaping function is the first attempt to apply PBRS to a complex, MAS. Specifically, the domain knowledge applied states that keepers can improve their performance by spreading out. By following this principle, each keeper off-the-ball creates a unique angle for the keeper with the ball to pass along. Therefore, one taker cannot mark multiple keepers at once as they could if the keepers stuck together.

To encourage separation, the following potential function was used:

$$\Phi(s) = dist(K_1, K_2) + dist(K_1, K_3) \tag{3.2}$$

Experimental Design

The experiments undergone were performed in RoboCup Soccer Simulator v11.1.0 compiled against RoboCup Soccer Simulator Base Code v11.1.0. The KeepAway player code used was keepaway-player v0.6. Takers were based upon the hand-coded policy publicly available in this release and keepers were implemented by adding to the provided keeper our own code for RL and reward shaping.

For keepers both with and without reward shaping, the SARSA algorithm was used with the parameters; $\alpha = 0.125, \gamma = 1.0$ and $\epsilon = 0.01$. For function approximation a tile coding function with 14 or 11 groups (one for each feature, dependent on Keeper or GetOpen state representation respectively) of 32 overlapping single-dimension tilings was used. All keepers used one group per feature in the state representation. Angles were divided into ten degree intervals and distances into three meter intervals. Position indices were not approximated. These parameters were based on the settings used by Stone et al. [2005].

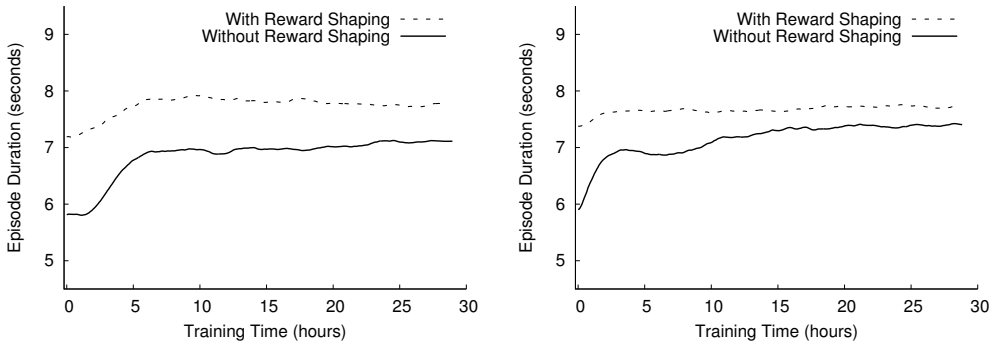
The base reward function, used by all agents, is a positive reward equal to the time passed between action choices with a large negative reward (-50) upon the start of a new episode to punish the receivers for losing possession. The supplemental reward from the shaping functions must be scaled to interact appropriately with this. A poor matching of scaling to the base reward function and state representation can reduce the gain in performance of a good heuristic [Grześ and Kudenko, 2009b; Grześ, 2010]. For these experiments, the value of separation was doubled before it was added to the basic reward function of the agents when receiving reward shaping.

This scaling factor was found through experimental testing. Therefore, it may not be the optimal setting. However, it is sufficient to show the improvement in performance the methods are capable of.

All experiments were repeated 30 times and performed on pitches of sizes 20×20 meters. All claims of significant differences are supported by two-tailed, two sample t-tests. Plots were made to ensure the assumption of normal distribution required for t-tests held for this data. The results provided in Section 3.2.2 illustrate the change in average episode length over all repeated experiments against time. Given that the keepers are learning in these experiments, the aim is to maximise the length of the average episode.

Results

As illustrated in Figure 3.9, keepers learning regardless of state representation were improved by using separation-based shaping.



(a) Keeper State Representation

(b) GetOpen State Representation

Figure 3.9: 3 Learning Keepers vs. 2 Hand-Coded Takers.

Specifically, Figure 3.9a shows shaped agents learning a significantly better ($p = 1 \times 10^{-8}$) performing joint policy than the baseline keepers, but, taking approximately the same time to do so. This result empirically demonstrates PBRS causing multiple independent learners to reach a different joint policy than when learning without PBRS.

Furthermore, Figure 3.9b demonstrates again agents learning a significantly better joint policy ($p = 0.07$) with PBRS than without. These results also show the more typical PBRS effect of reaching convergence quicker.

To conclude, these experiments on KeepAway support the concept that PBRS can also change what joint policy multiple independent learners will converge to.

The next section is a study on TakeAway, a distinct learning problem in the same environment. This is a significantly different task as it has opposing goals to the behaviour learnt in these experiments.

3.2.3 TakeAway

This section provides details on learning the takers' behaviour and three reward shaping techniques tested, including the first applications of potential-based advice in MARL.

Baseline Learner

The baseline learning taker combines the work of both previous papers [Iscen and Erogul, 2008; Min et al., 2008] on learning takers in KeepAway. As in both these papers, the takers can on each update choose either to tackle the keeper with the ball or mark a specific keeper. To tackle, the taker chases the ball and attempts to gain possession of the ball. To mark a keeper, the taker moves close to the keeper positioning itself between the ball and the keeper so as to gain possession if the ball is passed to that keeper.

To learn when to perform these actions, the agents use SARSA, ϵ-greedy action selection and tile coding, as Iscen and Erogul [2008] did. Additionally, they use the state representation and reward function, -1 for every cycle the episode continues to run and +10 for ending the episode, designed by Min et al. [2008]. Given the observations made by both papers regarding TakeAway agents' behaviours oscillating if allowed to make decisions too often, the agents only update their policy and make new action choices after every 15 cycles.

Image Label	Formal Definition
a	$dist(K_1, K_2)$
b	$dist(K_1, K_3)$
c	$dist(K_1, T_1)$
d	$dist(K_1, T_2)$
e	$dist(K_1, C)$
f	$dist(K_2, C)$
g	$dist(K_3, C)$
h	$dist(T_1, C)$
i	$dist(T_2, C)$
j	$min_{j \in 1,2} dist(K_{2-mid}, T_j)$
k	$min_{j \in 1,2} dist(K_{3-mid}, T_j)$
l	$min_{j \in 1,2} ang(K_2, K_1, T_j)$
m	$min_{j \in 1,2} ang(K_3, K_1, T_j)$

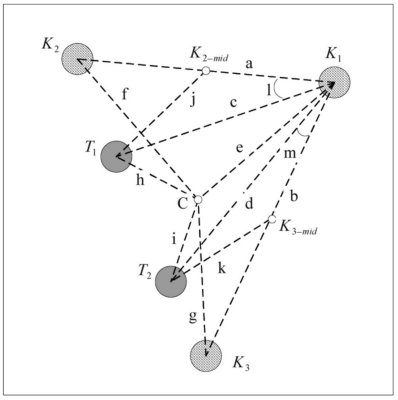


Figure 3.10: State Representation for Learning Takers [Min et al., 2008].

Figure 3.10 documents the features received by the takers in the chosen state representation. All reoccurring methods and symbols in the taker state representation represent the same meaning as previously introduced in the state representation descriptions of the baseline learning keepers. The one new symbol, K_{i-mid}, marks the mid-point between the keeper closest to the ball and the i-th closest keeper to the ball.

Separation-Based Reward Shaping

As with the keepers learning with reward shaping the learning takers can also benefit from increasing their separation. By following this principle, they are able to limit the passing options of the keepers and reduce the time the keepers maintain possession.This agent is intended to again show the benefits of applying PBRS to agents learning in a MAS and to add further to the growing evidence that the typical effect of PBRS on MARL is to both decrease the learning time and increase the performance of the final joint policy learnt.

Role-Based Advice

In experiments with the previous agent based upon a separation-based reward shaping, all taker agents will be homogeneous. A more interesting problem is that of heterogeneous agents, whereby different agents cooperating on the same team combine different skills to outperform their homogeneous counterparts [Balch, 1997].

Given the previous hypothesis, that takers sticking together is detrimental to performance, more complex prior domain knowledge can be incorporated stating that it is beneficial for one taker to tackle and another to fall back and mark. In effect, this new domain knowledge defines two roles; one of a tackling agent and one of a marking agent.

As this domain knowledge is action-based it becomes an implementation of potential-based advice [Wiewiora et al., 2003] and not simply PBRS. Therefore, additional requirements must be met if the addition of no preference to any one policy is to remain. As the knowledge is solely action-based the agents use look-back advice as recommended by Wiewiora et al. [2003]. Look back advice requires an on-policy learning algorithm and action selection based on relative differences in value, not absolute magnitude. Both of these conditions have been met by design of the baseline agent by the SARSA algorithm and ϵ-greedy policy.

Specifically, when considering an agent assuming the role of tackler, any state-action pair with a tackling action was given a potential of 2 and any state-action pair with a marking action a potential of 1. Combining these potentials and the formal definition of look back advice (Equation 2.10), Table 3.2 lists the additional rewards received by tackling agents where a is the agent's previous action, a' their new action, s the previous state, s' the current state, $F(s, a, s', a')$ the additional reward from look back advice and γ the same discount factor as the agent's update rule.

		$F(s, a, s', a')$		
a	a'	Description	Formula	Value
Mark	Tackle	Reward	$2 - \gamma^{-1}$	$+1$
Tackle	Mark	Punish	$1 - 2\gamma^{-1}$	-1

Table 3.2: Shaping Values of a Tackling Taker given $\gamma = 1$

By rewarding an agent when it switches from marking to tackling and punishing it when it changes the other way, the agent will be encouraged to tackle. Please see Listing 3.1 for clarification. Alternatively, giving any state-action pair with a tackling action a potential of 1 and any state-action pair with a marking action a potential of 2 reverses the reward and punishment. An agent receiving look back advice with these potentials would instead be encouraged to mark.

Listing 3.1: Tackler Heterogeneous Role Shaping Function

```
if not(a == a')
then if (a' == TacklingAction)
  then  F(s,a,s',a') = Reward
  else  F(s,a,s',a') = Punish
else F(s,a,s',a') = 0
```

These roles are not hard-coded, they do not limit the action choices available to the takers. Both takers can still choose either to mark or tackle and the ϵ-greedy policy will ensure agents explore the use of both action choices. Therefore, when it is necessary for the marking agent to tackle he will still make the correct decision and tackle, but in general it will choose to mark as the reward shaping function applied will make this appear more lucrative.

Combining Shaping Functions

Finally, one team of takers will incorporate both pieces of domain knowledge. This way the takers can be encouraged to take roles but also consider the benefit of separating.

Formally, the PBRS function changes from Equation 2.6, to:

$$F(s, a, s', a') = \tau_1 F_1(s, a, s', a') + \tau_2 F_2(s, a, s', a') \tag{3.3}$$

where F_1 and F_2 are the shaping functions for role-based advice and separation-based shaping respectively and τ_1 and τ_2 are two separate scaling factors.

In early empirical tests, scaling variables set to emphasise the role-based advice function showed the best performance. Therefore, the combined shaping agent will also emphasise the role-based advice function. This agent will still include the separation-based reward shaping function but by scaling the function appropriately it will have less of an impact on the resulting behaviour than the encouragement to take up a specific role.

Experimental Design

The experiments undergone were again performed in RoboCup Soccer Simulator v11.1.0 compiled against RoboCup Soccer Simulator Base Code v11.1.0. The KeepAway player code used was keepaway-player v0.6. This time the keepers were based upon the hand-coded policy publicly available in this release and takers were implemented by adding to the provided taker the necessary code for RL and PBRS.

For takers both with and without reward shaping, SARSA was used with the parameters; $\alpha = 0.125, \gamma = 1.0$ and $\epsilon = 0.01$. For function approximation a tile coding function with 13 groups (one for each feature in the state representation) of 32 overlapping single-dimension tilings was used. All takers used one group per each feature in the observation and split angles into ten degree intervals and distances into three meter intervals. These parameters were again based on those used by Stone et al. [2005].

For the separation-based reward shaping the difference in separation was doubled before added to the basic reward function, and the role-based advice was scaled by 5. Given that $\gamma = 1.0$ this effectively means agents with role-based advice are either rewarded or penalised by 5 for changing their action from marking to tackling and vice versa.

As stated earlier, when combining shaping functions, it was more beneficial to emphasise the heterogeneous role knowledge and so for changing their action these takers were either rewarded or penalised by 10 and for separation the change in distances were simply added. Given that role-based advice is to be the first shaping function in the combination, formalised in Equation 3.3, this corresponds to a τ_1 of 10 and a τ_2 of 1.

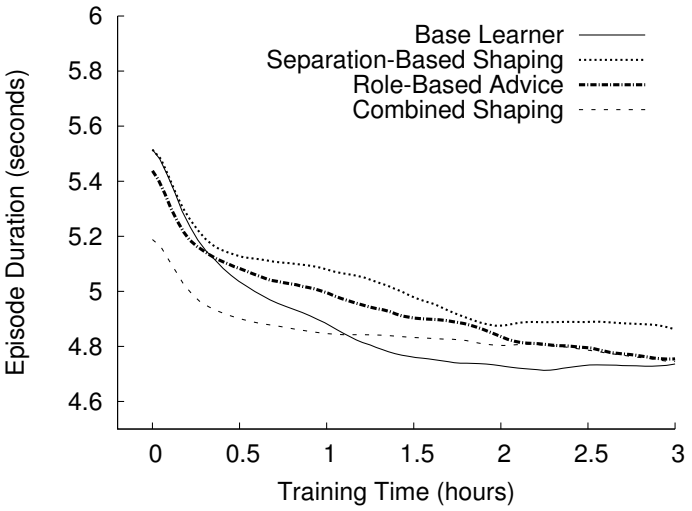
Experiments were performed on pitches of sizes 20×20, 30×30, 40×40, and 50×50 meters. These values were chosen to show the performance of these takers in similar contexts to previous work on learning the behaviour of takers and also in more complex problem domains.

Experiments with each combination of pitch size and reward shaping function were repeated 30 times. All claims of significant differences are supported by two-tailed, two sample t-tests. Plots were made to ensure the assumption of normal distribution required for t-tests held for this data. Given that there are now multiple types of agents being compared, it may have been more appropriate to use ANOVA for these experiments. The results provided illustrate the change in average episode length over all repeated experiments against time. Given that the takers are now learning, the aim is now to minimise the length of the average episode.

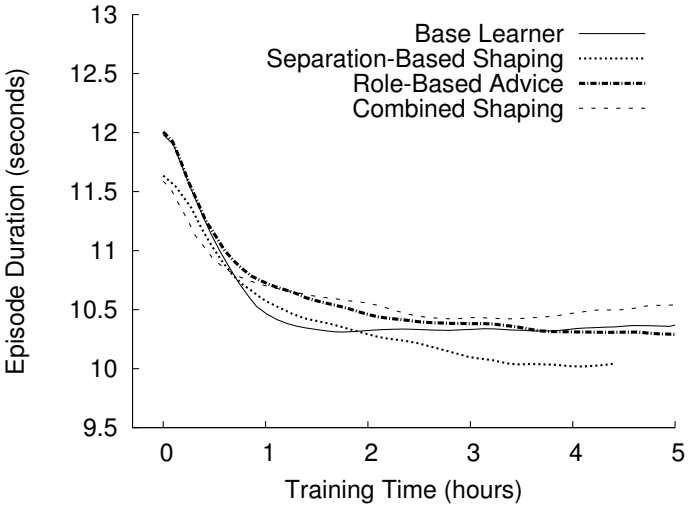
Results

In experiments on the simplest domains, all agents learnt good policies quickly with no significant difference ($p > 0.2$) in performance. For both pitches of size 20x20 and 30x30, illustrated in Figures 3.11a and 3.11b, it is important to consider that both axes represent small changes in time in their given dimension and the differences between agents is both brief and insignificantly small (only 0.4 seconds for pitch size 20x20). Therefore, TakeAway at pitches of this size is too simple to gain much benefit from reward shaping.

In problem domains where RL alone can quickly learn a policy of good performance, the additional work of designing a heuristic and implementing reward shaping, however simple that may be, is unnecessary. These methods are more beneficial in complex problem domains where RL alone takes a long time to converge and has a large difference in performance between the initial policy and the final policy converged to.



(a) 20x20

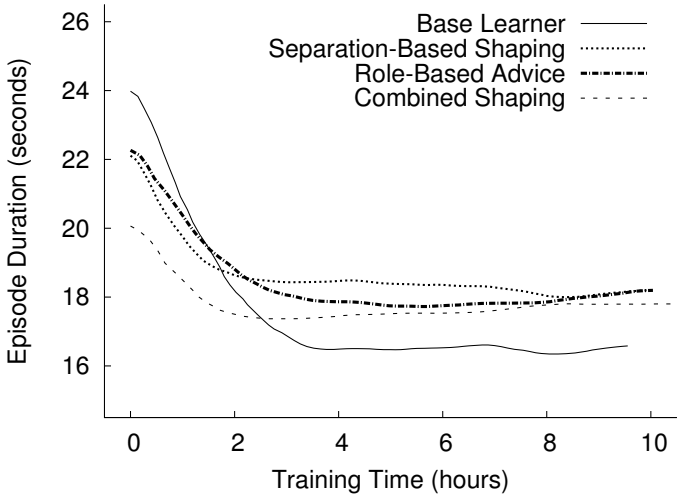


(b) 30x30

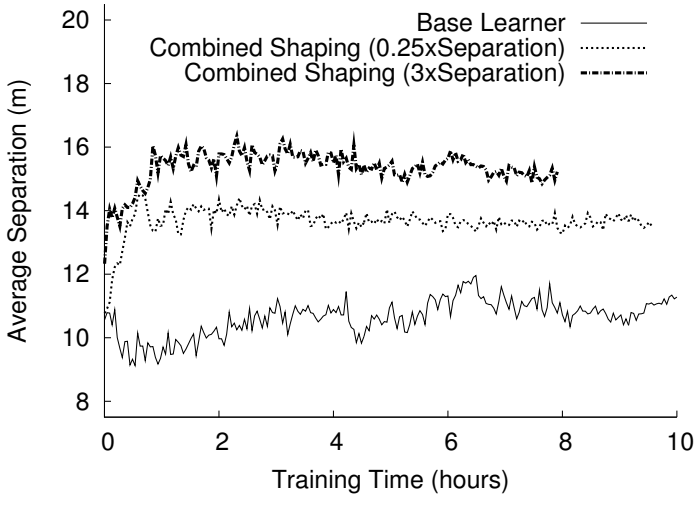
Figure 3.11: 2 Learning Takers vs. 3 Hand-Coded Keepers.

These results, however, have been included for comparison to previous work on learning takers. The baseline agent learns a slightly better joint policy than the best performing, learning taker from the existing published attempts [Min et al., 2008] which is quoted as converging on average to win possession in 5.8 seconds in games of 3v2 on pitches of size 20x20. All learning takers, both the pre-existing and this baseline learner, outperform the standard hand-coded takers defined by Stone et al. [2005] that perform consistently around 15 seconds. Therefore, the baseline learner developed is both a suitable and highly competitive test agent to compare the approaches with reward shaping to.

At a pitch size of 40x40 the problem appears to become sufficiently difficult, with the baseline learner unable to converge quickly as seen in Figure 3.12a. With this level of difficulty a clear difference in agents is now evident. All shaped agents immediately benefit from the additional domain knowledge with statistically significant differences ($p < 0.05$) in initial performance to the baseline takers.



(a) Overall Performance



(b) Average Separation

Figure 3.12: 2 Learning Takers vs. 3 Hand-Coded Keepers at 40x40.

During the early episodes of training, all shaped agents improve performance at a visually similar rate to the baseline learner and so maintain their positive difference in performance. After an hour of training the learning of takers using reward shaping begins to slow and the average

performance of the baseline learner starts to catch up. At two hours of training, the performance of all agents is equivalent ($p = 0.9$) but by 8 hours the baseline learner significantly outperforms the shaped agents ($p < 0.005$).

Convergence to a different joint-policy, at 40x40, has caused the difference in performance between agents with shaping and the baseline learner. The heuristics used are poorly matched to the other settings of variables at 40x40. The agents still benefit from directed exploration, by initially improving performance quicker than the baseline learner, but suffer as their final policy is different and represents a behaviour of lower performance. It would be an implementation decision to prioritise either the reduced training time of the shaped agents or the higher final performance of the baseline takers.

It is important to remember that whilst PBRS in single-agent can reduce learning time, it can also increase it if given a bad heuristic. Similarly, this result shows that PBRS for MARL can both cause agents to learn a better or a worse final joint policy.

Figure 3.12b[3] shows that increasing the scale of the separation shaping function causes agents to separate further on average. This is further empirical evidence that agents receiving reward shaping may learn different joint policies when in a common environment.

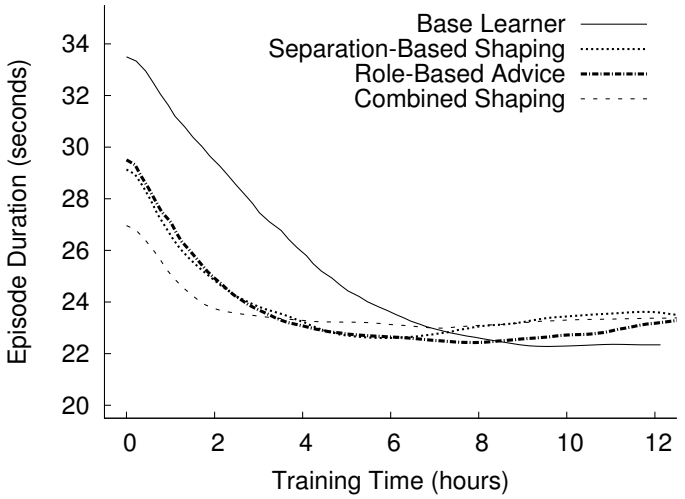
The results on pitches of 50x50, as illustrated by Figure 3.13, show a problem domain more suitable to the use of reward shaping. As previously seen in the change from pitch sizes of 30x30 to 40x40, there is a rise in difficulty when increasing the pitch size from 40x40 to 50x50. Given the yet again higher difficulty, a more significant improvement can and has been witnessed.

Firstly, there is now a highly significant difference ($p < 4 \times 10^{-8}$) between the initial performance of all shaped takers and the baseline learner. The most significant being between the baseline and takers receiving the combined advice of both heuristics ($p = 2 \times 10^{-17}$).

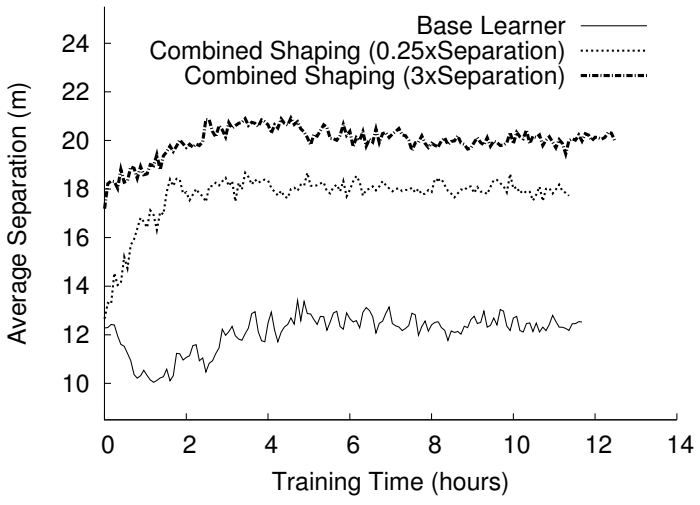
This gain in performance remains roughly constant throughout the first 4 hours of training. It then begins to shrink but still outperforms the baseline learner for up to approximately 8 hours. Even after the first 8 hours of training, the baseline learner can only match the performance of the novel approaches and never significantly outperforms any of them ($p > 0.1$ after 11 hours).

Finally, the agents solely encouraged to take heterogeneous roles did adhere to the encouragement and after convergence were seen to almost exclusively stick to their assigned roles. They did not, however, follow their assigned roles blindly and did deviate occasionally from them in states where they learnt it to be beneficial. By using RL with potential-based advice to encourage roles, these deviations from the encouraged role were possible whereas an agent with enforced roles would not provide such flexibility.

[3]The two combined agents documented in this figure and Figure 3.13b represent the best tuned solutions found for 40x40 and 50x50. It is worthwhile to note that changes in environment parameters will often require a change in scaling parameters when combining reward shaping functions. In this example, scaling by 0.25 at 50x50 and by 3 at 40x40 gave the best performance found.



(a) Overall Performance

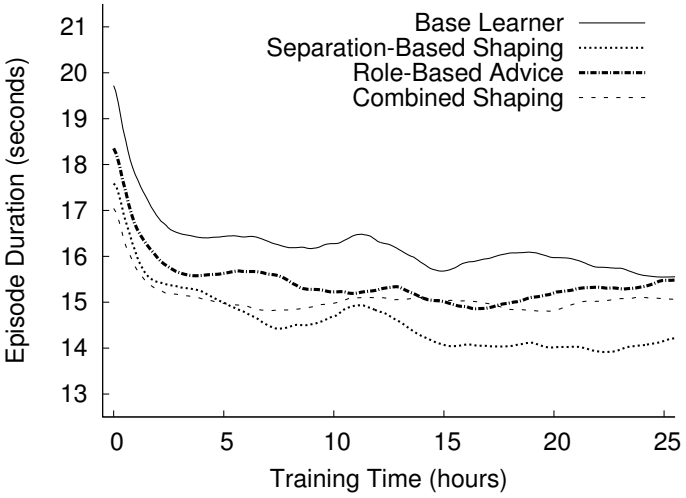


(b) Average Separation

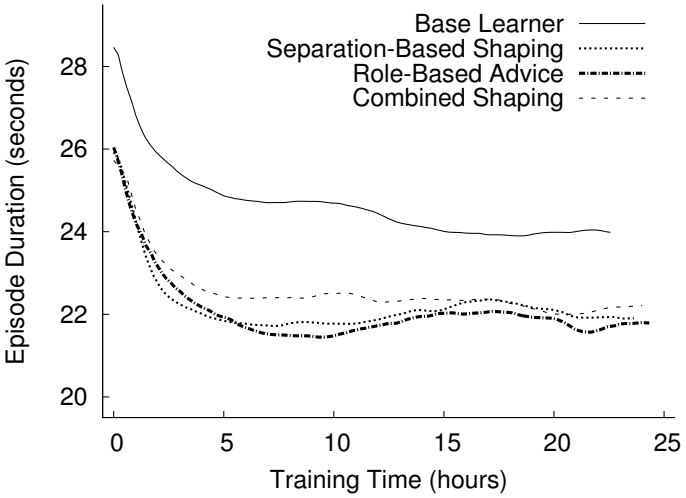
Figure 3.13: 2 Learning Takers vs. 3 Hand-Coded Keepers at 50x50.

Scaling Up

To further challenge the learning takers, more agents can be deployed. By adding agents to the learning team, cooperation becomes harder. However, to maintain the game's dynamics, keepers must also be added. Therefore, this section discusses games of three takers versus four keepers (3v4) and four takers versus five keepers (4v5).



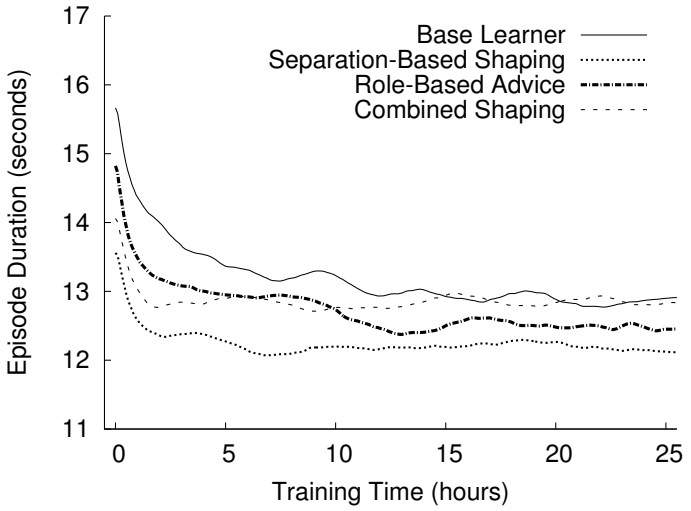
(a) Overall Performance at 40x40



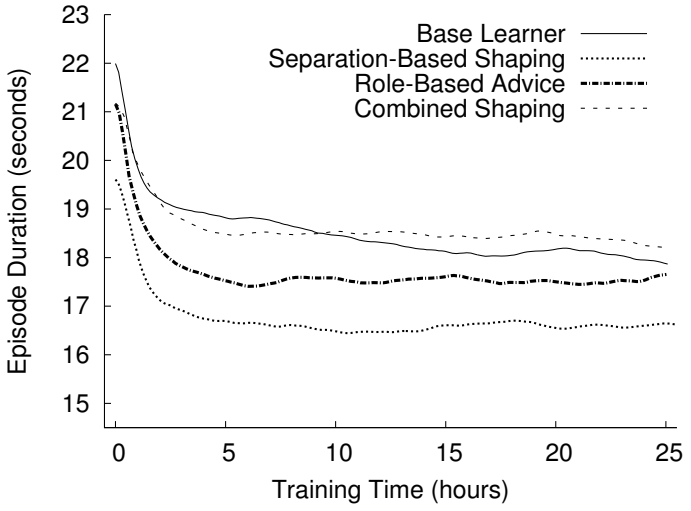
(b) Overall Performance at 50x50

Figure 3.14: 3 Learning Takers vs. 4 Hand-Coded Keepers.

The first results of 3v4 at 40x40, illustrated in Figure 3.14a, show yet again PBRS altering exploration sufficiently to benefit final performance. In this specific problem domain, the results



(a) Overall Performance at 40x40



(b) Overall Performance at 50x50

Figure 3.15: 4 Learning Takers vs. 5 Hand-Coded Keepers.

conclude that separation-based shaping is a more suitable heuristic than role-based advice. This is apparent because the separation-based shaped agents' joint policy represents the most significantly better performance than the baseline taker ($p = 3 \times 10^{-8}$).

At pitch sizes of 50x50, illustrated in Figure 3.14b, all shaped/advised agents significantly ($p < 0.03$) outperform the baseline agent in 3v4. Agents receiving separation-based shaping are again the best solution for 3v4, as they learn the policy on average two training hours quicker than the nearest competitor.

Finally, the number of agents was increased up to 4v5. At 40x40 with 4v5, as illustrated

in Figure 3.15a, all advised agents, both role-based and combined-shaped, learn joint policies equivalent to the joint policy learnt by the baseline agent ($p > 0.1$) but do so quicker due to directed exploration. Again, for this problem domain the separation-based shaped agents are the superior solution as they both learn quicker by directed exploration and also learn a joint policy representative of a performance significantly better than all other agents ($p = 0.007$).

At 50x50, illustrated in Figure 3.15b, the difference is further exaggerated and separation-based shaping is yet again clearly the dominant method, learning the quickest and to a highly statistically significant better performance than any other team of takers ($p = 3 \times 10^{-5}$). The combined shaped takers again match ($p = 0.5$) the performance of the baseline learner but do so with less training time, showing they maintain some benefits in this problem domain. Meanwhile, the role-based heuristic regains some suitability to this problem domain by slightly outperforming the baseline takers ($p = 0.09$).

Overall, the results from increasing the number of agents have shown a better ability to scale for the separation-based shaping than the role-based advice or combined shaping. However, this is a feature of the particular heuristics and not PBRS compared to potential-based advice. The reason being that the roles used were designed for teams of two. With two takers, one tackler and one marker is intuitive, however with three takers, one tackler and two markers is only intuitive if each marker sticks to a given keeper for a period of time. As it was coded the takers were only encouraged to pick a marking action and changes between which marking action were not considered. Therefore, marking takers oscillate between marking one agent and another frequently making it harder to coordinate and subsequently breaking the benefit of the roles. This also detrimentally affected the combined shaping agents, whose exploration was modified by both heuristics, unfortunately with the role-based advice commonly having a larger effect than the more beneficial knowledge of separation-based shaping.

3.3 Conclusion

In conclusion, this chapter has demonstrated the applicability and benefits of using potential-based reward shaping and advice in MAS. Specifically, they can affect both the time taken to learn and/or the performance of the final joint policy.

Although the specific reward shaping functions implemented have used domain specific knowledge the types of domain knowledge represented are generally applicable. For example, the potential function used in the plausibility study can be used in any environment where a distance metric to a desired goal is applicable.

Furthermore, from the RoboCup study, the knowledge that keepers and takers should try to stay separate is an example of knowledge regarding how agents should maintain states relative to each other. Maintaining a state relative to either team-mates or opponents is a common type of knowledge applicable in many MAS. For example, it has been shown in the predator/prey problem domain that it is beneficial for predators to consider the relative location of its supporting

predator to aid coordination [Tan, 1993]. Similarly, having one tackler and one marker is specific to takers in TakeAway but the knowledge that agents should specialise into roles is common in MAS. For example, again in the predator/prey problem domain, it has been shown that it is beneficial to have one predator take a hunting role and another take a scouting role [Tan, 1993].

By empirically demonstrating PBRS in three distinctly different learning tasks and with a wide range of algorithms, this chapter provides strong supporting evidence that these results will occur when the methods are added to any existing MARL solution.

These results, whilst advocating the use of PBRS in MARL, raise the question why has the typical effect changed? Agents learning different final policies is not compliant with the single-agent proof of policy invariance. The next chapter will explore, in theory, what changes when applying PBRS to multiple agents.

CHAPTER 4

Multi-Agent, Potential-Based Reward Shaping: In Theory

To discuss the theoretical implications of using PBRS in MARL, I will begin by considering the differences between single-agent and multi-agent problem domain representations. Stochastic Games (SG), unlike MDPs, share amongst all agents a common transition function and common states but neither of these are affected by shaping the reward function of one or more of the agents. Although the agents may change their own policy and alter their own exploration path due to the additional potential-based reward, this does not change the dynamics (transition function or states) of the environment, nor the set of actions the agent can take.

In fact, the only elements of a SG to change when one or more agent implements PBRS are the individual reward functions of those agents. If, as I will later prove to be true, these alterations to the individual reward functions do not change the best response policy of a shaped agent given a fixed set of policies followed by all other agents, the Nash equilibria of the underlying SG remain constant regardless of how many agents are using PBRS. This argument will be supported by first showing, in the following sub-section, that PBRS is still equivalent to Q-table initialisation in the multi-agent case. Both of these findings, as I will discuss in Section 4.3, have implications for the eventual policy that will be converged upon.

This Chapter will also cover how the potential function can change online whilst still maintaining the same guarantees, the general effect of PBRS and some implementation details required to maintain the theoretical guarantees of PBRS when applying the method to problem domains with finite episodes.

4.1 Equivalence to Q-Table Initialisation

The proof of Wiewiora [2003] of the equivalence of PBRS and Q-value initialisation was published in the context of single agent problem domains but also holds, as I will show, for problem domains with multiple agents.

From Wiewiora [2003] I quote:

> *Theorem 1* Given the same sequence of experiences during learning, $\Delta Q(s,a)$ always equals $\Delta Q'(s,a)$.

where $Q(s,a)$ is the modelled value function of an agent learning with PBRS, $Q'(s,a)$ is the modeled value function of an agent learning with Q-value initialisation, $\Delta Q(s,a)$ and $\Delta Q'(s,a)$ are how much the modeled value of action a in state s changes during the sequence of experiences for the agent receiving PBRS and the agent with Q-value initialisation respectively.

The original proof uses a fixed sequence of experiences for both agents. The theory can be extended to multiple agents simply by extending the definition of the sequence experienced from the 4-tuple $\langle s,a,r,s'\rangle$ to the $2n+2$-tuple $\langle s,a_1,a_2,...,a_n,r_1,r_2,...,r_n,s'\rangle$. Using the extended sequence and the inductive proof from Wiewiora [2003] the following proves that Theorem 1 holds also for MARL.

Proof By Induction

Consider any arbitrary agent i from the set of all agents. As before, $Q(s,a)$ is the modeled value function when the agent is learning with PBRS and $Q'(s,a)$ is the modelled value function had the same agent learnt without reward shaping but with Q-value initialisation. The former agent will be referred to as L and the latter as L'.

Agent L will update its Q-values by the rule[1] :

$$Q_i(s,a) \leftarrow Q_i(s,a) + \underbrace{\alpha\left(r_i + F(s,s') + \gamma \max_{a'} Q_i(s',a') - Q_i(s,a)\right)}_{\delta Q_i(s,a)} \tag{4.1}$$

where $F(s,s')$ is the PBRS function and $\delta Q_i(s,a)$ is the amount (scaled by α) that the Q value will be updated by. The current Q-values of Agent L can be represented formally as the initial value plus the change since:

$$Q_i(s,a) = Q_i^0(s,a) + \Delta Q_i(s,a) \tag{4.2}$$

where $Q_i^0(s,a)$ is agent i's initial Q-value of state-action pair (s,a).

[1] This proof and all other proofs in this chapter assume the use of Q-learning for the agents' update rules. Similar proofs can be produced using the same working for other RL algorithms. For further discussion on this topic, please see Section 4.7

Similarly agent L' updates its Q-values by the rule:

$$Q'_i(s,a) \leftarrow Q'_i(s,a) + \alpha \underbrace{(r_i + \gamma \max_{a'} Q'_i(s',a') - Q'_i(s,a))}_{\delta Q'_i(s,a)} \tag{4.3}$$

And its current Q-values can be represented formally as:

$$Q'_i(s,a) = Q^0_i(s,a) + \Phi(s) + \Delta Q'_i(s,a) \tag{4.4}$$

where $\Phi(s)$ is the potential for state s.

Base Case

Before either agent experiences anything, the Q-tables of L and L' are both their respective initial values, and therefore both ΔQ_i and $\Delta Q'_i$ are uniformly zero.

Inductive Case

Assuming $\Delta Q_i = \Delta Q'_i$, both L and L' will be updated by the same amount in response to experience $\langle s, a_1, a_2, ..., a_n, r_1, r_2, ..., r_n, s' \rangle$.

First consider the update performed by L:

$$\begin{aligned}
\delta Q_i(s,a) &= r_i + F(s,s') + \gamma \max_{a'} Q_i(s',a') - Q_i(s,a) \\
&= r_i + \gamma\Phi(s') - \Phi(s) + \gamma \max_{a'}(Q^0_i(s',a') + \Delta Q_i(s',a')) - Q^0_i(s,a) - \Delta Q_i(s,a)
\end{aligned} \tag{4.5}$$

Now consider the update performed by L':

$$\begin{aligned}
\delta Q'_i(s,a) &= r_i + \gamma \max_{a'} Q'_i(s',a') - Q'_i(s,a) \\
&= r_i + \gamma \max_{a'}(Q^0_i(s',a') + \Phi(s') + \Delta Q'(s'a')) - Q^0_i(s,a) - \Phi(s) - \Delta Q'(s,a) \\
&= r_i + \gamma \max_{a'}(Q^0_i(s',a') + \Phi(s') + \Delta Q(s'a')) - Q^0_i(s,a) - \Phi(s) - \Delta Q(s,a) \\
&= r_i + \gamma\Phi(s') - \Phi(s) + \gamma \max_{a'}(Q^0_i(s',a') + \Delta Q_i(s',a')) - Q^0_i(s,a) - \Delta Q_i(s,a) \\
&= \delta Q_i(s,a)
\end{aligned} \tag{4.6}$$

Therefore, the Q-tables of both L and L' are both updated by the same value and so ΔQ_i and $\Delta Q'_i$ remain equal.

□

Given that Theorem 1 of Wiewiora [2003] holds for the multi-agent context then so too does Theorem 2, again quoted from Wiewiora [2003]:

> *Theorem 2* If L and L' have learned on the same sequence of experiences and use an advantage-based policy, they will have an identical probability distribution for their next action.

where an advantage-based policy is one that chooses actions based not on the absolute magnitude of the Q-values but on their relative differences within the current state. Examples of advantage-based policies include greedy, ϵ-greedy and Boltzmann soft-max.

This is immediately apparent when considering both $\Delta Q_i = \Delta Q'_i$ from Theorem 1 and Equations 4.2 and 4.4. As the difference between the Q-values of agent L and agent L' are the potential of the state, the difference is consistent across all actions in any given state. Therefore, the actions maintain the same relative differences allowing an advantage-based policy to make the same action decisions.

Effectively, at any time in learning L and L' will behave the same way (make the same decisions with the same probabilities). To conclude, whether an agent is shaped or initialised it will have the same effect on all other agents in the environment, the learning dynamics are not changed by using one method or the other and the agents as a collective whole will converge or not upon the same joint policy regardless of whether the agent was shaped or initialised.

Finally, although the proof here was written specifically for Q-learning, this was simply in keeping with the original work of Wiewiora [2003]. In single-agent problem domains the equivalence of Q-table initialisation and PBRS can be proven also in SARSA and other temporal difference algorithms [Wiewiora, 2003]. Similarly, proofs for the multi-agent case are also possible for other algorithms.

4.2 Consistent Nash Equilibria

As already established, MARL agents will typically learn a joint policy representative of a Nash equilibrium. The typical concern of modifying a reward function is that the original goals of the agent will be altered. Ng et al. [1999] showed previously that in the single-agent context, the optimum policy was unchanged by the introduction of reward shaping provided the function was potential-based. To extend this to MARL, it must be considered whether implementing the same reward shaping in one or more agents in a SG will alter its points of equilibrium.

Formally, recall from Chapter 2, a joint policy π^{NE} is a Nash equilibrium provided:

$$\forall i \in 1 \dots n, \pi_i \in \Pi_i | R_i(\pi_i^{NE} \cup \pi_{-i}^{NE}) \geq R_i(\pi_i \cup \pi_{-i}^{NE}) \tag{4.7}$$

where n is the number of agents, Π_i is the set of all possible policies of agent i, R_i is the reward function for agent i, π_i^{NE} is a specific policy of agent i and π_{-i}^{NE} is the joint policy of

all agents except agent i following their own fixed specific policy. If the inequality holds for all agents, the joint policy π^{NE} of each agent following its policy π_i^{NE} is a Nash equilibrium.

Now consider any arbitrary agent i from the set of all agents. For the inequality above to hold for agent i, we must consider the set Π_i^{NE} of all joint policies consisting of each possible policy of agent i combined with π_{-i}^{NE}. Formally, this set contains:

$$\{(\pi_i \cup \pi_{-i}^{NE}) | \forall \pi_i \in \Pi_i\} \tag{4.8}$$

Each fixed joint policy in the set Π_i^{NE} will generate a fixed infinite sequence of experiences when followed consistently from the current state s_0 of the form:

$$\bar{s} = s_0, a_{0,0}, a_{0,1}, \ldots, a_{0,n}, r_{0,0}, r_{0,1}, \ldots, r_{0,n}, s_1, \ldots \tag{4.9}$$

where s_j is the state at time j, $a_{j,i}$ is the action taken by agent i at time j and $r_{j,i}$ is the reward received by agent i at time j.

Then using a similar proof as Asmuth et al. [2008], I will show below that the difference of the return received by agent i when following any arbitrary fixed sequence with or without PBRS is the potential of the state s_0.

Proof

The return for agent i when experiencing sequence $\bar{s}$ without shaping is:

$$U_i(\bar{s}) = \sum_{j=0}^{\infty} \gamma^j r_{j,i} \tag{4.10}$$

Now consider the same agent but with a reward function modified by PBRS. The return of the shaped agent experiencing the same sequence $\bar{s}$ is:

$$\begin{aligned}
U_{i,\Phi}(\bar{s}) &= \sum_{j=0}^{\infty} \gamma^j (r_{j,i} + F(s_j, s_{j+1})) \\
&= \sum_{j=0}^{\infty} \gamma^j (r_{j,i} + \gamma\Phi(s_{j+1}) - \Phi(s_j)) \\
&= \sum_{j=0}^{\infty} \gamma^j r_{j,i} + \sum_{j=0}^{\infty} \gamma^{j+1}\Phi(s_{j+1}) - \sum_{j=0}^{\infty} \gamma^j \Phi(s_j) \\
&= U_i(\bar{s}) + \sum_{j=1}^{\infty} \gamma^j \Phi(s_j) - \sum_{j=1}^{\infty} \gamma^j \Phi(s_j) - \Phi(s_0) \\
&= U_i(\bar{s}) - \Phi(s_0)
\end{aligned} \tag{4.11}$$

□

Therefore, any policy that previously maintained the inequality of Equation 4.7 will still maintain the inequality. Formally, I conclude:

$$\forall \pi_i \in \Pi_i | \quad (R_{i,\Phi}(\pi_i^{NE} \cup \pi_{-i}^{NE}) \geq R_{i,\Phi}(\pi_i \cup \pi_{-i}^{NE})) \leftrightarrow (R_i(\pi_i^{NE} \cup \pi_{-i}^{NE}) \geq R_i(\pi_i \cup \pi_{-i}^{NE})) \tag{4.12}$$

where $R_{i,\Phi}$ is the reward function of agent i when receiving both the environmental reward and PBRS.

As implementing reward shaping only affects the reward function of that agent, the remaining agents will also still maintain the same policies as part of the Nash equilibria. Whether the group will converge to this point depends on the learning algorithm used and is outside of this proof. However, it suffices to say that regardless of how many agents in the MAS are or are not implementing PBRS the points of equilibrium will remain constant.

4.3 Convergence Guarantees

In Section 4.1 I showed that an agent in a MAS receiving PBRS is equivalent to one whose Q-table was initialised with each state s set to the potential $\Phi(s)$ of that state. However, the implications of this proof in a MAS extend past showing that two methods of introducing domain knowledge are equivalent. Instead, it is worth considering the results of Wellman and Hu [1998], in which they showed that the joint policy converged upon in a learning MAS was highly sensitive to initial belief. This clearly applies directly to Q-table initialisation, where the initial values directly represent some initial belief, and therefore, given that the equivalence to initialisation is proven, also applies to PBRS. This can be reasoned intuitively by considering the following.

The MDP of an agent deployed in a common environment with other learning agents does not hold the Markov property as the transition probabilities are subject to change with the unseen but changing policies of the other agents. Therefore, the convergence to optimal policy guarantees of Q-learning do not hold. This has been demonstrated empirically in MARL applications with multiple independent Q-learners converging to sub-optimal joint policies [Babes et al., 2008].

Shaping alters the path of exploration an agent takes. In single-agent RL, as convergence to the optimal policy is guaranteed, this only affects the time taken to reach convergence. If a good heuristic is used, the time will be reduced as the number of sub-optimal actions taken will be reduced. Alternatively, if a bad heuristic is used, the agent will take longer to converge to the optimal policy.

However, the concept of an optimal policy in MARL is not as clear. Typically the goal is to learn a Nash equilibrium, but this does not necessarily identify a single goal as most applications will have multiple equilibria. With multiple agents in the same environment, altering the exploration of one will change the experiences of all agents [Kapetanakis and Kudenko,

2002; 2004]. The change in actions chosen by even just one agent now receiving PBRS will result in different state transitions occuring. The agents will then explore different areas of the joint policy space and, with multiple points of equilibrium possible, may converge to a different equilibrium then had the agent not received the reward shaping and subsequently not have altered its individual exploration path.

Therefore, in multi-agent problem domains, without the guarantee of convergence to a single optimum goal, shaping can lead to convergence on a different joint policy. When shaping one or more agents in an environment with multiple learning agents, a good heuristic will encourage higher global utility similar to how in single-agent problem domains the use was preferably to reduce the time taken to converge. Unfortunately, the techniques can also have a detrimental effect encouraging miscoordination and/or lead the agents to converge on a less beneficial joint policy by directing the agents away from frequently, or possibly ever, experiencing the equilibrium reached by non-shaped agents and instead trapping them in a sub-optimal point of equilibrium.

4.4 Dynamic Potential Functions

PBRS is typically implemented bespoke for each new environment using domain-specific heuristic knowledge [Babes et al., 2008; Devlin et al., 2011; Randløv and Alstrom, 1998] but some attempts have been made to automate [Grześ and Kudenko, 2008; Grześ and Kudenko, 2010; Laud, 2004; Marthi, 2007] the encoding of knowledge into a potential function.

All of these existing methods alter the potential of states online whilst the agent is learning. However, neither the existing single-agent proof of policy invariance [Ng et al., 1999] nor the multi-agent theoretical results in the previous section considered such dynamic shaping.

Furthermore, the opinion has been published that the potential function must converge before the agent can [Laud, 2004]. In the majority of implementations this approach has been applied [Grześ and Kudenko, 2010; Laud, 2004; Marthi, 2007] but in other implementations stability is never guaranteed [Grześ and Kudenko, 2008]. In this single agent example, despite common intuition, the agent was still seen to converge to an optimal policy.

Therefore, contrary to existing opinion it must be possible for an agent's policy to converge despite a continually changing reward transformation.

In this section I will cover the implications of a dynamic potential function on the three most important existing proofs in PBRS. Specifically, in subsection 4.4.1 I address the theoretical guarantees of policy invariance in single-agent problem domains [Ng et al., 1999] and consistent Nash equilibria in multi-agent problem domains [Devlin and Kudenko, 2011]. Later, in subsection 4.4.2, I will address Wiewiora's proof of equivalence to Q-table initialisation [Wiewiora, 2003].

4.4.1 Policy Invariance and Consistent Nash Equilibria

To extend PBRS to allow for a dynamic potential function, the Equation 2.6 must be extended to include time as a parameter of the potential function Φ. Informally, if the difference in potential is calculated from the potentials of the states at the time they were visited the guarantees of policy invariance or consistent Nash equilibria remain. Formally:

$$F(s,t,s',t') = \gamma\Phi(s',t') - \Phi(s,t) \tag{4.13}$$

where t is the time the agent arrived at previous state s and t' is the current time when arriving at the current state s' (i.e. $t < t'$).

To prove policy invariance in the single-agent case and consistent Nash equilibria in the multi-agent case it suffices to show that the return a shaped agent will receive for following a fixed sequence of states and actions is equal to the return the non-shaped agent would receive when following the same sequence minus the potential of the first state in the sequence [Asmuth et al., 2008; Devlin and Kudenko, 2011].

Therefore, consider the return U_i for any arbitrary agent i when experiencing sequence $\bar{s}$ in a discounted framework without shaping. Formally:

$$U_i(\bar{s}) = \sum_{j=0}^{\infty} \gamma^j r_{j,i} \tag{4.14}$$

where $r_{j,i}$ is the reward received at time j by agent i from the environment.

Given this definition of return, the true Q-values can be defined formally by:

$$Q_i^*(s,a) = \sum_{\bar{s}} Pr(\bar{s}|s,a) U_i(\bar{s}) \tag{4.15}$$

Now consider the same agent but with a reward function modified by adding a dynamic potential-based reward function of the form given in Equation 4.13. The return of the shaped agent $U_{i,\Phi}$ experiencing the same sequence $\bar{s}$ is:

$$\begin{aligned}
U_{i,\Phi}(\bar{s}) &= \sum_{j=0}^{\infty} \gamma^j (r_{j,i} + F(s_j,t_j,s_{j+1},t_{j+1})) \\
&= \sum_{j=0}^{\infty} \gamma^j (r_{j,i} + \gamma\Phi(s_{j+1},t_{j+1}) - \Phi(s_j,t_j)) \\
&= \sum_{j=0}^{\infty} \gamma^j r_{j,i} + \sum_{j=0}^{\infty} \gamma^{j+1}\Phi(s_{j+1},t_{j+1}) - \sum_{j=0}^{\infty} \gamma^j \Phi(s_j,t_j) \\
&= U_i(\bar{s}) + \sum_{j=1}^{\infty} \gamma^j \Phi(s_j,t_j) - \sum_{j=1}^{\infty} \gamma^j \Phi(s_j,t_j) - \Phi(s_0,t_0) \\
&= U_i(\bar{s}) - \Phi(s_0,t_0)
\end{aligned} \tag{4.16}$$

Then by combining 4.15 and 4.16, the shaped Q-function is:

$$\begin{aligned} Q^*_{i,\Phi}(s,a) &= \sum_{\bar{s}} Pr(\bar{s}|s,a)U_{i,\Phi}(\bar{s}) \\ &= \sum_{\bar{s}} Pr(\bar{s}|s,a)(U_i(\bar{s}) - \Phi(s,t)) \\ &= \sum_{\bar{s}} Pr(\bar{s}|s,a)U_i(\bar{s}) - \sum_{\bar{s}} Pr(\bar{s}|s,a)\Phi(s,t) \\ &= Q^*_i(s,a) - \Phi(s,t) \end{aligned} \tag{4.17}$$

where t is the current time.

As the difference between the original Q-values and the shaped Q-values is not dependent on the action taken, then in any given state the best (or best response) action remains constant regardless of shaping. Therefore, I can conclude that the guarantees of policy invariance and consistent Nash equilibria remain.

4.4.2 Non-Equivalence To Q-Table Initialisation

In both single-agent [Wiewiora, 2003] and multi-agent RL, PBRS with a static potential function is equivalent to initialising the agent's Q-table such that:

$$\forall s, a | Q(s,a) = \Phi(s) \tag{4.18}$$

where $\Phi(\cdot)$ is the same potential function as used by the shaped agent.

However, with a dynamic potential function this result no longer holds. The proofs require an agent with PBRS and an agent with the above Q-table initialisation to have an identical probability distribution over their next action provided the same history of states, actions and rewards.

If the Q-table is initialised with the potential of states prior to experiments ($\Phi(s, t_0)$), then any future changes in potential are not accounted for in the initialised agent. Therefore, after the agents experience a state where the shaped agent's potential function has changed, the probability distribution over subsequent action choices in the previous state will be different for each agent.

Formally this can be proved by considering agent L that receives dynamic PBRS and agent L' that does not but is initialised as in Equation 4.18. Agent L will update its Q-values by the rule:

$$Q(s,a) \leftarrow Q(s,a) + \underbrace{\alpha\left(r_i + F(s,t,s',t') + \gamma \max_{a'} Q(s',a') - Q(s,a)\right)}_{\delta Q(s,a)} \tag{4.19}$$

where $\Delta Q(s,a) = \alpha \delta Q(s,a)$ is the amount that the Q value will be updated by.

The current Q-values of Agent L can be represented formally as the initial value plus the change since:

$$Q(s,a) = Q_0(s,a) + \Delta Q(s,a) \tag{4.20}$$

where $Q_0(s,a)$ is the initial Q-value of state-action pair (s,a). Similarly, agent L' updates its Q-values by the rule:

$$Q'(s,a) \leftarrow Q'(s,a) + \alpha \underbrace{(r_i + \gamma \max_{a'} Q'(s',a') - Q'(s,a))}_{\delta Q'(s,a)} \tag{4.21}$$

And its current Q-values can be represented formally as:

$$Q'(s,a) = Q_0(s,a) + \Phi(s,t_0) + \Delta Q'(s,a) \tag{4.22}$$

where $\Phi(s,t_0)$ is the potential for state s before learning begins.

For the two agents to act the same they must choose their actions by relative difference in Q-values, not absolute magnitude, and the relative ordering of actions must remain the same for both agents. Formally:

$$\forall s,a,a' | Q(s,a) > Q(s,a') \Leftrightarrow Q'(s,a) > Q'(s,a') \tag{4.23}$$

In the base case this remains true, as both $\Delta Q(s,a)$ and $\Delta Q'(s,a)$ equal zero before any actions are taken, but after this the proof falters for dynamic potential functions.

Specifically, when the agents first transition to a state where the potential has changed agent L will update $Q(s,a)$ by:

$$\begin{aligned}
\delta Q(s,a) &= r_i + F(s,s') + \gamma \max_{a'} Q(s',a') - Q(s,a) \\
&= r_i + \gamma\Phi(s',t') - \Phi(s,t) + \gamma \max_{a'}(Q_0(s',a') + \Delta Q(s',a')) \\
&\quad -Q_0(s,a) - \Delta Q(s,a) \\
&= r_i + \gamma\Phi(s',t') - \Phi(s,t_0) + \gamma \max_{a'}(Q_0(s',a') + \Delta Q(s',a')) \\
&\quad -Q_0(s,a) - \Delta Q(s,a)
\end{aligned} \tag{4.24}$$

and agent L' will update $Q'(s,a)$ by:

$$\begin{aligned}
\delta Q'(s,a) &= r_i + \gamma \max_{a'} Q'(s',a') - Q'(s,a) \\
&= r_i + \gamma \max_{a'}(Q_0(s',a') + \Phi(s',t_0) + \Delta Q'(s'a')) \\
&\quad -Q_0(s,a) - \Phi(s,t_0) - \Delta Q'(s,a) \\
&= r_i + \gamma \max_{a'}(Q_0(s',a') + \Phi(s',t_0) + \Delta Q(s'a')) \\
&\quad -Q_0(s,a) - \Phi(s,t_0) - \Delta Q(s,a) \\
&= r_i + \gamma\Phi(s',t_0) - \Phi(s,t_0) + \gamma \max_{a'}(Q_0(s',a') + \Delta Q(s',a')) \\
&\quad -Q_0(s,a) - \Delta Q(s,a) \\
&= \delta Q(s,a) - \gamma\Phi(s',t') + \gamma\Phi(s',t_0) \qquad (4.25)
\end{aligned}$$

But the two are not equal as:

$$\Phi(s',t') \neq \Phi(s',t_0) \qquad (4.26)$$

Therefore, for this state-action pair:

$$Q'(s,a) = Q(s,a) + \Phi(s,t_0) - \alpha\gamma\Phi(s',t') + \alpha\gamma\Phi(s',t_0) \qquad (4.27)$$

but for all other actions in state s:

$$Q'(s,a) = Q(s,a) + \Phi(s,t_0) \qquad (4.28)$$

Once this occurs the differences in Q-values between agent L and agent L' for state s would no longer be constant across all actions. If this difference is sufficient to change the ordering of actions (i.e. Equation 4.23 is broken), then the policy of any rational agent will have different probability distributions over subsequent action choices in state s.

In single-agent problem domains, provided the standard necessary conditions are met, the difference in ordering will only be temporary as agents initialised with a static-potential function and/or those receiving dynamic PBRS will converge to the optimal policy. In these cases the temporary difference will only affect the exploration of the agents not their goal.

In multi-agent cases, as was shown earlier, altered exploration can alter final joint-policy and, therefore, the different ordering may remain. However, as I have proven in the previous sub-section, this is not indicative of a change in the goals of the agents.

In both cases, I have shown how an agent initialised as in Equation 4.18 can after the same experiences behave differently to an agent receiving dynamic PBRS. This occurs because the initial value given to a state cannot capture subsequent changes in its potential.

Alternatively, the initialised agent could reset its Q-table on each change in potential to reflect

the changes in the shaped agent. However, this approach would lose all history of updates due to experiences had and so again cause differences in behaviour between the shaped agent and the initialised agent. Furthermore, this method and other similar methods of attempting to integrate change in potential after the agent has begun to learn are also no longer strictly Q-table initialisation.

Therefore, I conclude that there is not a method of initialising an agent's Q-table to guarantee equivalent behaviour to an agent receiving dynamic PBRS.

4.5 Empirical Demonstration

To clarify the theorised effects of PBRS on MARL, an empirical study of Boutilier's coordination game [Boutilier, 1999] will be presented here. This domain has been chosen because it has both multiple Nash equilibria, and joint policies that are not representative of a Nash equilibrium. The former is necessary to demonstrate that PBRS can cause agents to learn a different joint policy, whilst the latter is needed to show that PBRS will never cause them to learn a joint policy that is not representative of a Nash equilibrium.

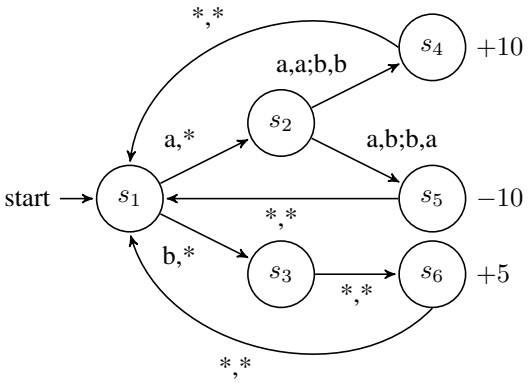


Figure 4.1: Boutilier's Coordination Game

The game, illustrated in Figure 4.1, has six stages and two agents, each capable of two actions (a or b). The first agent's first action choice in each episode decides if the agents will move to a state guaranteed to reward them minimally (s_3) or to a state where they must co-ordinate to receive the highest reward (s_2). However, in state s_2 the agents are at risk of receiving a large negative reward if they do not choose the same action.

In Figure 4.1, each transition is labeled with one or more action pairs such that the pair $a, *$ means this transition occurs if agent 1 chooses action a and agent 2 chooses either action. When multiple action pairs result in the same transition the pairs are separated by a semicolon(;).

The game has multiple Nash equilibria; the joint policies opting for the safety state s_3 or the joint policies of moving to state s_2 and coordinating on both choosing a or b. Any joint policy

receiving the negative reward is not a Nash equilibrium, as the first agent can choose to change its first action choice and so receive a higher reward by instead reaching state s_3.

Six sets of agents will be the focus of these experiments. One set will receive no reward shaping, to illustrate the average performance without heuristic knowledge. The other sets will receive PBRS from either:

- a good, static heuristic encouraging coordination to the highest reward;
- a bad, static heuristic encouraging miscoordination to the lowest reward;
- another static heuristic encouraging the agents to opt for the safety reward of state s_3;
- a uniform, random dynamic heuristic that never converges;
- or a random dynamic heuristic that never converges and encourages miscoordination.

The good heuristic, designed to encourage co-operation, gives states s_1, s_2 and s_4 the potentials 5, 10 and 15 respectively. All other states receive a potential of 0. Alternatively, the bad heuristic is designed to encourage miscoordination and so potentials of 5, 10 and 15 are given instead to states s_1, s_2 and s_5 respectively. Again all other states receive a potential of 0. The final static heuristic, gives zero potential to all states except states s_1, s_3 and s_6 which receive the linearly increasing potentials 5, 10 and 15 respectively.

The uniform random function will choose potentials in the range 0 to 50. Therefore, the additional rewards from shaping will often be larger than those received from the environment when following the optimal policy.

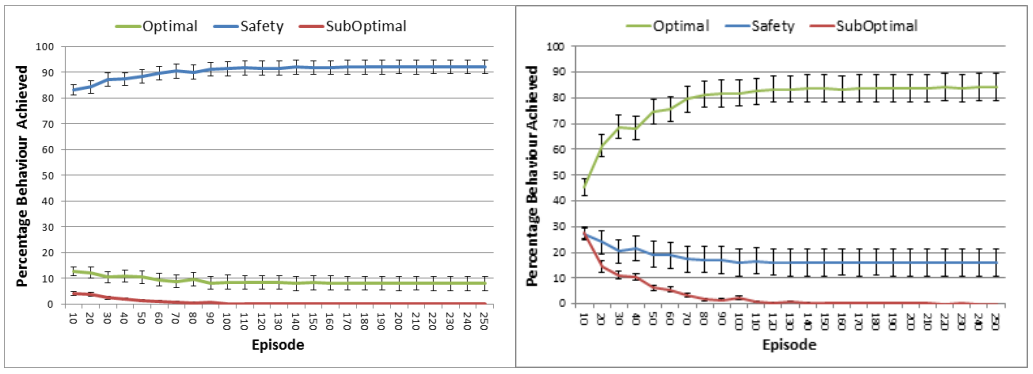
The negative bias random function will choose potentials in the range 35 to 50 for state s_5 (the suboptimal state) or 0 to 15 for all other states. This potential function is biased towards the suboptimal policy, as any transition into state s_5 will be rewarded positively and will often give a higher reward than transitioning to state s_4.

These experimental results are intended to clarify that multiple agents receiving PBRS, regardless of heuristic used, will only ever converge to Nash equilibria of the original system.

4.5.1 Results

All experiments were run for 500 episodes (1,500 action choices) and repeated 100 times. The illustrated results, plot the mean percentage of the last 100 episodes performing the optimal, safety and sub-optimal joint policies respectively. All figures include error bars illustrating the standard error from the mean. For clarity, graphs are plotted only up to 250 episodes as by this time all experiments had converged to a stable joint policy.

All agents, both with and without reward shaping, used Q-learning with ϵ-greedy exploration and a tabular representation of the environment. Experimental parameters were set as $\alpha = 0.5$,$\gamma = 0.99$ and ϵ begins at 0.3 and decays by 0.99 each episode. These parameters were chosen



(a) without Reward Shaping (b) with the Optimal Nash Equilibrium Encouraged

Figure 4.2: Boutilier's Coordination Game

from a large set of values tested in preliminary studies for the clarity of the results. Although the learning rate α is high, it does not affect the theoretical guarantees.

Figure 4.2a shows that the agents without reward shaping rarely (less than ten percent of the time) learn to perform an optimal joint-policy. However, as all joint policies representative of the suboptimal behaviour are not Nash equilbria, these agents never learn to behave this way.

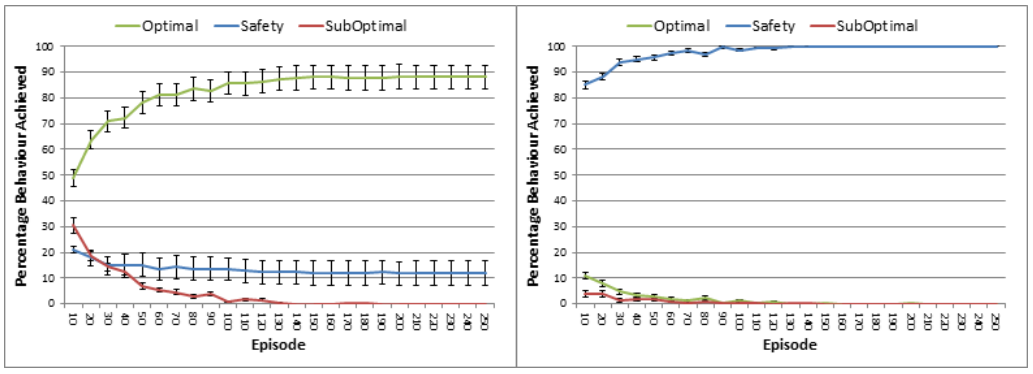
Figure 4.2b shows that, if provided a good heuristic, PBRS can significantly increase the probability of convergence to an optimal joint policy whilst still never learning to perform the sub-optimal behaviour.

Surprisingly, Figure 4.3a shows agents learning with PBRS encouraging miscoordination perform better on average than those encouraged to coordinate. This occurs because of the design of the game and the heuristic used. Encouraging miscoordination, encourages the agents to try state s_2. By modifying the exploration of the agents to include this state more often, there is a higher probability that the agents will coordinate and discover the optimal joint policy. Furthermore, given that the difference in potential between state s_2 and states s_4 and s_5 is only 5, transitioning from state s_2 to s_4 still receives a positive reward whilst transitioning from state s_2 to state s_5 still receives a negative reward.

More importantly, Figure 4.3a also shows that, despite being encouraged to miscoordinate, the agents never learn to do so as this behaviour is not a Nash equilibrium of the original system.

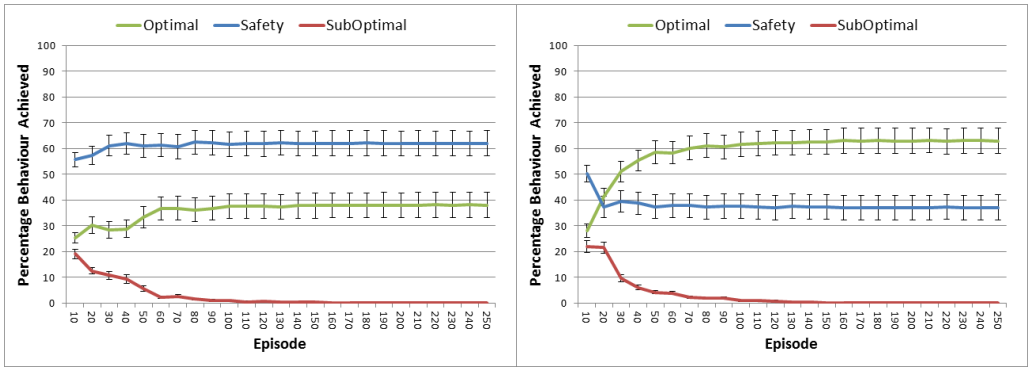
Finally, Figure 4.3b shows that, when encouraged to try the safety behaviour, agents will always learn to stick with the safety reward; lowering their average return compared to other agents but still never learning to follow the sub-optimal behaviour.

In all of these experiments, regardless of shaping, agents never converged to consistently perform the sub-optimal joint policy. This is because miscoordination in this game is not a Nash equilibrium, both with and without PBRS. Regardless of which joint policy is encouraged, if the additional reward is potential based, the Nash equilibria remain constant.



(a) with Miscoordination Encouraged (b) with the Safety Nash Equilibrium Encouraged

Figure 4.3: Boutilier's Coordination Game



(a) with Uniform, Random, Dynamic PBRS (b) with Negative Bias, Random, Dynamic PBRS

Figure 4.4: Boutilier's Coordination Game

Moving on to dynamic potential functions, as illustrated by Figure 4.4, both sets of agents receiving dynamic PBRS also learn the optimal policy more often than the agents without reward shaping.

Furthermore, please note, the agents never converge to perform the suboptimal joint policy. Instead the agents will only ever converge to the safety or optimal joint policies; the Nash equilibria of the unshaped and shaped systems. Thus demonstrating that, even with dynamic reward transformations that never stabilise, the Nash equilibria of the system remain the same provided the transformations are potential based.

4.6 Properties Invariant to Changes in Absolute Value

The experiments in the previous section were included to clarify the theoretical result of consistent Nash equilibria. When developing theoretical results for MARL, the emphasis is

typically on Nash equilibrium as this is the most common learning goal. However, the theoretical effect of PBRS can be generalised further.

Specifically, PBRS does not modify any property of the underlying MDP or SG invariant to changes in absolute value of expected return. Provided a property is only reliant on the relative difference or order of expected returns, PBRS will not affect it.

In single-agent RL, the proof of policy invariance occurs because the optimal policy is still the optimal policy after PBRS because it still has the highest expected return albeit now at a lower absolute value. In multi-agent RL, the proof of consistent Nash equilibria is possible because in a similar manner an agent's best response remains its best response despite all responses having a lower absolute value.

Finally, as a novel example, consider multi-objective RL [Vamplew et al., 2011] which has a significantly different goal to both single-agent and multi-agent RL. In multi-objective RL, an agent receives a vector of rewards to maximise as opposed to a single, scalar reward as is typical in most RL applications. The agent's goal then becomes to find one of or all the solutions along the Pareto front. Where the Pareto front is the set of policies that are not Pareto dominated by any other policy, and policy a is said to Pareto dominate policy b if policy a receives higher reward from one or more objectives than policy b and receives equal reward in all others. If PBRS were to be applied to multi-objective RL, the absolute value of all shaped objectives would be modified but the Pareto front would remain constant.

4.7 Application to Other Algorithms

So far in this chapter all proofs have presumed the use of multiple independent Q-Learning agents in fully observable environments. However, these proofs can be extended to other RL algorithms. As the number of MARL algorithms continues to grow, the proofs in this chapter will need to be altered to ensure PBRS can be applied whilst maintaining the same guarantees. This section is intended to clarify how to apply PBRS to novel algorithms to help future applications.

For example, some may consider the presumption of full observability to be uncharacteristic of MAS. However, by shaping agents based on the potential of observations (as opposed to fully observed states) and replacing all occurrence of states with observations in the arguments and proofs above, the same theoretical expectations can be proven in partially observable problem domains. Specifically, the Nash equilibria of a POMDP would remain the same but the agents exploration will alter and so convergence (if it occurs) may be to a different point of equilibrium. We recently explored the application of this approach in collaboration with Adam Eck and Dr. Leen-Kiat Soh [Eck et al., 2013]. Alternatively, joint-action Q-Learning agents would need to replace all occurrences of an individual's action a with the corresponding joint action $\mathbf{a}$ to construct similar proofs.

Other algorithms require more thorough manipulation, for example, consider SMDPs. As noted in Section 2.1.2, when learning in an SMDP as opposed to a regular MDP the typical

Q-Learning/SARSA update rule can be modified to:

$$Q(s,a) \leftarrow Q(s,a) + \alpha[r + \gamma^{\Delta t} Q(s',a') - Q(s,a)] \tag{4.29}$$

where s is the source state, a the action taken, α the learning rate, γ the discount factor, s' the resultant state, $a' = \max_{a'} Q(s',a')$ if using Q-Learning or a' is the action taken in the resultant state if using SARSA and Δt is the change in time between states s and s'.

To maintain the proofs of policy invariance in single-agent RL and consistent Nash equilibria in MARL given this update rule, the additional rewards provided by PBRS must become:

$$F(s,s') = \gamma^{\Delta t}\Phi(s') - \Phi(s) \tag{4.30}$$

where $\gamma^{\Delta t}$ is equal to the same value used in Equation 4.29 in the corresponding update.[2]

Currently, no algorithms have been found to date that PBRS cannot be manipulated for to provide the same guarantees proved earlier in this chapter or in the original proof for single-agent RL [Ng et al., 1999].

4.8 Finite Potential-Based Reward Shaping

Finally, all proofs in this chapter also presume the agents have an infinite sequence of experiences. However, in many practical applications (including all those presented within this thesis) agents learn in a finite, episodic environment. In this section I will prove that by removing the assumption of infinite experiences the relative difference in policy values can be altered and, therefore, extra conditions on the potential function are needed to maintain the theoretical guarantees in these applications.

Formally, when limiting Equation 4.14, the expected return U_i of agent i when receiving the original reward signal alone, to a finite sequence of experiences $\bar{s}^H$ where H is the number of states visited becomes:

$$U_i^H(\bar{s}) = \sum_{j=0}^{H-1} \gamma^j r_{j,i} \tag{4.31}$$

Now consider limiting Equation 4.16, the expected return $U_{i,\Phi}$ of the same agent when receiving PBRS, to the same finite sequence of experiences:

[2] Ng [2003] commented on a similar adaption of the additional rewards if using an alternative update rule for SMDP [Bradtke and Duff, 1995].

$$\begin{aligned} U^H_{i,\Phi}(\bar{s}^H) &= \sum_{j=0}^{H-1} \gamma^j (r_{j,i} + F(s_j, t_j, s_{j+1}, t_{j+1})) \\ &= \sum_{j=0}^{H-1} \gamma^j (r_{j,i} + \gamma\Phi(s_{j+1}, t_{j+1}) - \Phi(s_j, t_j)) \\ &= \sum_{j=0}^{H-1} \gamma^j r_{j,i} + \sum_{j=0}^{H-1} \gamma^{j+1}\Phi(s_{j+1}, t_{j+1}) - \sum_{j=0}^{H-1} \gamma^j \Phi(s_j, t_j) \\ &= U^H_i(\bar{s}) + \gamma^H \Phi(s_H, t_H) - \Phi(s_0, t_0) \end{aligned} \tag{4.32}$$

The difference in return the agent will receive when receiving PBRS compared to the environment's reward signal alone is now dependent on both the original state and the final state, due to the terms $-\Phi(s_0, t_0)$ and $\gamma^H\Phi(s_H, t_H)$ respectively. As the final state may change dependent on actions taken, the difference in policy values will no longer be constant across all policies. This can change the ordering of policies and, therefore, breaks the guarantees of policy invariance and consistent Nash equilibria.

If $\gamma < 1$, episodes are often of sufficient length for the term $\gamma^H\Phi(s_H, t_H)$ to become insignificantly small and, therefore, not alter the optimal policy or Nash equilibria.

However, to ensure the guarantees still hold, the potential of all final states in an environment must be set to 0. The simplest method to do so is to have all agents transition to an absorbing state after the final step of the episode[3]. As no reward is received from the environment on this transition, only the difference in potential, policy values are not altered by this additional state. Furthermore, given that $\Phi(s_H, t_H) = 0$, the term $\gamma^H\Phi(s_H, t_H)$ can be removed from Equation 4.32 thus maintaining all previous guarantees despite a finite number of experiences.

4.9 Conclusion

In conclusion, this chapter showed how two fundamental papers in single-agent reward shaping [Ng et al., 1999; Wiewiora, 2003] can be extended to provide similar guarantees in MARL. Specifically, applying PBRS to MARL does not alter the Nash equilibria of the underlying SG and, provided the potential function is static, each shaped agent is still equivalent to an agent with initial Q-values set to the potential of each state.

Furthermore, this chapter also proved how a dynamic potential function can be used to shape an agent without altering its optimal policy/best response provided the additional reward given is of the form:

$$F(s, t, s', t') = \gamma\Phi(s', t') - \Phi(s, t)$$

Contrary to previous opinion, this chapter included empirical evidence that, the dynamic potential

[3] As recommended in the original PBRS paper. [Ng et al., 1999]

function does not need to converge before the agent receiving shaping can. This result justifies a number of pre-existing implementations of dynamic reward shaping [Grześ and Kudenko, 2008; Grześ and Kudenko, 2010; Laud, 2004; Marthi, 2007] and encourages ongoing research into automated processes of generating potential functions.

In general, I have argued, PBRS does not alter any property of a SG that is invariant to changes in absolute value of rewards. Therefore, if the favoured solution concept of MARL someday becomes Pareto optimality, PBRS could still be applied without altering the goal from that of the unshaped agents.

However, PBRS does affect the exploration of the shaped agent. Therefore, it can change the joint policy converged upon as even just one agent's modified exploration can sufficiently redirect the search of joint policy space to converge to a different point of equilibrium. Whether the joint policy learnt is the Nash equilibrium of highest global utility, is dependent on the agents' learning algorithms. With multiple independent learners, no guarantee of convergence to the highest utility Nash equilibrium is provided. However, PBRS can, dependent on the heuristic, either increase or decrease the probability of converging to equilibria of higher global utility.

Therefore, how to design potential functions for MARL remains an open and interesting question. In the previous chapter, experiments were performed with a number of application specific heuristics encoded as potential functions. As I argued in the closing of that chapter, these could be applied in many MAS. However, the process is manual and domain dependent. In the next chapter, I will discuss a more general method of generating potential functions for MARL based on transforming multi-agent plans into additional rewards consistent with the required forms for the proofs given in this chapter.

CHAPTER 5

Designing Multi-Agent Potential Functions

Given the supporting theory has been proven in the previous chapter, it is important to now consider how to design suitable potential functions for MARL. Often when PBRS is applied to a new problem domain, the potential function is coded manually using the novel prior knowledge for that domain. This method was demonstrated for MARL earlier, in Chapter 3.

Therefore, this chapter will instead focus on developing a semi-automated method of designing a potential function for multi-agent PBRS. Specifically, I will detail a study on expanding the pre-existing work on plan-based reward shaping from single-agent problem domains to MARL.

In the next section, I will begin by reviewing methods of multi-agent planning emphasising the different paradigms that are used in the remainder of the chapter when experimenting with plan-based reward shaping in a multi-agent environment.

5.1 Multi-Agent Planning

Multi-agent planning is the combination of planning and coordination [De Weerdt et al., 2005]. Unlike single-agent planning which requires no coordination, multi-agent planning must co-ordinate multiple plans to prevent conflicts occurring stopping any one or all of the agents accomplishing their own goals.

Three general approaches are common, they are: decentralised planning for centralised plans, centralised planning for decentralised plans and decentralised planning for decentralised plans.

As the inclusion of this section is to survey methods applicable to plan-based reward shaping in MARL, the decentralised planning for centralised plans is not relevant. However, for complete

coverage, it suffices to say that single-agent plans can be generated by multiple, heterogeneous experts either sequentially with rolling back upon conflicts or generated in parallel and then merged [Ziparo, 2005].

5.1.1 Centralised Planning for Decentralised Plans

Centralised planning for distributed plans can be summarised as generating a plan then decomposing it and assigning it to multiple agents to execute. Decomposing the plan is an optimisation problem and task assignment methods can be used to complete the process. [Ziparo, 2005]

Planning centrally fails to efficiently make use of the computational power of a MAS as multiple agents remain idle throughout the planning phase. This approach is not always applicable to MAS as they were defined in Section 2.2, because the multiple independent owners represented by the agents will often not want to divulge their plans or goals to a central planning agent.

5.1.2 Decentralised Planning for Decentralised Plans

Alternatively, decentralised planning for decentralised plans uses the computational power of every agent throughout and owners do not need to divulge information regarding the plans or goals of their agent.

However, in such an approach the difficulty of coordination is paramount and its occurrence becomes a defining step in how different algorithms solve this approach. Specifically, coordination can occur pre-planning, interleaved with planning or post-planning.

Pre-Planning Coordination

To co-ordinate pre-planning, social laws such as always drive on the left can be designed to reduce the chance of conflicts in plans [Shoham and Tennenholtz, 1995]. Another approach is to introduce constraints based on an analysis of interdependencies in plans [De Weerdt et al., 2005].

However, the former approach only reduces the likeliness of conflict and the latter approach again violates the need for each agent's goals to remain private knowledge.

Interleaved Coordination

Alternatively, coordination can occur interleaved with planning. In this approach agents partially plan, execute, then plan some more. This pattern is repeated continuously throughout simulation. A classic example of this is the Partial Global Planning (PGP) framework [Durfee and Lesser, 1987] which was later expanded to the General Partial Global Planning (GPGP) framework [Decker and Lesser, 1992].

Post-Planning Coordination

Finally, to co-ordinate post-planning, contingency plans can be planned before execution and reverted to in times of conflict or locally re-planned at the time of conflict [De Weerdt et al.,

2005]. This approach introduces significant extra computation upon each conflict that occurs. Therefore, it is suitable for loosely coupled systems where conflict is unlikely but highly inefficient in, for example, competitive games where conflict is expected regularly.

Another approach [Brafman and Domshlak, 2008; Nissim et al., 2010], combines planning with constraint satisfaction algorithms. If the system is loosely coupled, it is feasible to solve the post-planning coordination problem by constraint satisfaction. However, this is another representative example of the extra computational power coordination requires and again requires agents to share their private plans.

Other post-planning approaches include submitting the completed plans to a controller agent for merging or iterating amongst agents, communicating or negotiating their own intended steps to check for conflicts and progressively building up each individual's plan without conflict [Ziparo, 2005]. However, these approaches yet again violate the sharing of private plans and goals.

5.1.3 Summary

Throughout this review it should be apparent that coordination is a recurring issue. Despite multiple approaches, no one method is commonly regarded as the solution and research has continued progressively attempting to find a better solution.

Furthermore, it has been argued that the common assumption in the field of multi-agent planning that learning is not required is inherently flawed and limiting the potential of what can be achieved [De Weerdt et al., 2005]. This argument supports the idea that the combination of fields, multi-agent planning and MARL, could be mutually beneficial to both. The next section, where I will discuss how to use existing methods of multi-agent planning to extend plan-based reward shaping, is an example of this combination.

5.2 Multi-Agent, Plan-Based Reward Shaping

Based on the two relevant approaches to multi-agent planning, discussed in the previous section, I propose two methods of extending plan-based reward shaping to MARL.

The first, joint-plan based reward shaping, employs the concept of centralised planning for decentralised plans and so generates, where possible, plans without conflict. This shaping is expected to outperform the alternative but may not be possible in competitive environments where agents are unwilling to cooperate.

Alternatively, individual-plan-based reward shaping, requires no cooperation as each agent plans as if it is alone in the environment.

Unfortunately, the application of individual-plan-based reward shaping to multi-agent problem domains is not as simple in practice as it may seem. The knowledge given by multiple individual plans will often be conflicted and agents may need to deviate significantly from this prior knowledge when acting in their common environment. However, reward shaping only

encourages a path of exploration, it does not enforce a joint-policy. Therefore, it may be possible that RL agents, given conflicted plans initially, can learn to overcome their conflicts and eventually follow coordinated policies.

For both methods, the STRIPS plan of each agent is translated into a list of states so that, whilst acting, an agent's current state can be compared to all plan steps. The potential of the agent's current state then becomes:

$$\Phi(s) = \omega * CurrentStepInPlan \tag{5.1}$$

where ω is a scaling factor and $CurrentStepInPlan$ is the index of the corresponding state in the state-based representation of the agent's plan (for example see Listing 5.5).

If the current state is not in the state-based representation of the agent's plan, then the potential used is that of the last state experienced that was in the plan. This was implemented in the original work to not discourage exploration off of the plan and is now more relevant because, in the case of individual plans, strict adherence to the plan by every agent will not be possible. This feature of the potential function makes plan-based reward shaping an instance of dynamic PBRS [Devlin and Kudenko, 2012].

Finally, to preserve the theoretical guarantees of PBRS in episodic problem domains, the potential of all goal/final states is set to zero. These potentials are then used as in Equation 2.6 to calculate the additional reward given to the agent.

In the next section I will introduce a problem domain and the specific implementations of both proposed methods in that domain.

5.3 Empirical Study

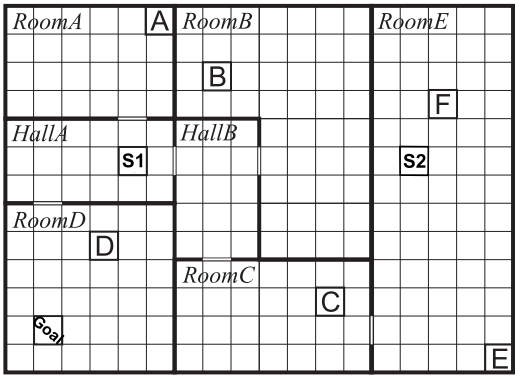


Figure 5.1: Multi-Agent, Flag-Collecting Problem Domain

The chosen problem for this study is a flag collecting task in a discrete, grid-world domain with two agents attempting to collect six flags spread across seven rooms. An overview of this world is illustrated in Figure 5.1 with the goal location labeled as such, each agent's starting

location labeled Si where i is their unique id and the remaining labeled grid locations being flags and their unique id.

At each time step an agent can move up, down, left or right and will deterministically complete their move provided they do not collide with a wall or the other agent. Once an agent reaches the goal state their episode is over regardless of the number of flags collected. The entire episode is completed when both agents reach the goal state. At this time both agents receive a reward equal to one hundred times the number of flags they have collected in combination. No other rewards are given by the environment at any other time. To encourage the agents to learn short paths, the discount factor γ is set to less than one.[1]

Additionally, as each agent can only perceive its own location and the flags it has already picked up, the problem domain is partially observable; a characteristic feature of many MAS.

Given this domain, the plans of agent 1 and agent 2 with joint-plan based reward shaping are documented in Listings 5.1 and 5.2. It is important to note that these plans are coordinated with no conflicting actions.

Listing 5.1: Joint-Plan for Agent 1
Starting in HallA

```
MOVE(hallA,roomA)
TAKE(flagA,roomA)
MOVE(roomA,hallA)
MOVE(hallA,hallB)
MOVE(hallB,roomB)
TAKE(flagB,roomB)
MOVE(roomB,hallB)
MOVE(hallB,hallA)
MOVE(hallA,roomD)
```

Listing 5.2: Joint-Plan for Agent 2
Starting in RoomE

```
TAKE(flagF,roomE)
TAKE(flagE,roomE)
MOVE(roomE,roomC)
TAKE(flagC,roomC)
MOVE(roomC,hallB)
MOVE(hallB,hallA)
MOVE(hallA,roomD)
TAKE(flagD,roomD)
```

Alternatively, Listings 5.3 and 5.4 document the plans used to shape agent 1 and agent 2 respectively when receiving individual-plan-based reward shaping. However, now both plans cannot be completed as each intends to collect all flags. How, or if, the agents can learn to overcome this conflicting knowledge is the focus of this investigation.

[1] Experiments with a negative reward on each time step and $\gamma = 1$ made no significant change in the behaviour of the agents.

Listing 5.3: Individual Plan for Agent 1 Starting in HallA

```
MOVE(hallA, hallB)
MOVE(hallB, roomC)
TAKE(flagC, roomC)
MOVE(roomC, roomE)
TAKE(flagE, roomE)
TAKE(flagF, roomE)
MOVE(roomE, roomC)
MOVE(roomC, hallB)
MOVE(hallB, roomB)
TAKE(flagB, roomB)
MOVE(roomB, hallB)
MOVE(hallB, hallA)
MOVE(hallA, roomA)
TAKE(flagA, roomA)
MOVE(roomA, hallA)
MOVE(hallA, roomD)
TAKE(flagD, roomD)
```

Listing 5.4: Individual Plan for Agent 2 Starting in RoomE

```
TAKE(flagF, roomE)
TAKE(flagE, roomE)
MOVE(roomE, roomC)
TAKE(flagC, roomC)
MOVE(roomC, hallB)
MOVE(hallB, roomB)
TAKE(flagB, roomB)
MOVE(roomB, hallB)
MOVE(hallB, hallA)
MOVE(hallA, roomA)
TAKE(flagA, roomA)
MOVE(roomA, hallA)
MOVE(hallA, roomD)
TAKE(flagD, roomD)
```

As mentioned in Section 5.2, these plans must be translated into state-based knowledge. Listing 5.5 shows this transformation for the joint-plan starting in hallA (listed in Listing 5.1) and the corresponding value of ω.

Listing 5.5: State-Based Joint-Plan for Agent 1 Starting in HallA

```
0  robot-in_hallA
1  robot-in_roomA
2  robot-in_roomA  taken_flagA
3  robot-in_hallA  taken_flagA
4  robot-in_hallB  taken_flagA
5  robot-in_roomB  taken_flagA
6  robot-in_roomB  taken_flagA  taken_flagB
7  robot-in_hallB  taken_flagA  taken_flagB
8  robot-in_hallA  taken_flagA  taken_flagB
9  robot-in_roomD  taken_flagA  taken_flagB
```

$$\omega = MaxReward/NumStepsInPlan = 600/9$$

In all these experiments, regardless of knowledge used, the scaling factor ω was set so that the maximum potential of a state is the maximum reward of the environment. As the scaling factor

affects how likely the agent is to follow the heuristic knowledge [Grześ, 2010], maintaining a constant maximum across all heuristics compared ensures a fair comparison. For environments with an unknown maximum reward the scaling factor ω can be set experimentally or based on the designer's confidence in the heuristic.

For comparison, I implemented a team of agents with no prior knowledge/shaping and a team with the domain-specific knowledge that collecting flags is beneficial. These flag-based agents value a state's potential equal to one hundred times the number of flags it alone has collected. This again ensures that the maximum potential of any state is equal to the maximum reward of the environment.

I also considered the combination of this flag-based heuristic with the general methods of joint-plan-based and individual-plan-based shaping. These combined agents value the potential of a state to be:

$$\begin{aligned} \Phi(s) = &\quad (CurrentStepInPlan + NumFlagsCollected) * \omega \\ \omega = &\quad MaxReward/(NumStepsInPlan \\ &\quad + NumFlagsInWorld) \end{aligned} \tag{5.2}$$

where $NumFlagsCollected$ is the number of flags the agent has collected itself, $NumStepsInPlan$ is the number of steps in its state-based plan and $NumFlagsInWorld$ is the total number of flags in the world (i.e. for this domain $NumFlagsInWorld = 6$).

All agents, regardless of shaping, implemented SARSA with $\epsilon-$greedy action selection and eligibility traces. For all experiments, the agents' parameters were set such that $\alpha = 0.1$, $\gamma = 0.99$, $\epsilon = 0.1$ and $\lambda = 0.4$. For these experiments, all initial Q-values were zero.

All experiments have been repeated thirty times with the mean discounted reward per episode presented in the following graphs. All claims of significant differences are supported by two-tailed, two sample t-tests with significance $p < 0.05$ (unless stated otherwise). Plots were made to ensure the assumption of normal distribution required for t-tests held for this data. Given that there are multiple types of agents being compared, it may have been more appropriate to use ANOVA for these experiments.

5.3.1 Results

Figure 5.2 shows all agents, regardless of shaping, learn quickly within the first 300 episodes. In all cases, some knowledge significantly improves the final performance of the agents as shown by all shaped agents out-performing the base agent with no reward shaping.

Agents shaped by knowledge of the optimal joint-plan (both alone or combined with the flag-based heuristic) significantly outperform all other agents, consistently learning to collect all six flags. Please note the joint-plan-based agents' illustrated performance in Figure 5.2 does not

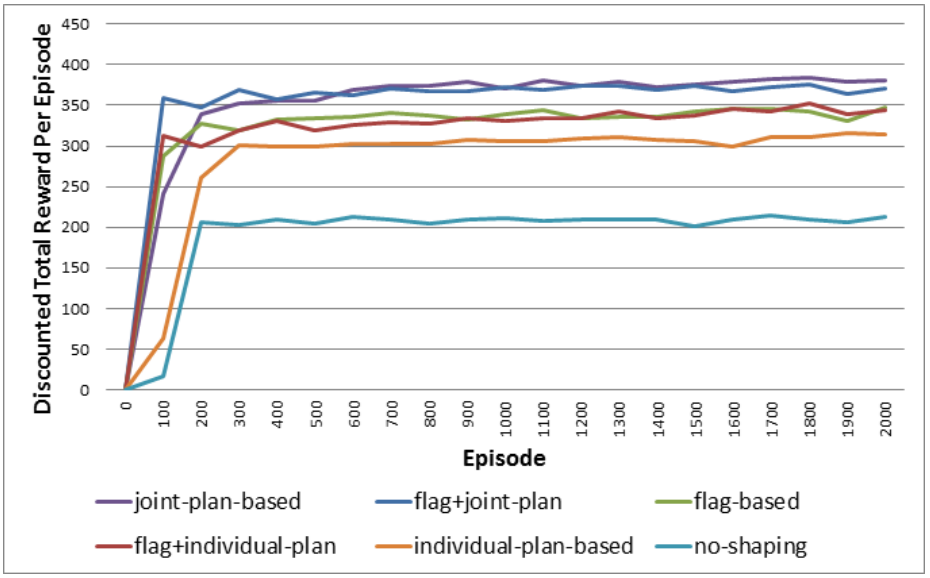


Figure 5.2: Initial Results

reach 600 as the value presented is discounted by the time it takes the agents to complete the episode.

The individual-plan-based agents are unable to reach the same performance as they are given no explicit knowledge of how to coordinate. However, some knowledge, regardless of the number of conflicts, is better than no knowledge. The flag based heuristic can be seen to improve coordination slightly in the agents receiving combined shaping from both types of knowledge, but not sufficiently to overcome the conflicts in the two individual plans.

5.4 Scaling Up

Given the initial results, this section discusses the effect of scaling up the size of the problem domain on these two approaches to multi-agent, plan-based reward shaping. To increase the complexity, I extended the problem domain by adding six extra flags (consequently $MaxReward$ now equals 1200) as illustrated in Figure 5.3, and then adding a third agent as illustrated in Figure 5.5.

3 agents is still relatively small for a MAS, however, this extension is intended to highlight the effect of adding agents to the problem domain. For studies with far larger numbers of agents please see our more recent work on potential-based difference rewards [Devlin et al., 2014] which I will discuss further in Section 6.3.

5.4.1 Extra Flags

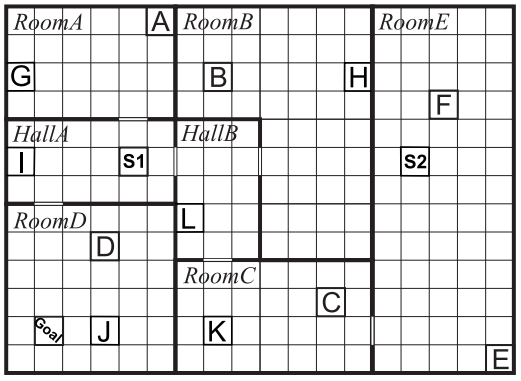


Figure 5.3: Scaled Up Problem Domain

As shown in Figures 5.4, the results with 12 flags and 2 agents were similar to those in the original domain except for a longer time to convergence. The additional time need to learn is due to the larger state space. This experiment shows that multi-agent plan-based reward shaping, and more generally PBRS, can maintain their benefits as the complexity of the problem domain grows.

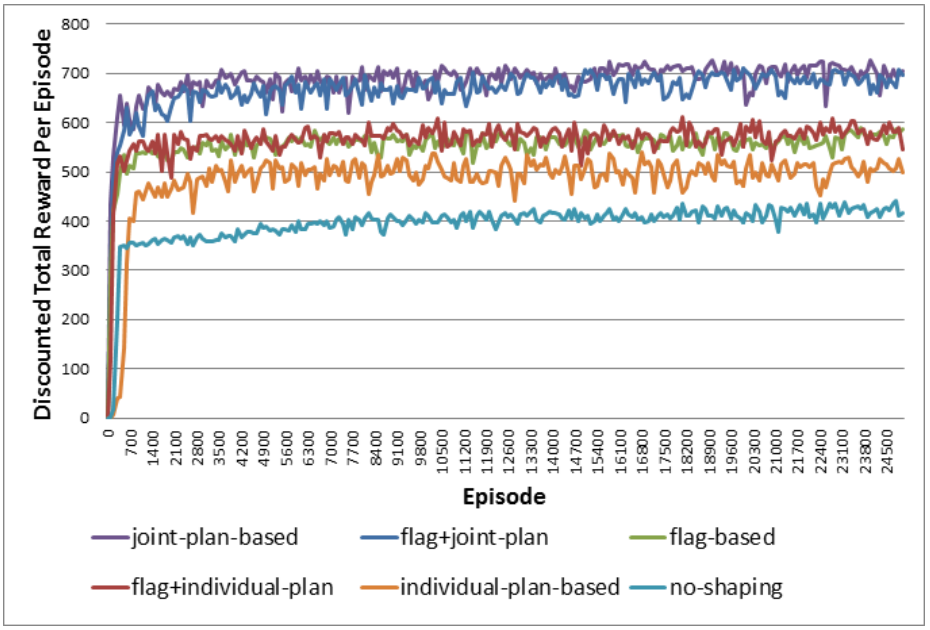


Figure 5.4: Pessimistic Initialisation in the Scaled Up Problem Domain

5.4.2 Extra Agent

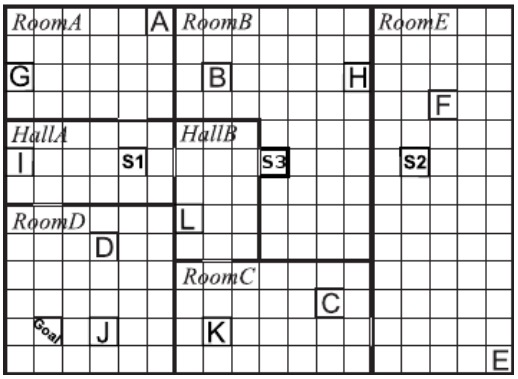


Figure 5.5: Scaled Up Problem Domain with 3 Agents

Finally, Figure 5.6 shows the results for the scaled up setting with 12 flags and 3 agents illustrated in Figure 5.5. Under these settings, the performance of all agents is more variable due to the extra uncertainty the additional agent causes. This is to be expected as the underlying state-action space has grown exponentially whilst, as each agent only considers it's own location and collection of flags, the state space learnt by each agent has not grown. For similar reasons, the agents without shaping or shaped by any potential function that includes the flag heuristic perform significantly worse now than when there were only two agents acting and learning in the environment.

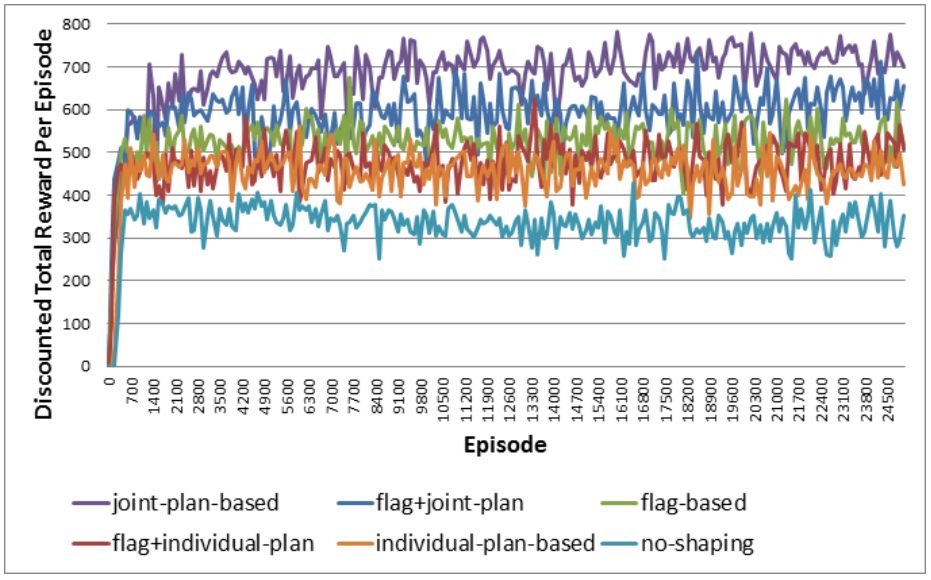


Figure 5.6: Pessimistic Initialisation in the Scaled Up Problem Domain with 3 Agents

Alternatively, the agents shaped by individual plans or joint plans alone have remained robust

to the changes and converge on average to policies of equivalent performance to their counterparts with two agents in the environment. This was expected with the joint-plan agents, as the plans received take into account the third agent and coordinate task allocation prior to learning, but is a positive result for the scalabilty of individual-plan-based reward shaping.

5.5 Overcoming Conflicted Knowledge

This sections discusses a number of options and attempts to help individual-plan-based agents learn regardless of the erroneous knowledge they receive due to the decentralised generation of their guiding plans.

The difference in final performance between individual-plan-based agents and joint-plan-based agents is caused by the conflicted knowledge in the individual plans. By examining the policies learnt by both groups of agents, a significant difference in behaviour becomes apparent.

Specifically, Figure 5.7 illustrates the typical behaviour learnt by joint-plan-based agents. Note that in these examples the agents have learnt the low level implementation of the high level plan provided.

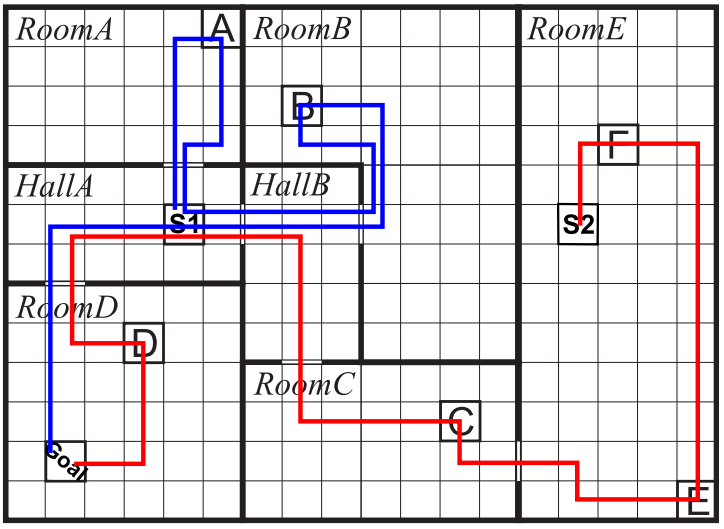


Figure 5.7: Example Behaviour of Joint-Plan-Based Agents

Meanwhile, Figure 5.8 illustrates the typical behaviour learnt by individual-plan-based agents. This time note that agent 1 has opted out of receiving its shaping reward by moving directly to the goal and not following its given plan. The resultant behaviour allows the agents to receive the maximum goal reward from collecting all flags, but at a longer time delay and, therefore, a significantly greater discount.

Occasionally the agents coordinate better with agent 1 collecting flag D or, even rarer, flags D and A. Whilst this is the exception, it is interesting to note that the agent not following its

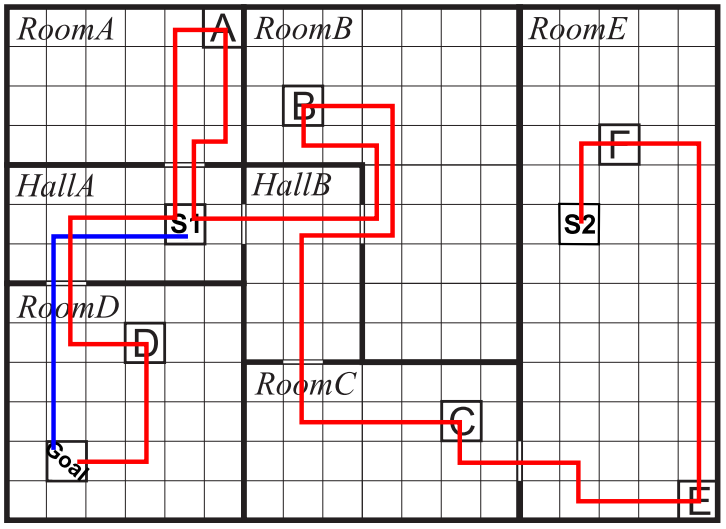


Figure 5.8: Example Behaviour of Individual-Plan-Based Agents

plan will always choose actions that take away from the end of the other agent's plan rather than follow the first steps of their own plan.

One plausible solution would be to combine individual-plan-based reward shaping with FCQ-learning [De Hauwere et al., 2011] to switch to a joint-action representation in states where coordination is required. Another may be to introduce communication between the agents. However, as both multiple independent learners and individual-plan-based reward shaping were designed to avoid sharing information amongst agents, I did not explore these options further.

Without sharing information, agent 1 could be encouraged not to opt out of following its plan by switching to a competitive reward function. However, as illustrated by Figure 5.9, although this closed the gap between individual-plan-based and joint-plan-based agents, the change was detrimental to the team performance of all agents regardless of shaping.

Specifically, individual-plan-based agent 1 did, as expected, start to participate and collect some flags but collectively they would not collect all flags. Both agents would follow their plans to the first two or three flags but then head to the goal as the next flag would not reliably be there. For similar reasons joint-plan-based agents would also no longer collect all flags. Therefore, the reduction in the gap between individual-plan-based and joint-plan-based agents was at the cost of no longer finding all flags. I consider this an undesirable compromise and so will not cover this approach further.

Instead, in the following subsections I will discuss two approaches that lessened the gap by improving the performance of the individual-plan-based agents.

The first of these approaches is increasing exploration in the hope that the agents will experience and learn from policies that coordinate better than those encouraged by their individual plans. The second approach was to improve the individual plans by reducing the

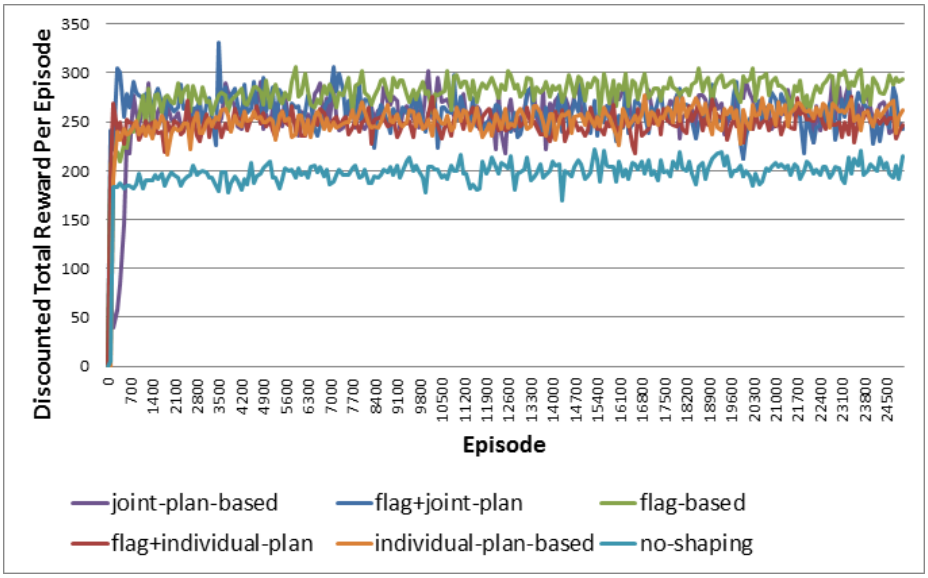


Figure 5.9: Competitive Reward

number of conflicts or increasing the time until conflict.

Both methods enjoy some success and provide useful insight in to how future solutions may overcome incorrect or conflicted knowledge. Where successful, these approaches provide solutions where multiple agents can be deployed without sharing their goals, broadcasting their actions or communicating to coordinate.

5.5.1 Increasing Exploration

Setting all initial Q-values to zero, as was mentioned in Section 5.3, is a pessimistic initialisation given that no negative rewards are received in this problem domain. Agents given pessimistic initial beliefs tend to explore less as any positive reward, however small, once received specifies the greedy policy and other policies will only be followed if randomly selected by the exploration steps [Sutton and Barto, 1998].

With reward shaping and pessimistic initialisation an agent becomes more sensitive to the quality of knowledge they are shaped by. If encouraged to follow the optimal policy they can quickly learn to do so, as is the case in the initial study with the joint-plan-based agents. However, if encouraged to follow incorrect knowledge, such as the conflicted plans of the individual-plan-based agent, they may converge to a sub-optimal policy.

The opposing possibility is to instead initialise optimistically by setting all Q-values to start at the maximum possible reward. In this approach agents explore more as any action gaining less than the maximum reward becomes valued less than actions yet to be tried [Sutton and Barto, 1998].

Figure 5.10 shows the outcome of optimistically initialising the agents with Q-values of 600,

the maximum reward agents can receive in this problem domain.

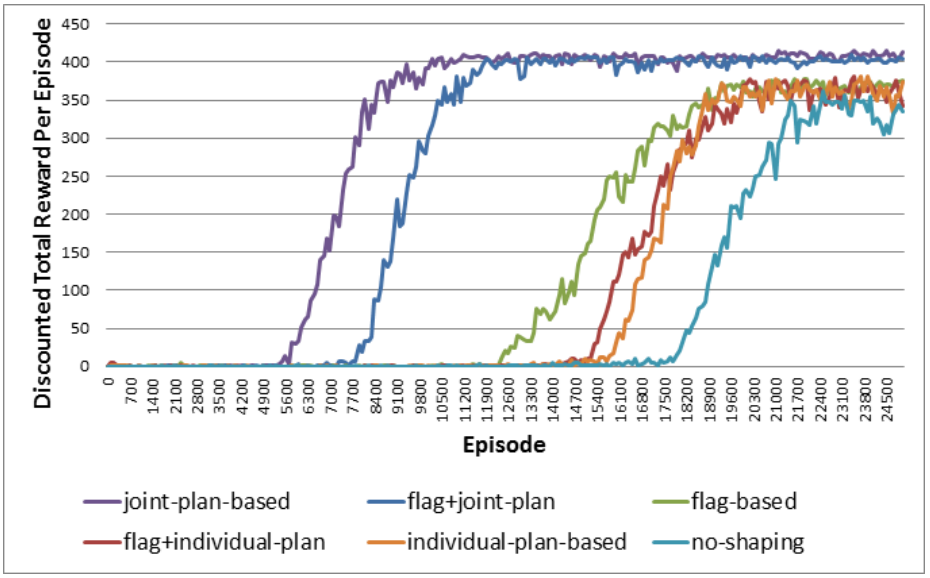


Figure 5.10: Optimistic Initialisation

As would be expected, increased exploration causes the agents to take longer to learn a suitable policy. However, all agents (except for those receiving flag-based or combined-flag+joint-plan shaping) learn significantly better policies than their pessimistic equivalents[2]. This reduces the gap in final performance between all agents and the joint-plan-based agents, but the difference that remains is still significant.

Despite that, the typical behaviour learnt by optimistic individual-plan-based agents is the same as the behaviour illustrated in Figure 5.7. However, it occurs less often in these agents than it occurred in the pessimistic joint-plan-based agents. This illustrates that conflicts can be overcome by optimistic initialisation but it cannot be guaranteed, by this method alone, that the optimal joint-plan will be learnt.

Furthermore, it takes time for the individual-plan-based agents to learn how to overcome the conflicts in their plans. However, this time is still less than it takes the agents with no prior knowledge to learn. Therefore, given optimistic initialisation, the benefit of reward shaping is now more important in the time to convergence instead of the final performance.

To conclude, these experiments demonstrate that some conflicted knowledge can be overcome given sufficient exploration.

5.5.2 Improving Knowledge

An alternative approach to overcoming conflicted knowledge would be to improve the knowledge. The results in this section illustrate that if the amount of the plan that can be followed

[2]For individual-plan-based agents $p = 0.064$, for all others $p < 0.05$.

increases then the time to convergence decreases (when optimistically initialised) or the final performance increases (when pessimistically initialised).

The individual-plan-based agents received shaping based on plans to both collect all six flags. If these plans are followed the agents will collide at their second planned flag to collect. The agent that does not pick up the flag will no longer be able to follow their plan and will therefore receive no further shaping rewards. Instead, these experiments test three groups of agents that are shaped by less conflicted plans.

Specifically, plan-based-6 agents still both plan to collect all six flags, but the initial conflict is delayed until the second or third flag. The comparison of these agents to the individual-plan-based agents will show whether the timing of the conflict affects performance.

Plan-based-5 agents plan to collect just five flags each, reducing the number of conflicted flags to 4. Comparing this to both previous agents and subsequent agents will show whether the number of conflicts affects performance. These agents also experience their first conflict on the second or third flag.

Plan-based-4 agents plan to collect four flags each, reducing the number of conflicted flags to two and delaying the first conflict until the third flag. This agent will contribute to conclusions both on timing of conflicts and amount of.

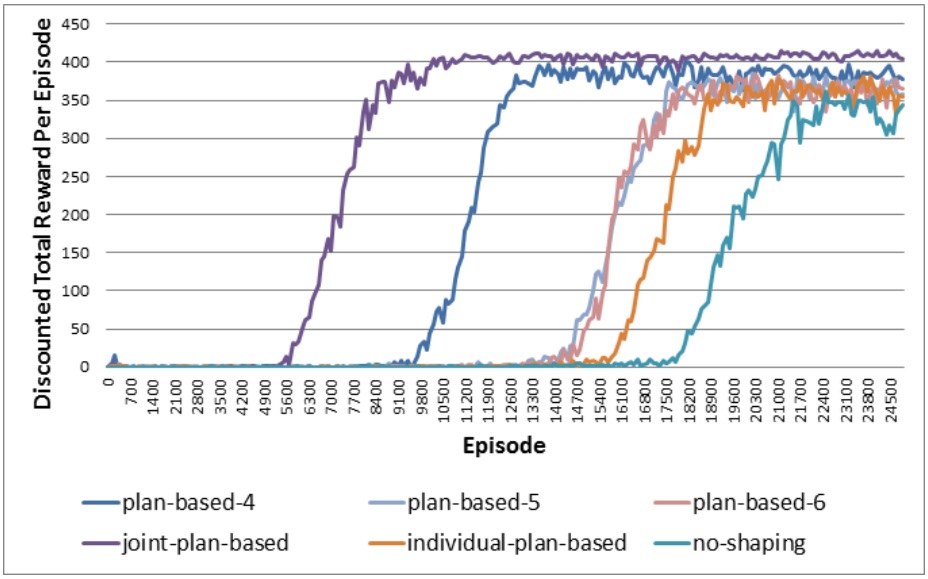


Figure 5.11: Optimistic Partial Plans

As can be seen in Figure 5.11, both the timing of the conflict and the amount of conflict affect the agents' time to convergence. Little difference in final performance is evident in these results as the agents are still benefiting from optimistic initialisation.

Alternatively, reducing the amount of incorrect knowledge can also affect the final perform-

ance of the agents if these agents use pessimistic initialisation as illustrated by Figure 5.12.

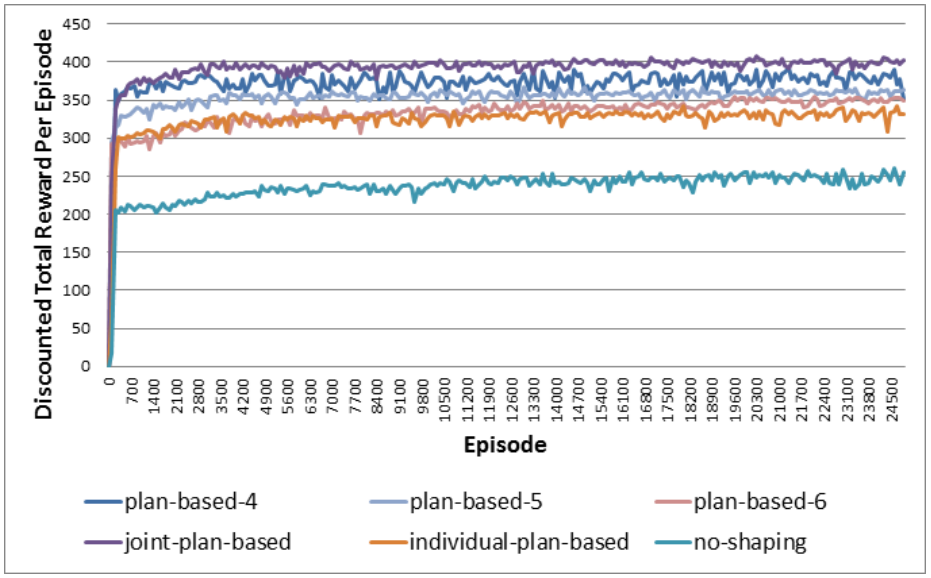


Figure 5.12: Pessimistic Partial Plans

However, to make plans with only partial overlaps, agents require some coordination or joint-knowledge that would not typically be available to multiple independent learners. If the process of improving knowledge could be automated, for instance with an agent starting an episode shaped by its individual plan and then refining the plan as it notices conflicts (i.e. plan steps that never occur), the agent may benefit from the improved knowledge and so alter its final performance without the need for optimistic initialisation.

5.5.3 Scaling Up

To further test these two approaches to overcoming conflicted knowledge, I tested them in the problem domain with six extra flags illustrated in Figure 5.3.

As shown in Figure 5.13, the results for agents guided by partial plans and pessimistic initialisation were again effectively the same as those in the original domain except for a slightly longer time to convergence as would be expected due to the larger state space.

The results for optimistic initialisation, however, took significantly longer. Figure 5.14 illustrates the results of just one complete run for this setting as performing any repeats would be impractical.

Whilst these results may be obtained quicker using function approximation or existing methods of improving optimistic exploration [Grześ and Kudenko, 2009a], they highlight the poor ability of optimistic initialisation to scale to large domains. Therefore, these experiments further support that automating the reduction of incorrect knowledge by an explicit belief revision mechanism would be more preferable than increasing exploration by optimistic initialisation

as the latter method does not direct exploration sufficiently. Instead optimistic initialisation encourages exploration to all states randomly taking considerable time to complete. A gradual refining of the plan used to shape an agent would encourage initially a conflicted joint-policy, which is still better than no prior knowledge, and then on each update exploration would be directed towards a more coordinated joint-plan.

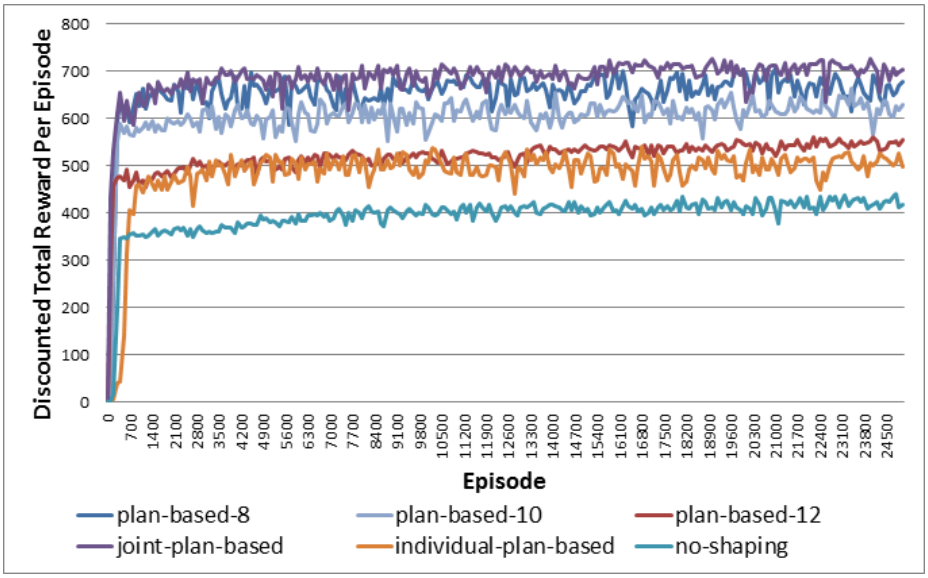


Figure 5.13: Pessimistic Partial Plans in the Scaled Up Problem Domain

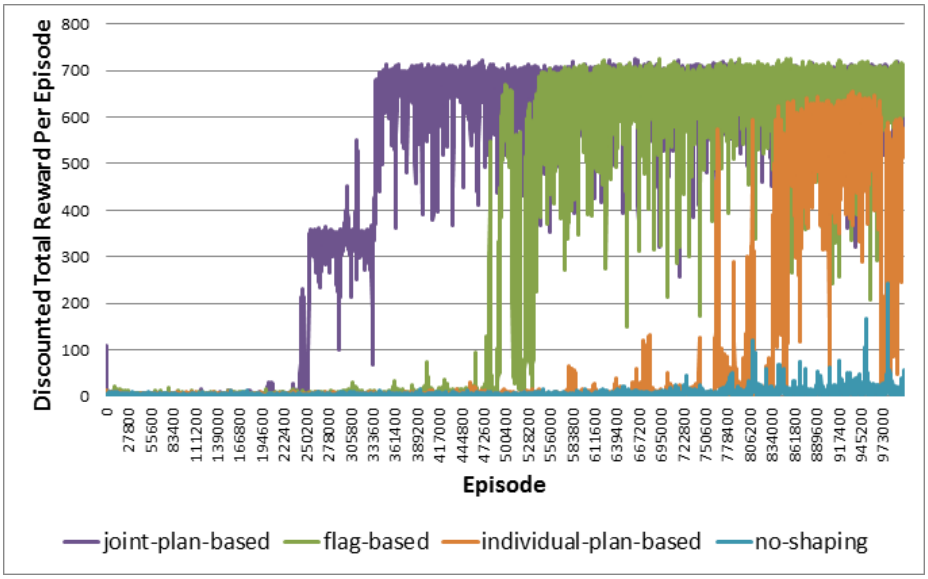


Figure 5.14: Optimistic Initialisation in the Scaled Up Problem Domain

5.6 Conclusion

In conclusion, I have demonstrated two approaches to using plan-based reward shaping in MARL. Ideally, plans are devised and coordinated centrally so each agent starts with prior knowledge of its own task allocation and the group can quickly converge to an optimal joint-policy.

Where this is not possible, due to agents unwilling to share information, plans made individually can shape the agent. Despite conflicts in the simultaneous execution of these plans, agents receiving individual-plan-based reward shaping still outperformed those without any prior knowledge in all experiments.

Furthermore, these conflicts can be overcome if shaping is combined with domain specific knowledge (i.e. flag-based reward shaping), the agent is initialised optimistically or the amount of conflicted knowledge is reduced. The first of these approaches requires a bespoke encoding of knowledge for any new problem domain and the second, optimistic initialisation, becomes impractical in larger domains.

Therefore, my research group has been motivated to pursue in ongoing work the approach of automatically improving knowledge by an explicit belief revision mechanism [Ethymiadis et al., 2013]. This approach will be discussed further in the next chapter amongst other areas of future work and the conclusion of this thesis.

CHAPTER 6

Conclusion and Future Work

To conclude, I recall from earlier the hypothesis of this thesis. Specifically:

> Given sufficient domain knowledge, multi-agent potential-based reward shaping can reduce the time a group of reinforcement learning agents need to learn a suitable behaviour and direct the agents towards convergence on a different joint policy whilst also guaranteed not to modify the agents' original intended goal.

In numerous experiments, ranging from 2 agent gridworld navigation tasks up to complex 5v4 robotic soccer simulations, I have demonstrated empirically that PBRS can significantly reduce the time needed for MARL to learn a suitable behaviour. Many of these experiments were also examples of cases where, because of the guidance given by PBRS, the agents converged to joint policies representative of higher performance than the joint policies learnt by the same agents without reward shaping.

Furthermore, I proved that the points of equilibrium that multiple learning agents can converge to is not altered by any number of them implementing PBRS. As the set of joint policies the agents may learn remains consistant, PBRS has not modified the agents' original intended goal. Their exploration, however, is altered. This effect is the cause for both the reductions in time needed to learn and the increases in final performance when given sufficient domain knowledge.

6.1 Summary of Contributions

To summarise, the most significant contributions of this thesis are:

Empirical Studies of MARL Algorithms with PBRS

Predominately in Chapter 3 but also in Section 4.5 and Chapter 5, a number of studies with a wide range of MARL algorithms were conducted to illustrate the effect of multi-agent PBRS. These all contributed to the conclusion that, given sufficient domain knowledge, PBRS could alter both the time multiple RL agents need to learn a suitable behaviour and which joint policy they learned.

Proof of Consistant Nash Equilibria when applying PBRS to MARL

In Chapter 4, I proved that, multi-agent PBRS does not alter the set of Nash Equilibria of the underlying MAS. Fruthermore, provided the potential function is static, multi-agent PBRS is still equivalent to Q-table initialisation. However, as discussed in Section 4.3, the combination of these results explains the ability for multi-agent PBRS to cause agents to learn different final joint policies to the same agents without PBRS.

Dynamic PBRS

In Section 4.4, I proved how the potential function could change whilst agents were learning and still maintain the same guarantees. This contribution significantly increased the space of reward functions guaranteed not to alter the Nash equilibria of the underlying SG. Furthermore, given that this approach breaks the equivalence to Q-table initialisation, dynamic PBRS provides the unique ability to guide agents by knowledge gained whilst they are learning without altering their intended goal.

Generalised Effect of PBRS

In Section 4.6 of Chapter 4, I generalised the effect of PBRS, concluding that it does not alter any property invariant to changes in absolute value. This conclusion explains the proven effect on both single and multi agent domains and will be useful for future applications of PBRS in novel contexts e.g. multi-objective reinforcement learning.

Multi-Agent Plan-Based Reward Shaping

In Chapter 5, I presented a novel extension of plan-based reward shaping to MARL. By automating the translation of multi-agent plans to a potential function, the benefits of multi-agent PBRS can be accessed by researchers not familiar with the details of how to implement PBRS correctly to ensure the theoretical results hold.

6.2 Limitations

Despite the contributions listed in the previous section, this thesis does have limitations. The most significant of which I will discuss further here.

Multi-Agent PBRS Can Slow the Rate of Learning and Reduce Final Performance

If guided by a poor heuristic, multi-agent PBRS' ability to alter the rate of learning and joint policy agents converge to can become a negative feature. This is unavoidable, but the damage can be reduced if the agents are given the ability to revise the potential function. This approach is motivated by Section 5.5.2, and will be discussed further in the following section on future.

Is Consistent Nash Equilibria Desirable?

Ideally, a reward transformation would guarantee to make the Pareto optimal policy of a SG become the sole Nash equilbria after the agents receive reward shaping. However, the aim of this thesis was not to find such a reward transformation. Instead my aim was to explore what the effect of PBRS on MARL is, which proved to be consistent Nash equilibria.

No Study of a Fully Competitive Environment

Most of the empirical studies covered in this thesis were purely cooperative problem domains, but none covered a fully competitive task. However, the RoboCup studies covered the more general case given that they included a mix of both competitive and cooperative elements. I hypothesise that if multi-agent PBRS was used in a fully competitive problem domain, it would give a competitive edge to the agent receiving PBRS provided the knowledge is suitable.

More Agent Studies

The largest study in this thesis included just 9 agents. Given that this was within the context of simulated RoboCup soccer, MARL to that scale is still very complex. Some of the experiments took weeks to complete all repeats. However, in theory, there is no reason these results won't have the same effect at the scale of hundreds or thousands of agents. Furthermore, all experiments presented in this thesis support the conclusion that scaling up the number of agents exaggerates the beneficial effect of PBRS.

Only considered knowledge based solutions

Finally, this study presumed prior knowledge of the problem domain. In single-agent RL, there are methods of using PBRS without prior knowledge and still increasing the rate of learning. No studies have been made on attempting this in MARL. This thesis does, however, contribute theoretically towards these methods as they typically rely on changing the potential function whilst the agent is learning. In the next section, amongst other possible areas for future work, I will discuss methods of automatic reward shaping that could be extended to MARL and a plausible, novel, multi-agent specific method.

6.3 Future Work

Finally, I will address a few open problem areas that I see potential benefit in exploring further.

Plan-Based Reward Shaping with Belief Revision

In my experiments attempting to overcome the gap in performance between individual-plan-based agents and joint-plan-based agents, I concluded that an automated process of improving the knowledge represented by the potential function is needed. The concept of the system, illustrated in Figure 6.1, is that agents guided by erroneous knowledge can discover steps in the high level plan it cannot complete, or notice facts not accounted for in the high level plan. By allowing the agents to alter this information in the high level knowledge base (KB) and replanning, the reward shaping can adjust over time to encourage a correct plan. Dynamic PBRS allows this method in theory, but to implement and test it would be a thesis in its own right [Ethymiadis et al., 2013; 2014].

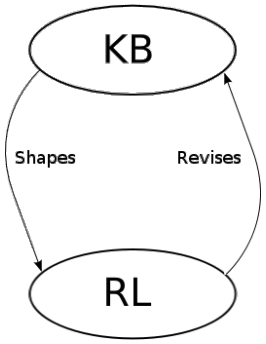


Figure 6.1: Plan-Based Reward Shaping with Belief Revision

Petri-Net-Based Reward Shaping

Alternatively, or potentially combined with the methods of belief revision, the choice of STRIPS as a representation of multi-agent knowledge could be questioned. Perhaps for MARL, reference nets [Köhler et al., 2001] (a form of petri net) may be more appropriate given their established history of use in modeling MAS [Celaya et al., 2009; Köhler et al., 2001; Moldt and Wienberg, 1997]. This would, however, require the sharing of information avoided by individual-plan-based agents but could be an interesting comparison to joint-plan-based agents.

Automatic Reward Shaping for MARL

As alluded to earlier, reward shaping has also been applied to single-agent problem domains where prior knowledge is unavailable by automating the assignment of potentials to states. Both of the following methods could plausibly be applied to multi-agent problem domains.

Marthi [2007] first generated an abstract MDP far simpler than the intended problem domain, solved this, and then used the value function of the abstract MDP's optimal policy to shape the

agent learning in the problem domain dramatically decreasing the time to convergence.

Grześ and Kudenko [2008] achieved a similar result by concurrently learning from the original MDP two value functions of different levels of discretisation. The more abstract value function is learnt quicker, due to the smaller state-action space, and is then used to shape the rewards used to learn the lower level value function from which the agent makes its action decisions.

Potential-Based Difference Rewards

Another plausible method of automating PBRS, specific to MARL, would be to use difference rewards as a potential function. As mentioned in Section 2.4.2, PBRS and difference rewards may not neccesarily be mutually exclusive concepts.

Difference rewards represent the generally applicable piece of MA knowledge that, in cooperative environments each agent should try to contribute to the global utility. Using this as a reward function has improved agents' performance in many problem domains [Tumer and Wolpert, 2000; Tumer and Khani, 2009; Agogino and Tumer, 2012; Agogino et al., 2012] but provides no theoretical guarantees regarding the effect on the underlying SG. Perhaps, if the difference reward is used as a potential function the same benefit in agents' performance can be achieved, whilst guaranteed not to alter the original intended goal and not needing any domain specific knowledge.

Since the original submission of this thesis, I have visited Oregon State University and tested this idea in collaboration with Logan Yliniemi, Professor Kagan Tumer and Dr. Daniel Kudenko [Devlin et al., 2014]. Agents using the counterfactual from difference rewards as a potential function outperformed agents without shaping or with many manual heuristics, but were outperformed by agents learning from difference rewards alone. However, this approach does have the theoretical guarantees of PBRS whilst the theoretical effect of difference rewards is currently unclear. Furthermore, shaping difference rewards by PBRS with manual heuristics can significantly improve the learning performance of the agents.

6.4 Closing Remarks

Deploying RL agents with no prior knowledge is rarely necessary. By imparting the knowledge you have of the task they are learning, the time they need to learn can be reduced and, in MAS, their final performance improved. I hope by now that the method of PBRS is both intuitive and easily implemented by any reader.

References

Abbeel, P., A. Coates, and A. Y. Ng (2010). Autonomous helicopter aerobatics through apprenticeship learning. *The International Journal of Robotics Research 29*(13), 1608–1639.

Abbeel, P. and A. Y. Ng (2004). Apprenticeship learning via inverse reinforcement learning. In *Proceedings of the twenty-first international conference on Machine learning*, pp. 1. ACM.

Agogino, A., C. H. Parker, and K. Tumer (2012). Evolving large scale uav communication systems. In *Proceedings of the Genetic and Evolutionary Computation Conference*, pp. 1023–1030.

Agogino, A. and K. Tumer (2012). A multiagent approach to managing air traffic flow. *Autonomous Agents and Multi-Agent Systems 24*(1), pp. 1–25.

Akchurina, N. (2009). Multiagent reinforcement learning: algorithm converging to Nash equilibrium in general-sum discounted stochastic games. In *Proceedings of The Eighth International Conference on Autonomous Agents and Multiagent Systems*, pp. 725–732.

Akyildiz, I., W. Lee, M. Vuran, and S. Mohanty (2006). NeXt generation/dynamic spectrum access/cognitive radio wireless networks: a survey. *Computer Networks 50*(13), pp. 2127–2159.

Asmuth, J., M. Littman, and R. Zinkov (2008). Potential-based shaping in model-based reinforcement learning. In *Proceedings of the Twenty-Third AAAI Conference on Artificial Intelligence*, pp. 604–609.

Babes, M., E. de Cote, and M. Littman (2008). Social reward shaping in the prisoner's dilemma. In *Proceedings of The Seventh Annual International Conference on Autonomous Agents and Multiagent Systems*, pp. 1389–1392.

Balch, T. (1997). Learning Roles: Behavioral Diversity in Robot Teams. In *AAAI Workshop on Multiagent Learning*.

Barto, A. and S. Mahadevan (2003). Recent advances in hierarchical reinforcement learning.

Discrete Event Dynamic Systems 13(4), pp. 341–379.

Bianchi, R., C. Ribeiro, and A. H. Costa (2004). Heuristically accelerated q-learning: A new approach to speed up reinforcement learning. In *Advances in Artificial Intelligence*, Volume 3171 of *Lecture Notes in Computer Science*, pp. 245–254.

Bianchi, R., C. Ribeiro, and A. H. Costa (2007). Heuristic selection of actions in multiagent reinforcement learning. In *Proceedings of the Sixteenth Twentieth International Joint Conference on Artifical Intelligence*, pp. 690–696.

Bianchi, R., C. Ribeiro, and A. H. Costa (2008). Accelerating autonomous learning by using heuristic selection of actions. *Journal of Heuristics 14*(2), pp.135–168.

Boutilier, C. (1999). Sequential optimality and coordination in multiagent systems. In *Proceedings of the International Joint Conference on Artificial Intelligence*, Volume 16, pp. 478–485.

Boutsioukis, G., I. Partalas, and I. Vlahavas (2012). Transfer learning in multi-agent reinforcement learning domains. In *Recent Advances in Reinforcement Learning*, pp. 249–260. Springer.

Bowling, M. and M. Veloso (2002). Multiagent learning using a variable learning rate. *Artificial Intelligence 136*(2), pp. 215–250.

Bradtke, M. and S. Duff (1995). Reinforcement learning methods for continuous-time markov decision problems. *Advances in Neural Information Processing Systems*, pp. 393–400.

Brafman, R. and C. Domshlak (2008). From one to many: Planning for loosely coupled multi-agent systems. In *Proceedings of the Eighteenth International Conference on Automated Planning and Scheduling*, pp. 28–35.

Brafman, R. and M. Tennenholtz (2003). R-max-a general polynomial time algorithm for near-optimal reinforcement learning. *The Journal of Machine Learning Research 3*, pp. 213–231.

Buşoniu, L., R. Babuska, and B. De Schutter (2008). A Comprehensive Survey of MultiAgent Reinforcement Learning. *IEEE Transactions on Systems Man & Cybernetics Part C Applications and Reviews 38*(2), pp. 156 – 172.

Buşoniu, L., B. De Schutter, and R. Babuska (2005). Multiagent reinforcement learning with adaptive state focus. In *Proceedings of the Seventeenth Belgium-Netherlands Conference on Artificial Intelligence*, pp. 35–42.

Busoniu, L., R. Babuska, B. De Schutter, and D. Ernst (2010). *Reinforcement learning and dynamic programming using function approximators*. CRC Press.

Celaya, J., A. Desrochers, and R. Graves (2009). Modeling and analysis of multi-agent systems using petri nets. *IEEE International Conference on Systems, Man and Cybernetics 4*(10), pp. 1439 – 1444.

Chakraborty, D. and P. Stone (2010). Convergence, Targeted Optimality, and Safety in Multiagent Learning. *Proceedings of the Twenty-Seventh International Conference on Machine Learning*, pp. 191–198.

Claus, C. and C. Boutilier (1998). The dynamics of reinforcement learning in cooperative multiagent systems. In *Proceedings of the National Conference on Artificial Intelligence*, pp. 746–752.

De Hauwere, Y. (2011). *Sparse Interations in Multi-Agent Reinforcement Learning*. Ph. D. thesis, Vrije Universitet Brussel.

De Hauwere, Y., S. Devlin, D. Kudenko, and A. Nowé (2012). Context sensitive reward shaping in a loosely coupled multi-agent system. *Adaptive and Learning Agents*.

De Hauwere, Y., S. Devlin, D. Kudenko, and A. Nowé (2013). Context sensitive reward shaping for sparse interaction mas. *Proceedings of the 25th Benelux Conference on Artificial Intelligence*.

De Hauwere, Y., P. Vrancx, and A. Nowé (2011). Solving delayed coordination problems in mas (extended abstract). In *The Tenth International Conference on Autonomous Agents and Multiagent Systems*, pp. 1115–1116.

De Weerdt, M., A. Ter Mors, and C. Witteveen (2005). Multi-agent planning: An introduction to planning and coordination. *Handouts of the European Agent Summer*.

Decker, K. and V. Lesser (1992). Generalizing the partial global planning algorithm. *International Journal of Intelligent and Cooperative Information Systems 2*(2), pp. 319–346.

Devlin, S., M. Grześ, and D. Kudenko (2009). Reinforcement learning in robocup keepaway with partial observability. In *IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology.*, pp. 201–208.

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

Devlin, S. and D. Kudenko (2011). Theoretical considerations of potential-based reward shaping for multi-agent systems. In *Proceedings of The Tenth Annual International Conference on Autonomous Agents and Multiagent Systems*, pp. 225–232.

Devlin, S. and D. Kudenko (2012). Dynamic potential-based reward shaping. In *Proceedings of The Eleventh Annual International Conference on Autonomous Agents and Multiagent Systems*, pp. 433–440.

Devlin, S., L. Yliniemi, K. Tumer, and D. Kudenko (2014). Potential-based difference rewards for multiagent reinforcement learning. In *Proceedings of The Thirteenth Annual International Conference on Autonomous Agents and Multiagent Systems (To Appear)*.

Dixit, A., S. Skeath, and D. Reiley (2004). *Games of strategy*. WW Norton New York.

Durfee, E. and V. Lesser (1987). Planning coordinated actions in dynamic domains. Technical report, University of Massachusetts.

Eck, A., S. Leen-Kiat, S. Devlin, and D. Kudenko (2013). Potential-based reward shaping for pomdps (extended abstract). In *Proceedings of The Twelfth Annual International Conference on Autonomous Agents and Multiagent Systems*, pp. 1123–1124.

Ernst, D., P. Geurts, and L. Wehenkel (2005). Tree-based batch mode reinforcement learning. In *Journal of Machine Learning Research*, pp. 503–556.

Ethymiadis, K., S. Devlin, and D. Kudenko (2013). Overcoming erroneous domain knowledge in plan-based reward shaping (extended abstract). In *Proceedings of The Twelfth Annual International Conference on Autonomous Agents and Multiagent Systems*, pp. 1245–1246.

Ethymiadis, K., S. Devlin, and D. Kudenko (2014). Knowledge revision for reinforcement learning with abstract mdps. (extended abstract). In *Proceedings of The Thirteenth Annual International Conference on Autonomous Agents and Multiagent Systems (To Appear)*.

Fachantidis, A., I. Partalas, G. Tsoumakas, and I. Vlahavas (2012). Transferring task models in reinforcement learning agents. *Neurocomputing*.

Fasli, M. (2006). *Agent technology for e-commerce*. John Wiley & Sons.

Fudenberg, D. and J. Tirole (1991). *Game Theory*. Cambridge, MA: MIT Press.

Grześ, M. (2010). *Improving Exploration in Reinforcement Learning through Domain Knowledge and Parameter Analysis*. Ph. D. thesis, University of York.

Grześ, M. and D. Kudenko (2008). Multigrid Reinforcement Learning with Reward Shaping. *Proceedings of the International Conference on Artificial Neural Networks*, pp. 357–366.

Grześ, M. and D. Kudenko (2008). Plan-based reward shaping for reinforcement learning. In *Proceedings of the Fourth IEEE International Conference on Intelligent Systems*, pp. 22–29.

Grześ, M. and D. Kudenko (2009a). Improving optimistic exploration in model-free reinforcement learning. *Adaptive and Natural Computing Algorithms*, pp. 360–369.

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

Grześ, M. and D. Kudenko (2010). Online learning of shaping rewards in reinforcement learning. *Artificial Neural Networks-ICANN 2010*, pp. 541–550.

Haykin, S. (2005). Cognitive radio: brain-empowered wireless communications. *IEEE Journal on Selected Areas in Communications 23*(2), pp. 201–220.

Hengst, B. (2012). Hierarchical approaches. In M. Wiering and M. Otterlo (Eds.), *Reinforcement Learning*, Volume 12 of *Adaptation, Learning, and Optimization*, pp. 293–323. Springer Berlin Heidelberg.

Hu, J. and M. Wellman (2003). Nash Q-learning for general-sum stochastic games. *The Journal of Machine Learning Research 4*, pp. 1039–1069.

Iscen, A. and U. Erogul (2008). A new perspective to the keepaway soccer: the takers. In *Proceedings of the Seventh International Conference on Autonomous Aagents and Multiagent Systems*, pp. 1341–1344.

Kaelbling, L., M. Littman, and A. Cassandra (1998). Planning and acting in partially observable stochastic domains. *Artificial Intelligence 101*, pp. 99–134.

Kalyanakrishnan, S. and P. Stone (2010). Learning complementary multiagent behaviors: a case

study. In *RoboCup 2009: Robot Soccer World Cup XIII*, Volume 5949 of *Lecture Notes in Computer Science*, pp. 153–165. Springer Berlin / Heidelberg.

Kapetanakis, S. and D. Kudenko (2002). Reinforcement learning of coordination in cooperative multi-agent systems. In *Proceedings of the National Conference on Artificial Intelligence*, pp. 326–331.

Kapetanakis, S. and D. Kudenko (2004). Reinforcement learning of coordination in heterogeneous cooperative multi-agent systems. In *Proceedings of the Second International Conference on Autonomous Agents and Multiagent Systems*, pp. 119–131.

Köhler, M., D. Moldt, and H. Rölke (2001). Modelling the structure and behaviour of petri net agents. *Applications and Theory of Petri Nets*, pp. 224–241.

Lagoudakis, M. G. and R. Parr (2003). Least-squares policy iteration. *The Journal of Machine Learning Research 4*, 1107–1149.

Lange, S., T. Gabel, and M. Riedmiller (2012). Batch reinforcement learning. In M. Wiering and M. Otterlo (Eds.), *Reinforcement Learning*, Volume 12 of *Adaptation, Learning, and Optimization*, pp. 45–73. Springer Berlin Heidelberg.

Lau, Q. P., M. L. Lee, and W. Hsu (2011). Distributed coordination guidance in multi-agent reinforcement learning. In *Proceedings of the Twenty-Third IEEE International Conference on Tools with Artificial Intelligence*, pp. 456–463.

Lau, Q. P., M. L. Lee, and W. Hsu (2012). Coordination guided reinforcement learning. In *Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems*, pp. 215–222.

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

Lauer, M. and M. Riedmiller (2000). An algorithm for distributed reinforcement learning in cooperative multi-agent systems. In *Proceedings of the Seventeenth International Conference on Machine Learning*, pp. 535–542. Morgan Kaufmann.

Lazaric, A. (2012). Transfer in reinforcement learning: A framework and a survey. In *Reinforcement Learning*, pp. 143–173. Springer.

Littman, M. (1994). Markov games as a framework for multi-agent reinforcement learning. In *Proceedings of the Eleventh International Conference on Machine Learning*, pp. 157–163.

Littman, M. (2001). Friend-or-foe Q-learning in general-sum games. In *Proceedings of the Eighteenth International Conference on Machine Learning*, pp. 322–328.

Makar, R., S. Mahadevan, and M. Ghavamzadeh (2001). Hierarchical multi-agent reinforcement learning. In *Proceedings of the Fifth International Conference on Autonomous agents*, pp. 246–253. ACM.

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

Matarić, M. (1994). Reward functions for accelerated learning. In *Proceedings of the Eleventh*

International Conference on Machine Learning, pp. 181–189.

Matarić, M. (1997). Reinforcement learning in the multi-robot domain. *Autonomous Robots 4*(1), pp. 73–83.

Min, H., J. Zeng, J. Chen, and J. Zhu (2008). A Study of Reinforcement Learning in a New Multiagent Domain. In *IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology.*, Volume 2.

Mitchell, T. (1997). Machine learning. *Mac Graw Hill*.

Moldt, D. and F. Wienberg (1997). Multi-agent-systems based on coloured petri nets. *Application and Theory of Petri Nets*, pp. 82–101.

Myerson, R. B. (1990). Game theory: Analysis of conflict. *Cambridge/Mass*.

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

Ng, A. (2003). *Shaping and policy search in reinforcement learning*. Ph. D. thesis, University of California, Berkeley.

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

Nissim, R., R. Brafman, and C. Domshlak (2010). A General, Fully Distributed Multi-Agent Planning Algorithm. In *Proceedings of The Ninth Annual International Conference on Autonomous Agents and Multiagent Systems*.

Nowé, A., P. Vrancx, and Y.-M. De Hauwere (2012). Game theory and multi-agent reinforcement learning. In *Reinforcement Learning*, pp. 441–470. Springer.

Price, B. and C. Boutilier (2003). Accelerating reinforcement learning through implicit imitation. *J. Artif. Intell. Res.(JAIR) 19*, 569–629.

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

Randløv, J. and P. Alstrom (1998). Learning to drive a bicycle using reinforcement learning and shaping. In *Proceedings of the Fifteenth International Conference on Machine Learning*, pp. 463–471.

Ripley, B. (2008). *Pattern recognition and neural networks*. Cambridge Univ Pr.

Rummery, G. and M. Niranjan (1994). On-line q-learning using connectionist systems. Technical report, University of Cambridge, Department of Engineering.

Russell, S. and A. L. Zimdars (2003). Q-decomposition for reinforcement learning agents. In *Proceedings of the Twentieth International Conference on Machine Learning*, pp. 656–663.

Shapley, L. S. (1953). Stochastic games. *Proceedings of the National Academy of Sciences of the United States of America 39*(10), 1095.

Shoham, Y. and K. Leyton-Brown (2008). *Multiagent systems: algorithmic, game-theoretic, and logical foundations*. Cambridge University Press.

Shoham, Y., R. Powers, and T. Grenager (2007). If multi-agent learning is the answer, what is

the question? *Artificial Intelligence 171*(7), pp. 365–377.

Shoham, Y. and M. Tennenholtz (1995). On social laws for artificial agent societies: off-line design. *Artificial Intelligence 73*(1-2), pp. 231–252.

Stone, P. (2007). Multiagent learning is not the answer. It is the question. *Artificial Intelligence 171*(7), pp. 402–405.

Stone, P., G. Kuhlmann, M. E. Taylor, and Y. Liu (2006). Keepaway soccer: From machine learning testbed to benchmark. In *RoboCup-2005: Robot Soccer World Cup IX*, Volume 4020, pp. 93–105. Springer Verlag.

Stone, P. and R. S. Sutton (2001). Scaling reinforcement learning toward robocup soccer. In *Proceedings of the Eighteenth International Conference on Machine Learning*, pp. 537–544.

Stone, P., R. S. Sutton, and G. Kuhlmann (2005). Reinforcement learning for RoboCup-soccer keepaway. *Adaptive Behavior 13*(3), pp. 165–188.

Stone, P. and M. Veloso (1999). Team-partitioned, opaque-transition reinforcement learning. In *Proceedings of the Third Annual Conference on Autonomous Agents*, pp. 206–212. ACM.

Stone, P. and M. Veloso (2000). Multiagent systems: A survey from a machine learning perspective. *Autonomous Robots 8*(3), pp. 345–383.

Sutton, R. S. and A. G. Barto (1998). *Reinforcement Learning: An Introduction*. MIT Press.

Sutton, R. S., D. Precup, and S. Singh (1999). Between mdps and semi-mdps: A framework for temporal abstraction in reinforcement learning. *Artificial intelligence 112*(1), pp. 181–211.

Tamar, A., D. D. Castro, and R. Meir (2012). Integrating a partial model into model free reinforcement learning. *The Journal of Machine Learning Research 13*, pp. 1927–1966.

Tan, M. (1993). Multi-Agent Reinforcement Learning: Independent vs. Cooperative Agents. In *Proceedings of the Tenth International Conference on Machine Learning*, Volume 337.

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

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

Thorndike, E. L. and R. S. Woodworth (1901). The influence of improvement in one mental function upon the efficiency of other functions.

Tumer, K. and N. Khani (2009). Learning from actions not taken in multiagent systems. *Advances in Complex Systems 12*(04), pp. 455–473.

Tumer, K. and D. Wolpert (2000). Collective Intelligence and Braess' Paradox. In *Proceedings of the National Conference on Artificial Intelligence*, pp. 104–109.

Tuyls, K. and G. Weiss (2012). Multiagent learning: Basics, challenges, and prospects. *AI Magazine 33*(3), 41.

Vamplew, P., R. Dazeley, A. Berry, R. Issabekov, and E. Dekker (2011). Empirical evaluation methods for multiobjective reinforcement learning algorithms. *Machine learning 84*(1-2), pp. 51–80.

Vrancx, P., Y.-M. De Hauwere, and A. Nowé (2011). Transfer learning for multi-agent coordination. In *Proceedings of The Third International Conference on Agents and Artificial Intelligence*, pp. 263–272.

Wang, X. and T. Sandholm (2003). Reinforcement learning to play an optimal Nash equilibrium in team Markov games. *Advances in Neural Information Processing Systems*, pp. 1603–1610.

Watkins, C. and P. Dayan (1992). Q-learning. *Machine learning 8*(3), pp. 279–292.

Weiss, G. (Ed.) (2013). *Multiagent systems* (Second ed.). The MIT press.

Wellman, M. and J. Hu (1998). Conjectural equilibrium in multiagent learning. *Machine Learning 33*(2), pp. 179–200.

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

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

Wolpert, D. and K. Tumer (1999). An introduction to collective intelligence. Technical Report cs.LG/9908014, NASA Ames Research Center.

Wooldridge, M. (2002). *An Introduction to MultiAgent Systems*. John Wiley and Sons.

Ziparo, V. (2005). Multi-Agent Planning. Technical report, University of Rome.

Index

Action Selection
    ϵ-Greedy, 16, 39, 44, 48
    Boltzmann/Soft-Max, 16
    Greedy, 16
Agent, 14
Apprenticeship Learning, 37
ASFQ-Learning, 26, 40

Batch Reinforcement Learning, 21

Coordination Guided Reinforcement Learning, 35

Difference Rewards, 31, 101
Discount Factor, 18
Distributed Q-Learning, 26, 41
Dynamic Programming, 17

Eligibility Traces, 19, 39
Episode, 19

FCQ-Learning, 26, 40
Function Approximation, 20

General-Sum Games, 24

Heuristic Selection of Actions, 36
Hierarchical Reinforcement Learning, 34

Imitation Learning, 37
IPM, 36

Joint-Action Learners, 25, 41, 75

Learning Rate, 18
Look-Ahead Advice, 29
Look-Back Advice, 29

Markov Decision Processes, 16
Markov property, 16
Monte Carlo Methods, 19
Multi-Agent Planning, 79
    Centralised, 80
    Decentralised, 80
Multi-Agent Systems, 21
Multi-Objective Reinforcement Learning, 75
Multiple Independent Learners, 25, 39

Nash Equilibrium, 24

Optimistic Initialisation, 15

Pareto Optimality, 24, 75, 99
Partially-Observable MDP, 17, 75
Pessimistic Initialisation, 15, 41
Plan-based Reward Shaping, 30
Policy, 15

Potential-Based Advice, 28
Potential-Based Reward Shaping, 27

Q-Learning, 18, 39, 41

Random Initialisation, 15
Reinforcement Learning, 14

SARSA, 18, 44, 48
Semi-MDP, 16, 75
State-Space Explosion, 20, 21, 23
Stochastic Games, 23

Tabular State Representations, 20
Temporal Difference Learning, 18
Tile Coding, 20, 44, 48
Transfer Learning, 36

Update Rule, 18

Value Function, 15
Value Function Initialisation, 33

WoLFQ-PHC, 26

List of Symbols and Acronyms

α Learning Rate

γ Discount Factor

λ Decay Rate of Eligibility Traces

Φ Potential Function

π Policy

$Q(s, a)$ Value Function

ASFQ-Learning Adaptive State Focus Q-learning

FCQ-Learning Future Coordinating Q-learning

MARL Multi-Agent Reinforcement Learning

MAS Multi-Agent System

MDP Markov Decision Process

PBRS Potential Based Reward Shaping

POMDP Partially-Observable Markov Decision Process

RL Reinforcement Learning

SG Stochastic Game

SMDP Semi-Markov Decision Process

WoLFQ-PHC Win or Learn Fast Q - Policy Hill Climb