Improving Exploration in Reinforcement Learning

through Domain Knowledge and Parameter Analysis

MAREK GRZEŚ

Ph.D. Thesis
This thesis is submitted in partial fulfilment of the requirements for the degree of Doctor of Philosophy.

THE UNIVERSITY of York

Artificial Intelligence Group
Department of Computer Science
United Kingdom

March 2010

For all those most important to me people, for whom I did not have enough time while doing this research, but without whom this would never have been possible.

Abstract

This thesis presents novel work on how to improve exploration in reinforcement learning using domain knowledge and knowledge-based approaches to reinforcement learning. It also identifies novel relationships between the algorithms' and domains' parameters and the exploration efficiency.

The goal of solving reinforcement learning problems is to learn how to execute actions in order to maximise the long term reward. Solving this type of problems is a hard task when real domains of realistic size are considered because the state space grows exponentially with each state feature added to the representation of the problem.

In its basic form, reinforcement learning is *tabula rasa*, i.e. it starts learning with very limited knowledge about the domain. One of the ways of improving the performance of reinforcement learning is the principled use of domain knowledge. Knowledge is successful in related branches of artificial intelligence, and it is becoming increasingly important in the area of reinforcement learning as well. Reinforcement learning algorithms normally face the problem of deciding whether to execute explorative of exploitative actions, and the paramount goal is to limit the number of executions of suboptimal explorative actions. In this thesis, it is shown how domain knowledge and understanding of algorithms' and domains' properties can help to achieve this.

Exploration is an immensely complicated process in reinforcement learning and is influenced by numerous factors. This thesis presents a new range of methods for dealing more efficiently with the exploration-exploitation dilemma which is a crucial issue of applying reinforcement learning in practice. Reward shaping was used in this research as a well established framework for incorporating procedural knowledge into model-free reinforcement learning. Two new ways of obtaining heuristics for potential-based shaping were introduced and evaluated: high level symbolic knowledge and the application of different hypothesis spaces to learn the heuristic.

These techniques open the way to improve reinforcement learning via reward shaping in situations when there is no information about the potential function. In the work on potential-based reward shaping, the actual shaping reward under different conditions was also specified and empirically evaluated. In the context of model-based reinforcement learning, a novel technique to incorporate knowledge into the initial MDP-models was proposed, evaluated, and proven to meet properties of PAC-MDP learning. One of the important factors which influence exploration in reinforcement learning is the concept of eligibility traces. The last part of this research focused on a detailed analysis of how eligibility traces influence exploration under a multitude of conditions.

The contribution of this thesis shows how to learn the potential function for reward shaping when it is not available, and also shows formal specification of the actual shaping reward under a multitude of conditions. It also shows how to use partial knowledge about effects of actions to create knowledge-based and theoretically correct implementations of PAC-MDP learning. Novel relationships between eligibility traces and exploration efficiency were also identified. Findings of this thesis extend current understanding and improve the exploration efficiency of reinforcement learning algorithms.

Contents

List of Tables . . . 9
List of Figures . . . 13
Acknowledgements . . . 14
Declaration . . . 16

1 Introduction and Motivation **18**
1.1 Reinforcement Learning . . . 18
1.2 Topic of the Thesis . . . 19
1.3 Motivation . . . 20
1.4 Hypothesis . . . 21
1.5 Goals . . . 22
1.6 Overview of the Thesis . . . 22
1.7 Summary of Achievements . . . 23
1.8 Structure of the Thesis . . . 24

2 Background and Field Review **25**
2.1 Reinforcement Learning . . . 25
2.1.1 Standard Reinforcement Learning Problem . . . 26
2.1.2 Direct Policy Search Approach . . . 28
2.1.3 Markov Decision Processes . . . 28
2.1.4 Temporal Difference Learning . . . 29
2.2 Exploration in Reinforcement Learning . . . 31
2.3 Eligibility Traces . . . 33
2.4 Reward Shaping . . . 34
2.5 Function Approximation . . . 36
2.6 Symbolic Planning . . . 37
2.7 Domain Knowledge in Reinforcement Learning . . . 38
2.8 Motivating Remarks . . . 39
2.8.1 Why does reward shaping influence exploration? . . . 40
2.8.2 Why do eligibility traces influence exploration? . . . 40
2.8.3 What is reward shaping for? . . . 40
2.8.4 Can we not do something easier instead of reward shaping? . . . 40
2.8.5 Why is exploration according to the current policy necessary? . . . 41

3 Plan-based Reward Shaping **42**
3.1 Introduction . . . 42
3.2 Plan-based Reward Shaping . . . 44
3.2.1 The Potential Function from the STRIPS Plan . . . 44
3.2.2 Manual Definition of the Potential Function . . . 46
3.2.3 The Potential Function from Abstract MDP . . . 46
3.3 Experimental Domain . . . 47
3.4 Evaluated Algorithms and Parameters . . . 48
3.5 Potential Function for Experimental Domain . . . 49
3.5.1 Low Level Model . . . 49
3.5.2 High Level Representation . . . 49
3.5.3 High Level Planning Problems . . . 51
3.6 Empirical Results . . . 52
3.6.1 Pessimistic Exploration . . . 53
3.6.1.1 Results with Optimal Plans . . . 53
3.6.1.2 Results with Sub-optimal Plans . . . 56
3.6.2 Optimistic Exploration . . . 59
3.6.2.1 Results with Optimal Plans . . . 60
3.6.2.2 Results with Sub-optimal Plans . . . 61
3.7 Plan-based Reward Shaping with Function Approximation . . . 64
3.8 Plan-based Reward Shaping in Model-based Learning . . . 67
3.9 Summary and Discussion . . . 68

4 Reward Shaping and Mixed Resolution Function Approximation **71**
4.1 Introduction . . . 71
4.2 Background . . . 73
4.2.1 Value Function Approximation with Tile Coding . . . 73
4.2.2 Reward Shaping . . . 74
4.3 Mixed Resolution Tile Coding . . . 75
4.4 Learning the Potential Function for Reward Shaping . . . 75
4.4.1 Related Work . . . 75
4.4.2 A Novel Algorithm . . . 77
4.4.3 Use of Tilings . . . 78
4.4.4 Properties of the Algorithm . . . 78
4.5 Experimental Design . . . 80
4.6 Experimental Domains . . . 81
4.6.1 Mountain Car . . . 81
4.7 Car Parking . . . 82
4.7.1 Boat . . . 84
4.8 Results . . . 85
4.8.1 Mountain Car . . . 85
4.8.2 Car Parking . . . 87
4.8.3 Boat . . . 88
4.9 Summary and Discussion . . . 94

5 Analysis of Reward Shaping 96
5.1 Introduction . . . 96
5.2 Reward Shaping . . . 97
5.3 Reward and the Discount Factor . . . 98
5.4 Running Examples and Algorithms . . . 99
5.4.1 Random Walk . . . 99
5.4.2 Maze . . . 99
5.4.3 S-maze . . . 100
5.5 Positive and Negative Potential Functions and $\gamma = 1$ . . . 101
5.6 Positive and Negative Potential Functions and $\gamma < 1$ . . . 104
5.6.1 The Potential Function, Discount Factor, and the Actual Shaping Reward 104
5.6.1.1 Positive Potential Function . . . 105
5.6.1.2 Negative Potential Function . . . 106
5.6.1.3 Positive and Negative Potential Functions . . . 107
5.6.2 An Empirical Comparison of Positive and Negative Potential Functions . 108
5.6.2.1 Evaluation with the Step Reward R_s . . . 109
5.6.2.2 Evaluation with the Goal Reward R_g . . . 110
5.7 Modified Reward Shaping Evaluation . . . 113
5.7.1 Empirical Tests with R_s . . . 116
5.7.2 Empirical Tests with R_g . . . 117
5.8 The Potential Function in Multi-Goal Domains . . . 119
5.9 Summary and Discussion . . . 120

6 PAC-MDP Learning with Knowledge-based Admissible Models 123
6.1 Introduction . . . 123
6.2 PAC-MDP Algorithms . . . 124
6.3 Near Bayesian Learning . . . 125
6.4 Domain Knowledge and Admissible Models . . . 126
6.4.1 Optimistic Determinization . . . 126
6.4.2 Free Space Assumption . . . 128
6.4.3 Maximal Probability . . . 129
6.5 PAC-MDP Learning with Admissible Models . . . 130
6.5.1 The AO Model . . . 131
6.5.2 The FSA Model . . . 131
6.5.3 Maximal Probability Knowledge . . . 131
6.6 Experimental Validation . . . 132
6.7 Results . . . 133
6.8 Summary and Discussion . . . 139

7 Analysis of Exploration with Eligibility Traces 141
7.1 Introduction . . . 142
7.1.1 Eligibility Traces . . . 142
7.1.2 Motivation . . . 143
7.2 Experimental Design . . . 144
7.2.1 Parameters . . . 145
7.2.2 Experimental Domain . . . 146
7.3 Goal-based Rewards . . . 147
7.3.1 Pessimistic Exploration . . . 148
7.3.2 Optimistic Exploration . . . 150

7.3.3 Resetting the Trace . . . 151
7.3.4 Changing Learning Rates . . . 152
7.3.4.1 Pessimistic Exploration . . . 152
7.3.4.2 Optimistic Exploration . . . 154
7.3.5 Changing Exploration Rates . . . 155
7.3.5.1 Pessimistic Exploration . . . 156
7.3.5.2 Optimistic Exploration . . . 157
7.4 Step Rewards . . . 158
7.4.1 Optimistic Exploration . . . 159
7.4.2 Semi-Optimistic Exploration . . . 159
7.4.3 Resetting the Trace . . . 163
7.4.4 Changing Learning Rates . . . 163
7.4.5 Changing Exploration Rates . . . 165
7.5 Further Analysis on Random Walk . . . 166
7.6 Non-converging Settings . . . 166
7.7 Summary and Discussion . . . 168

8 Conclusion and Future Work 171
8.1 Overview . . . 171
8.2 Hypothesis . . . 172
8.2.1 Brief Summary of What Was Achieved . . . 172
8.3 Summary of Main Contributions and Findings . . . 173
8.3.1 Plan-based Reward Shaping . . . 173
8.3.2 Reward Shaping and Mixed Resolution Function Approximation . . . 175
8.3.3 Analysis of Reward Shaping . . . 176
8.3.4 PAC-MDP Learning with Knowledge-based Admissible Models . . . 178
8.3.5 Analysis of Exploration with Eligibility Traces . . . 179
8.4 Limitations . . . 181
8.5 Future Work . . . 182
8.6 Final Remarks . . . 183

References 184

Index 193

Citation Index 195

List of Symbols 198

List of Tables

5.1 The influence of the type of the additive potential function and of the discount factor, γ, on the actual shaping reward when conditions are violated. 108
5.2 The influence of the type of the multiplicative potential function and of the discount factor, γ, on the actual shaping reward when conditions are violated. 108
5.3 Shaping rewards from positive and negative potential functions on RW-16 with $\gamma = 0.9$. 111

6.1 The use of knowledge-based admissible models. 131

7.1 The length of the eligibility trace which is computed according to the following function: $f(\gamma, \lambda) = \lfloor \log_{(\gamma\lambda)} 10^{-9} \rfloor$. Values shown in this table are domain independent since the length of the trace depends on γ and λ only. 150

List of Figures

2.1 The standard reinforcement learning problem (Sutton & Barto 1998). 27

3.1 The map of the maze problem. S is the start position and G the goal position. Capital letters represent flags which can be collected. 48
3.2 The optimal STRIPS plan. 53
3.3 SARSA results with all reward shaping types, correct plans and pessimistic exploration. 54
3.4 The histogram which presents how many times abstract states were entered during first 50 iterations of the single run of the SARSA algorithm. 55
3.5 SARSA learning with pessimistic exploration when the planner does not know about the rigid fact `(next-to roomE roomC)`. 56
3.6 SARSA learning with pessimistic exploration when the planner assumes the existence of the transition E to B, i.e. the rigid fact `(next-to roomE roomC)` which does not exist in the actual environment. 57
3.7 SARSA learning with pessimistic exploration when the planner does not know about flag B. 58
3.8 SARSA learning with pessimistic exploration when the plan is not correct (wrong sequence of abstract states). 59
3.9 SARSA results with optimistic exploration and all reward shaping types. 60
3.10 SARSA results with optimistic exploration and all reward shaping types. 61
3.11 SARSA learning with optimistic exploration when the planner does not know about the rigid fact `(next-to roomE roomC)`. 62
3.12 SARSA learning with optimistic exploration when the planner assumes the existence of the transition E to B, i.e. the rigid fact `(next-to roomE roomC)` which does not exist in the actual environment. 63
3.13 SARSA learning with optimistic exploration when the planner assumes the existence of the transition E to B, i.e. the rigid fact `(next-to roomE roomC)` which does not exist in the actual environment. 63
3.14 SARSA learning with optimistic exploration when the planner does not know about flag B, i.e. when one of goal predicates is missing. 64
3.15 SARSA learning with optimistic exploration when the planner does not know about flag B, i.e. when one of goal predicates is missing. 65

3.16 SARSA learning with optimistic exploration when the plan is not correct (wrong sequence of abstract states). 65

4.1 Tile coding examples with a different resolution. Three tilings with tiles of three units in a) and six units in b). 74
4.2 The mountain car task (Sutton & Barto 1998). 81
4.3 The car parking task (Cichosz 1995). The domain state is described by $\langle x_t, y_t, \theta_t \rangle$. 83
4.4 The boat task (Jouffe 1998). 84
4.5 Results on the mountain car problem ($\lambda = 0.7$). The top graph shows the first 25×10^2 episodes, and the bottom graph shows the remaining 25×10^2 episodes. 86
4.6 Results on the mountain car problem ($\lambda = 0$). 87
4.7 The car parking problem with original settings ($\lambda = 0.7$). 88
4.8 The car parking problem with the tripled size of the working area ($\lambda = 0.7$). . . . 89
4.9 The car parking problem with original settings ($\lambda = 0$). 89
4.10 The car parking problem with the tripled size of the working area ($\lambda = 0$). 90
4.11 The boat problem with 5 actions ($\lambda = 0.7$). 90
4.12 The boat problem with 20 actions ($\lambda = 0.7$). 91
4.13 The boat problem with 40 actions ($\lambda = 0.7$). 91
4.14 The boat problem with 5 actions ($\lambda = 0$). 92
4.15 The boat problem with 20 actions ($\lambda = 0$). 92
4.16 The boat problem with 40 actions ($\lambda = 0$). 93

5.1 The random walk domain. 99
5.2 The stochastic navigation maze domain (Maze). 100
5.3 The stochastic navigation maze task - S-maze (Sutton & Barto 1998: Figure 9.5). 101
5.4 Results on RW-64 with positive and negative potential functions. 102
5.5 Results on S-maze with positive and negative potential functions. 102
5.6 Results on RW-64 with a negative potential function and scaling of the shaping reward. 103
5.7 Results on Maze with a positive potential function and scaling of the shaping reward. 104
5.8 Results on S-maze with a positive potential function and scaling of the shaping reward. 105
5.9 Results on RW-32 with $\gamma = 0.95$, R_s, and positive and negative potential functions. 109
5.10 Results on RW-40 with $\gamma = 0.95$, R_s, and positive and negative potential functions. 110
5.11 Results on Maze with $\gamma = 0.95$, R_s, and positive and negative potential functions. 112
5.12 Results on S-maze with $\gamma = 0.99$, R_s, and positive and negative potential functions. 112
5.13 Results on RW-128 with $\gamma = 0.95$, R_g, and the positive potential function only (negative does not converge). 113
5.14 Results on Maze with $\gamma = 0.95$, R_g, and positive and negative potential functions. 114
5.15 Results on RW-40 with $\gamma = 0.95$, Equation 5.17, R_s, and the positive and negative potential functions. 117
5.16 Results on RW-128 with $\gamma = 0.95$, Equation 5.17, R_g, and the positive and negative potential functions. 117
5.17 Results on Maze with $\gamma = 0.95$, Equation 5.17, R_s, and the positive and negative potential functions. 118
5.18 Results on RW-128 with $\gamma = 0.95$, Equation 5.17, R_g, and the positive and negative potential functions. 118

5.19 Results on Maze with $\gamma = 0.95$, Equation 5.17, R_g, and positive and negative potential functions. . . . 119
5.20 A simple MDP where the straightforward application of potential-based reward shaping of Ng et al. (1999) leads to a different policy than non-shaped learning. . 119

6.1 AO knowledge and $\varrho = 1$. . . . 134
6.2 AO knowledge and $\varrho = 0.8$. . . . 135
6.3 FSA knowledge and $\varrho = 1$. . . . 135
6.4 FSA knowledge and $\varrho = 0.8$. . . . 136
6.5 Default versions of tested algorithms without AO and FSA knowledge. . . . 137
6.6 AO knowledge and $\varrho = 0.8$. . . . 137
6.7 FSA knowledge and $\varrho = 0.8$. . . . 138

7.1 Domain properties. . . . 146
7.2 The stochastic navigation maze task - S-maze (Sutton & Barto 1998: Figure 9.5). 148
7.3 Results for the goal-based reward function with pessimistic initialisation. Each curve corresponds to a different value of λ. The top graph shows the first 10×10^2 episodes, and the bottom graph is for the whole period of learning, i.e. 25×10^3 episodes. . . . 149
7.4 Trajectories during the learning process with the goal-based reward function, pessimistic initialisation and $\lambda = 0.9$. When counting figures from the left to the right, the following numbers of learning episodes correspond to these figures: 1, 3, 5, 7 in the first row and 13, 15, 1.2×10^3, 2.5×10^4 in the second row. . . . 150
7.5 Results for the goal-based reward type with an optimistic initialisation. Each curve corresponds to a different value of λ. . . . 151
7.6 Results for the goal-based reward type with pessimistic initialisation and $\alpha = 0.01$. Each curve corresponds to a different value of λ. . . . 153
7.7 Results for the goal-based reward type with pessimistic initialisation and $\alpha = 0.9$. Each curve corresponds to a different value of λ. . . . 153
7.8 Results for the goal-based reward type with optimistic initialisation and $\alpha = 0.01$. Each curve corresponds to a different value of λ. . . . 154
7.9 Results for the goal-based reward type with optimistic initialisation and $\alpha = 0.05$. Each curve corresponds to a different value of λ. . . . 155
7.10 Results for the goal-based reward type with optimistic initialisation and $\alpha = 0.9$. Each curve corresponds to a different value of λ. . . . 155
7.11 Results for the goal-based reward type with pessimistic initialisation and $\epsilon = 0.01$. Each curve corresponds to a different value of λ. . . . 156
7.12 Results for the goal-based reward type with pessimistic initialisation and $\epsilon = 0.7$. Each curve corresponds to a different value of λ. . . . 157
7.13 Results for the goal-based reward type with optimistic initialisation and $\epsilon = 0.01$. Each curve corresponds to a different value of λ. . . . 158
7.14 Results for the goal-based reward type with optimistic initialisation and $\epsilon = 0.9$. Each curve corresponds to a different value of λ. . . . 158
7.15 Results for the step reward function, R_s, with an optimistic initialisation. Each curve corresponds to a different value of λ. . . . 160
7.16 Results for the step reward function with a semi-optimistic initialisation. Each curve corresponds to a different value of λ. . . . 162

7.17 An explored (visited) area of the state space when the step reward function is used with the semi-optimistic initialisation of the Q-table and $\gamma = 0.99$: after 10^4 iterations in the left part and after 10^6 iterations in the right part of the figure. 163
7.18 Results for the step reward with optimistic initialisation and $\alpha = 0.01$. Each curve corresponds to a different value of λ. 164
7.19 Results for the step reward with optimistic initialisation and $\alpha = 0.9$. Each curve corresponds to a different value of λ. 164
7.20 Results for the step reward with optimistic initialisation and $\epsilon = 0.01$. Each curve corresponds to a different value of λ. 165
7.21 Results for the goal-based reward with pessimistic initialisation on RW. Each curve corresponds to a different value of λ. 167

Acknowledgements

Daniel, thank you very much for being a great supervisor. I really appreciate the amount of freedom you gave me, your help, reading promptly my drafts, and constant encouragement. I had a good time in York, during which I learned a lot. It was a great pleasure for me to have the opportunity to work with you.

I thank my examiners, James Cussens and Karl Tuyls, for their feedback and interest in my work, and asking me all those difficult questions. At the end of the day, it is a pleasure to see that somebody is interested in what you have been doing for over three years.

I address my acknowledgements to our QinetiQ collaborators. Discussions with Malcolm Strens at the beginning of our project were very stimulating for me. I also thank Andrew Gardner for being supportive and patient, and always willing to help and cooperate.

I am grateful to Prof. Michael Littman for hosting me during my visit to his research group at Rutgers University, for motivating discussions, and also for pointing out that my work which I was doing towards my PhD was bound by the exploration problem in reinforcement learning in particular.

I gratefully acknowledge the financial support from QinetiQ and the UK Ministry of Defence for the scholarship which funded my studies in York and support from the Gibbs Trust for funding my research visit to Rutgers University.

I am very grateful to my colleagues from the Computer Science Department at the University of York, Pierre Andrews, Emine Gökçe Aydal, Leonardo Freitas, Teodor Ghetiu, Thomas Lampert, Bernadette Martínez Hernández, Sergio Mena, Jan Tobias Mühlberg, Silvia Quarteroni, Marcelo Romero, Sevil Şen, Frank Zeyda, Bartosz Ziółko, and many others, who made my time at the University of York pleasant, motivating, and unforgettable. To Burçu Çan, Richard Ribeiro, and Malihe Tabatabaie, I am additionally grateful for their help when I was away from York. I am also very grateful to Sam Devlin for proof reading my entire thesis. I thank Juan Perna for

being a great and patient tango teacher and for his constant willingness to boost social life at our department and beyond. Special thanks also to Enda Ridge for his sober and genuine suggestions at the very beginning of my studies in York, which helped me to understand many subtleties of research work and the process of studying towards a PhD degree. I am also grateful to my friends from beyond the university (in York and elsewhere). In particular to Robert Kamiński for our extraordinary travelling adventures together during the years of my studies in York.

I am very grateful to Prof. Ralph Huntsinger for encouraging me to apply for a PhD programme by saying that *'if you do not try, your chances are zero'*.

I thank all administrative staff, Filomena Ottaway, Pauline Greenhough, and Judith Warren in particular, for their help through my studies in York. The professional attitude of the technical support team was also very helpful. The fact that our department has its own, internal Linux distribution is impressive.

I thank my parents: my father for passing on to me his continuous courage and curiosity to understand technical issues deeply and thoroughly, and to my mother for patience and wisdom.

I am very grateful to my wife Marta. Through her love came all the understanding, support, and encouragement, which guided me along ups and downs of my PhD research.

'As one journey ends, so another begins'. With this optimistic note, I would like to thank my new project leader, Jesse Hoey, for being very supportive when I was about to submit my thesis, and for sharing my enthusiasm after I successfully defended it.

Marek Grześ

June 2010, Dundee, Scotland

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.

I hereby give consent for my thesis, if accepted, to be made available for photocopying and for inter-library loan, and for the title and summary to be made available to outside organisations.

Signed .....(candidate)

Date .....

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

1. Marek Grześ and Daniel Kudenko. PAC-MDP learning with knowledge-based admissible models. In *Proceedings of the 9th International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, 2010. ACM Press.

2. Marek Grześ and Daniel Kudenko. Online learning of shaping rewards in reinforcement learning. *Neural Networks*, *23, pages 541–550, 2010.*

3. *Marek Grześ and Daniel Kudenko. Theoretical and empirical analysis of reward shaping in reinforcement learning. In* Proceesings of International Conference on Machine Learning and Applications*, 2009. IEEE Computer Society.*

4. *Marek Grześ and Daniel Kudenko. Learning shaping rewards in model-based reinforcement learning. In* Proceedings of AAMAS 2009 Workshop on Adaptive Learning Agents (ALA 2009)*, 2009.*

5. *Marek Grześ and Daniel Kudenko. Improving optimistic exploration in model-free reinforcement learning. In* Proceedings of the International Conference on Adaptive and Natural Computing Algorithms (ICANNGA'09)*, volume 5495 of* LNCS*, 2009. Springer.*

6. *Marek Grześ and Daniel Kudenko. Reinforcement learning with reward shaping and mixed resolution function approximation.* International Journal of Agent Technologies and Systems (IJATS)*,* 1(2), pages 36–54, 2009.

7. Marek Grześ and Daniel Kudenko. Plan-based reward shaping for reinforcement learning. In *Proceedings of the 4th IEEE International Conference on Intelligent Systems (IS'08)*, pages 22–29, 2008. IEEE.

8. Marek Grześ and Daniel Kudenko. Robustness analysis of SARSA(λ): Different models of reward and initialisation. In *Proceedings of the 13th International Conference on Artificial Intelligence: Methodology, Systems, Applications (AIMSA'08)*, LNAI, 2008. Springer-Verlag.

9. Marek Grześ and Daniel Kudenko. Multigrid reinforcement learning with reward shaping. In *Proceedings of the 18th International Conference on Artificial Neural Networks (ICANN'08)*, LNCS, 2008. Springer-Verlag.

10. Marek Grześ and Daniel Kudenko. An empirical analysis of the impact of prioritised sweeping on the DynaQ's performance. In *Proceedings of the 9th International Conference on Artificial Intelligence and Soft Computing (ICAISC'08)*, LNAI, pages 1041–1051, 2008. Springer-Verlag.

11. Marek Grześ and Daniel Kudenko. Learning potential for reward shaping in reinforcement learning with tile coding. In *Proceedings of the AAMAS'08 Workshop on Adaptive and Learning Agents and Multi-Agent Systems (ALAMAS-ALAg'08)*, pages 17–23, 2008.

12. Marek Grześ and Daniel Kudenko. Plan-based reward shaping for reinforcement learning. In *Proceedings of the AAMAS'08 Workshop on Adaptive and Learning Agents and Multi-Agent Systems (ALAMAS-ALAg'08)*, pages 9–16, 2008.

CHAPTER 1

Introduction and Motivation

This chapter presents the outline of this thesis and summarises its motivation, goals, and achievements. The first section provides a high level introduction to reinforcement learning and the subsequent section introduces the topic of this thesis. Then, Section 1.3 presents arguments why this research is worth doing. The central research idea - hypothesis - is in Section 1.4 and research goals are specified in Section 1.5. The overview of the thesis and explanation on how the work was evolving is in Section 1.6. The major achievements of the work which was done towards this thesis are in Section 1.7. The structure of the rest of this thesis is discussed in Section 1.8.

1.1 Reinforcement Learning

Reinforcement learning is about learning from rewards and punishments (Sutton & Barto 1998). The learning entity, the agent, is situated in a particular environment. In each state of the environment it can execute one of the available actions and a numerical reward, which is previously unknown, is given after executing the chosen action. The goal of learning is to learn how to execute actions in order to maximise the long term reward which is given to the agent. Algorithms which tackle this type of learning problem are known as reinforcement learning algorithms. A solution to the reinforcement learning problem is a *policy* which determines which action should be executed in a given state in order to maximise the long term reward.

It is a common problem in science that theoretical models become intractable when applied to real environments of a considerable and realistic size. The same situation exists also in reinforcement learning (RL). A substantial part of RL research has focused on algorithms which require the so called Markov property to be satisfied. Without going into technical details at this level of

presentation, it is sufficient to say that the Markov property requires the state space representation in the algorithm to capture enough details so that the optimal decisions can be made when the information about only the current state is available. This allows for neat and convenient mathematical modelling, and also easiness of algorithm design and analysis, but also yields one of the most serious problems: *an exponential state space explosion*. An exponential state space explosion is caused by the fact that each state feature added to the encoding of the state space yields exponential increase in the number of states. When the state space grows exponentially in this way, the inherent problems of reinforcement learning such as the *exploration-exploitation* and the *temporal credit assignment problems* become more significant. When the state space is huge, exploration strategies which can avoid visiting irrelevant areas of the state space become essential. The temporal credit assignment problem is also more significant in larger domains because there are more possible past decisions which influence the current reward. In effect, the value function of a particular state depends on the value of a huge number of states. This naturally leads to the need for heuristics and approximate solutions.

The characteristic feature of RL is that the agent usually has significant influence on the training data which is available (e.g., in the form of samples) because the agent itself decides which actions are executed during learning and thus how it moves in the environment. This creates two distinct issues in the design of RL algorithms: (1) how to represent and approximate an optimal policy from existing data, and (2) how to select actions during learning so that the number of suboptimal actions which are executed during the training process is minimised. The second issue is named the exploration-exploitation problem.

While learning, the RL agent uses an arbitrary exploration strategy to select actions to obtain more information from the environment (i.e. samples) and improve its estimation of the policy. For practical reasons, when facing huge state spaces, practitioners are forced to use exploration which is based on the current policy with a certain degree of randomness which deviates from such a policy.

1.2 Topic of the Thesis

An important issue in RL is that the way the policy is approximated and updated influences the actual exploration which is based on the current policy. Another important factor which can positively influence exploration of the RL algorithm is domain knowledge. Domain knowledge can be used in different ways. This thesis pays significant attention to reward shaping. Reward shaping applies heuristic knowledge in order to give additional (artificial) reward to the agent in order to improve its learning. This external reward is in addition to the original reward which comes from the environment. Such a reward when based on good heuristics may already in the early stages of learning increase the value of the best actions and those actions will be chosen more often. This fact emphasises the indirect influence of the shaping reward on exploration.

In this thesis, we are looking for both novel and better ways of improving exploration, and for

better understanding of the interactions between the processes of (1) estimating the policy and (2) deciding on exploration. In particular, the focus is on the use of existing domain knowledge via reward shaping or similar ways, obtaining knowledge when it is not available, or restructuring it through reasoning and/or learning when it is not easily applicable in RL. In addition to the explicit use of knowledge to improve exploration in RL, this thesis analyses also specific RL features which govern the way the algorithm approximates the policy and which as a result influence the actual exploration.

1.3 Motivation

The primary motivation for this research is the fact that the RL community is continually seeking ways to improve RL algorithms so that they could be more useful for practical applications in domains of realistic size. The research path undertaken in this thesis is particularly worth doing because domain knowledge is a principal element which allows for reducing the space of candidate hypotheses in machine learning (Mitchell 1997; Wilkins & desJardins 2001) and these kind of advancements are necessary for transferring RL into practice. To give a good example of the significance of domain knowledge and heuristics, we refer to one of the most important tools of artificial intelligence: informed search. Informed search methods which employ heuristics to guide the search process are substantially more efficient than non-informed search techniques which are based on a systematic state expansion only (Russell & Norvig 2002). The research of this thesis is actually about making an equivalent use of heuristics in the area of RL. Additional inspiration for the work on knowledge-based RL comes from more sophisticated search techniques which use symbolic state spaces (Boutilier et al. 1999) and constitute the area of symbolic planning (Ghallab et al. 2004). The existence of domain-configurable planners in symbolic planning shows both the success of the domain-configurable concept, and provides useful patterns of how to incorporate background knowledge in learning/planning algorithms (Nau 2007). Solutions which apply temporal logic (Bacchus & Kabanza 2000; Doherty & Kvarnström 2001; Kuter & Nau 2005; Rintanen 2000), hierarchical task networks (Currie & Tate 1991; Nau 2003; Wallace 2004) or ideas of planning based on model checking (Cimatti et al. 1997) are of particular interest. Issues of knowledge representation and incorporation are of paramount importance for reinforcement learning where they have not been tackled in a systematic and methodological way.

Artificial intelligence was proposed to escape from the limitations of the mathematics in control theory in the 1950s (Russell & Norvig 2002). Calculus and matrix algebra are best applicable to systems described by sets of continuous variables and exact analysis is typically possible only for linear systems. The tools of logical inference and symbolic methods allowed dealing with a number of things, e.g., natural language, vision and planning. Symbolic methods allow also for creating different levels of abstractions and hierarchies, and for representing knowledge in principled ways, using, for example, first order logic and different types of temporal logic (e.g.,

temporal action logic Doherty et al. 1998). Reinforcement learning which is a branch of artificial intelligence is still mostly close to the mathematical/numerical methods. Hence, the basic principles which led to the creation of artificial intelligence could still lead to novel improvements in reinforcement learning.

The development of existing RL algorithms was significantly biased by the assumption that there is no domain knowledge available and that algorithms learn '*tabula rasa*' which means that existing algorithms are in many cases not prepared for the use of domain knowledge. An important question arises of how to use domain knowledge in RL. A good answer in the literature comes with the invention of reward shaping. Reward shaping applies heuristic knowledge in order to give additional (artificial) reward to the agent in order to improve its learning. This external reward is in addition to the original reward which comes from the environment (Asmuth et al. 2008; Randløv & Alstrom 1998; Ng et al. 1999). Thus, the domain knowledge can be incorporated into RL algorithms using reward shaping. The problem is, *that domain knowledge is not always in the form which can be easily expressed in a way which is directly applicable to reward shaping or another relevant way like initialisation*. Thus, the work of this thesis on methods of restructuring existing knowledge (which is not directly applicable to reward shaping) or obtaining knowledge when it is not available using reasoning and learning is desirable.

Another motivation for particular lines of the research undertaken in this thesis is that the issue of how knowledge about the domain and about properties of RL algorithms can be exploited in order to make better decisions on when and how to deploy RL algorithms.

1.4 Hypothesis

The hypothesis for the undertaken research is:

> *A detailed analysis of knowledge used in reinforcement learning as well as an analysis of domains' and algorithms' properties will create new ways in which domain knowledge is incorporated, help to improve exploration in reinforcement learning and determine when and how to deploy reinforcement learning algorithms.*

If knowledge is used properly in RL, we can expect:

- to obtain ways of improving existing RL algorithms by means of domain knowledge
- for algorithms with rigorous theoretical requirements to provide ways of using knowledge which do not violate theoretical assumptions
- to identify empirically or theoretically justified relationships between algorithm properties and its parameters against its performance on a given family of domains
- to use all the above to improve exploration, that is, to improve the learning speed and/or the quality of the final solution

The importance of and need for this line of research has already been highlighted in the RL community (Sutton & Barto 1998) and this thesis suggests the knowledge-based approach to the problem of improving exploration in RL.

1.5 Goals

The problem of exploration is an important challenge in RL and we address it in this thesis. The main objective of this thesis is to improve exploration via domain knowledge and identification of specific properties of domains and algorithms. The last objective translates into search for correlations between domain properties (knowledge about the domain) and exploration of RL algorithms.

In order to incorporate knowledge into RL, techniques which do not change the learning problem should be used or developed. A significant part of this thesis uses reward shaping as a way of incorporating domain knowledge.

In our first goal, we tackle the problem of learning with reward shaping in the situation when domain knowledge is not in the form which could be used in a straightforward way with reward shaping. Reward shaping is a relatively new technique and a number of issues have not been understood in the area, and our goal is to see how learning with reward shaping influences exploration under different conditions.

The next goal was to extend our work to the area of PAC-MDP reinforcement learning algorithms which constitute a special type of model-based algorithms. The aim was also to develop novel knowledge-based techniques to improve exploration.

The last goal was to look into the dependencies of the performance of RL algorithms under different parameters and different domain properties. It was to look for predictions on how a given configuration will behave in a given situation.

1.6 Overview of the Thesis

The first line of research in this thesis is to search for new methods of defining heuristic functions for reward shaping. Reward shaping is a method to incorporate knowledge into RL learning but it does not say anything about how to obtain this knowledge. We tackle the problem of how to find this knowledge. Thus, the first work towards this thesis looks for a methodology of reasoning about the shaping reward using symbolic planning when some high level symbolic knowledge is available. The continuation of this work tackles the same problem in the case when there is no symbolic high level knowledge. While working on these two issues, we noticed that there are specific and important properties about reward shaping which were not discussed in the literature. With a support of theoretical justification, we presented these findings in the subsequent contribution.

The research mentioned in the previous paragraph is for model-free RL. We extended our efforts to the model-based case as well. The first research in this direction was also applying

reward shaping and the outcomes were presented in (Grześ & Kudenko 2010). This work was further extended and other ways of incorporating the same knowledge were identified. They are based on building special knowledge-based MDP models and are presented in the thesis. In this case, the special type of model-based RL was used and our work was shown to be theoretically proven to meet all the requirements of this particular model-based paradigm.

In the evaluation of numerous algorithms throughout the thesis, eligibility traces were also used in several cases. Our work on reward shaping allowed us to identify novel relationships between types of the reward and the performance of eligibility traces of different length. A detailed analysis which presents our finding on this problem constitutes the last major contribution of this thesis.

1.7 Summary of Achievements

The main aim of this thesis was to tackle the exploration-exploitation problem in RL. Overall, the work done towards this thesis was successful in designing new ways of using domain knowledge to improve exploration of RL algorithms. The work on correlating the robustness of particular RL algorithms against certain classes of domains led to novel findings. The specific achievements of this thesis can be summarised as follows:

- Domain knowledge often cannot be directly translated into a heuristic function which would be easy to use in reward shaping. The first contribution of this thesis is a methodology for obtaining the shaping reward from symbolic knowledge using reasoning techniques based on symbolic planning. The results show that depending on particular details of exploration, this approach when applied to the domain with challenging exploration can either improve the quality of the final solution or considerably improve the convergence rate, which means reduce the number of suboptimal actions which are attempted.

- Further work led to the contribution which shows how to obtain heuristics for reward shaping in the case when there is no symbolic knowledge in the domain. This approach applies smaller hypothesis space for learning the heuristic function for reward shaping, and a more detailed one for learning the actual solution. The empirical evaluation shows the proof of concept that this idea can improve the convergence rate of RL, which specifically means the reduction of the number of attempted suboptimal actions.

- The next major contribution concerns the theoretical and empirical analysis of reward shaping which showed how the actual exploration of the algorithm changes when the heuristic function used for reward shaping is specified in different ways. In particular, it was shown theoretically what the actual shaping reward is when the shaping heuristic function is positive or negative. The influence of the MDP discount factor, γ, and the type of the reward function was also identified and explained theoretically.

- The next contribution tackles the same issue of incorporating knowledge into RL, but this time a particular type of model-based RL algorithms, i.e. PAC-MDP algorithms, were considered. The proposed solution uses partial knowledge about actions in the environment in a theoretically proven way. Two specific types of available knowledge are used and it is shown how to use this knowledge in a way which preserves all theoretical properties of PAC-MDP algorithms and reduces the number of states which need to be explored.
- The last contribution presents a detailed analysis of eligibility traces with regard to their influence on exploration. The type of the reward function and the initialisation of the Q-table were identified as crucial factors in the analysis, which determine the influence of eligibility traces. Additionally, it was shown how the learning rate and the exploration rate influence exploration under eligibility traces of different length.

1.8 Structure of the Thesis

The rest of the thesis is organised as follows:

- The technical introduction to reinforcement learning and related issues are contained in Chapter 2. The review of the relevant literature was presented both in Chapter 2 and also more specific discussion was added in the introduction and motivation sections of each chapter which contains the contribution of this thesis.
- After that, we start addressing the main goals of the thesis. Firstly, Chapter 3 presents the work on the use of high level symbolic knowledge - which does not lend itself easily into the shaping reward - for reward shaping. Here, the reasoning techniques based on symbolic planning are proposed to obtain heuristics which can be used with reward shaping.
- Then, Chapter 4 extends the work on obtaining heuristics for reward shaping to the case when there is no symbolic knowledge in the domain. The approach with high level learning of reward shaping is proposed.
- Our work on reward shaping revealed novel properties of reward shaping which are presented in Chapter 5 which contains empirical and theoretical analysis of reward shaping under a number of conditions.
- After that, the work on tackling the issue of incorporating knowledge into a particular type of model-based RL algorithms, i.e. PAC-MDP algorithms, is tackled in Chapter 6.
- Chapter 7 presents a comprehensive evaluation of the influence of eligibility traces on exploration.
- The thesis is concluded in Chapter 8. This chapter summarises also the achievements of the thesis and discusses future work.

CHAPTER 2

Background and Field Review

This chapter presents a technical introduction to the area of reinforcement learning and discusses work related to the research of this thesis. A more detailed discussion of related work is additionally contained in each of Chapters 3-7. This approach to the organisation of the thesis was applied in order to present smooth and detailed introduction to each contribution chapter, to motivate the undertaken research and to show the most relevant existing work in the context of a given chapter.

2.1 Reinforcement Learning

Reinforcement learning is about learning from rewards and punishments (Kaelbling et al. 1996; Sutton & Barto 1998). The learning entity, the *agent*, is situated in a particular environment. In each state of the environment, it can execute one of available actions and the numerical reward, which usually is unknown beforehand, is given after executing the chosen action. The action when executed yields not only such a numerical reward (feedback from the environment) but also affects the next state of the world (also the next state usually cannot be predicted beforehand). In consequence, a given action influences not only the current reward but also future rewards. The goal of learning is to learn how to execute actions in order to maximise the long term reward which is given to the agent. Algorithms which tackle this type of learning problem are known as reinforcement learning algorithms. A solution to the reinforcement learning problem is a *policy*, π, which determines which action should be executed in a given state in order to maximise the long term reward. In reinforcement learning, the agent is not told by the environment which action should be executed in a given situation. Instead, the algorithm has to

learn these decisions using immediate and future rewards (feedback from the environment). In comparison to reinforcement learning, in *supervised learning* the agent is informed by the environment what the decision of the algorithm should be in a particular situation (i.e. what is the target value of a specific instance of training data).

This chapter provides a short introduction to reinforcement learning. The rest of this section is structured by our view on what the term 'reinforcement learning' actually means. This issue is not trivial and it often leads to lively discussions in the RL community (see for example posts at `rl-list@googlegroups.com`) which indicate that even researchers involved in research on RL algorithms do not fully agree on the definition of the area. Similar to Whiteson (2007), we claim that the best way to explain reinforcement learning is to distinguish two separate issues: (1) *reinforcement learning problems*, and (2) *reinforcement learning algorithms*, that is, algorithms which solve reinforcement learning problems. The reason for this is that the term reinforcement learning is often wrongly associated with only a specific family of temporal-difference learning algorithms and the Q-learning algorithm in particular. Additionally, other non-temporal difference learning algorithms are applied to problems which can be classified as RL problems, but usually are not named as such. Algorithms which solve such reinforcement learning problems should be named reinforcement learning algorithms even if they are not based on temporal-difference learning. Reinforcement learning defines a problem (see Section 2.1.1) and not a particular set of methods or algorithms. Algorithms which solve these problems can be named reinforcement learning algorithms. The rest of this section introduces RL in a way which is structured to provide a detailed explanation of these issues, and to show where the research of this thesis is located. First, we start with the definition of the standard RL problem and then move to two major types of algorithms which solve these problems.

2.1.1 Standard Reinforcement Learning Problem

Without going into any details on specific algorithms, the notion of the standard reinforcement learning problem is introduced here. The agent is situated in an environment which can be in one of many states $s_t \in \mathbb{S}$. Knowing the current state, s_t, the agent decides which is the best action, a_t, for this state and executes this action in the environment. After that, the environment changes its state to a new state, s_{t+1}, and the agent is given an immediate reward, $r_t \in \mathbb{R}$. A solution to the reinforcement learning problem is a *policy*, π, which determines which action should be executed in a given state in order to maximise the long term reward. The meaning of the long term reward can be different depending whether it is better to have higher immediate reward now and lower reward later on or whether it is desirable to wait for a higher profit later on.

It is convenient when the state representation, i.e. the definition of the state space, $\mathbb{S}$, satisfies the Markov property. Formally, the Markov property means that probabilities of the next state and the next reward are conditionally independent from the entire past given the current state, s_t, and the current action, a_t. Mathematically, the Markov property is satisfied if the following

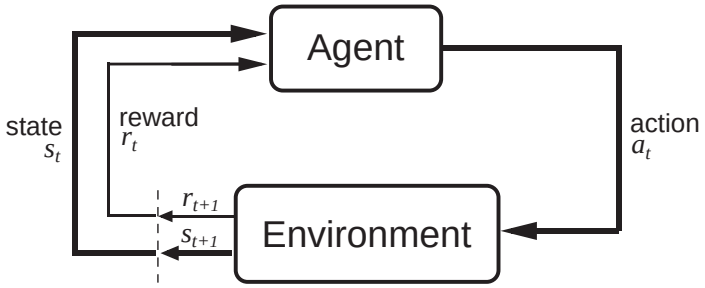


Figure 2.1: The standard reinforcement learning problem (Sutton & Barto 1998).

equation holds:

$$P(s_{t+1} = s', r_{t+1} = r | s_t, a_t, r_t, s_{t-1}, a_{t-1}, r_{t-1}, ..., r_1, s_0, a_0) \\ = P(s_{t+1} = s', r_{t+1} = r | s_t, a_t). \quad (2.1)$$

From the practical point of view, one can say that the Markov property requires the state space representation to capture enough details so that the optimal decisions can be made when the information about only the current state is available. Thus, the Markov property may not be a feature of the environment, especially in the context of reinforcement learning problems. It may rather refer to the way the state representation is defined, and the fact whether this property is satisfied or not depends on how much of the environment information is incorporated in the state representation.

When the Markov property is satisfied, the reinforcement learning problem can be solved with a policy which is a function of the current state only, i.e. $\pi : \mathbb{S} \rightarrow \mathbb{A}$. Otherwise, the policy would need to be a function of a certain number of past states or the entire history in the worst case, e.g., $\pi : \mathbb{S} \times \mathbb{S} \times ... \times \mathbb{S} \rightarrow \mathbb{A}$. When the first case is considered, the state space scales exponentially with each variable added to it, but theoretical algorithm design becomes neat and straightforward. In the latter case, the algorithm would need to deal with storing and reasoning about information which is required for making optimal decisions about a given world state. In practice, the first approach is used in most of RL research and most algorithms for fully observable environments are designed to work on the condition that the Markov property is satisfied or nearly satisfied.

Two special cases of the standard RL problem are particularly challenging and are tackled by RL research. The first case is when the transition probabilities between states and the reward function are not known beforehand. The challenge here is that the agent has to learn from experience how the environment responds to its actions (curse of modelling). The second case is when the transition probabilities and the reward function are known, but the number of states is so huge that exact methods become infeasible (curse of dimensionality), and for such cases

RL can still give results which may be far from optimal but still reasonably good (Bertsekas & Tsitsiklis 1996; van Eck & van Wezel 2008; Szita & Lörincz 2006).

2.1.2 Direct Policy Search Approach

The goal of RL is to find a policy, $\pi(s)$, which for each state, s, determines an action so that the required notion of the long term reward is maximised when chosen actions are executed. The direct policy search approach to RL uses search techniques to directly search the space of policies to find one which maximises the long term reward. When the parameter vector, θ, parameterises the policy space, the goal is to find the parameter vector, θ^*, which maximises the required criteria. Numerous search techniques were applied to direct policy search (Russell & Norvig 2002). The first direct policy search algorithm for RL problems was REINFORCE of Williams (1992). This algorithm applied optimisation based on gradient methods, and such optimisation by the gradient-following approach received considerable attention in the area afterwards (Sutton et al. 1999; Baird 1998; Ng & Jordan 2000; Kohl & Stone 2004). Other non-systematic search techniques like evolutionary algorithms (Böhm et al. 2005; Moriarty et al. 1999), neuro-evolutionary algorithms (Stanley 2004; Whiteson 2007), or cross-entropy methods (Szita & Lörincz 2006) were also successfully applied to the policy search problem. It is worth emphasising once more here that we are interested in solving the RL problem, and different search methods like gradient descent, evolutionary algorithms, etc. are used for implementing the search process in the space of policies.

The advantage of RL based on direct policy search is the fact that it can deal well and naturally with the state explosion problem. Even if the domain has a huge state space, the policy which requires far fewer parameters θ_i may be sufficient to express the optimal policy and can be found easier. Another advantage is that these methods may be more robust when the Markov property is not preserved (Schmidhuber 2001). A critical factor for direct policy search methods is the stochasticity in the domain. In stochastic domains, they require substantially more re-sampling to get accurate fitness estimates (Taylor et al. 2006).

2.1.3 Markov Decision Processes

The next approach to solving RL problems usually is modelled using Markov decision processes (MDPs). For this reason, this section introduces MDPs formally before discussing temporal-difference learning in Section 2.1.4.

An MDP is defined as a tuple $(\mathbb{S}, \mathbb{A}, T, R, \gamma)$, where $s \in \mathbb{S}$ is the state space, $a \in \mathbb{A}$ is the action space, $T(s, a, s')$ is the probability that action a when executed in state s will lead to state s', $R(s, a, s')$ is the immediate reward received when action a, taken in state s, results in a transition to state s', and $\gamma \in [0, 1]$ is the discount factor which determines how the long-term reward is calculated from immediate rewards (Puterman 1994). The problem of solving an MDP is to find a policy, $\pi(s)$, (i.e. mapping from states to actions) which maximises the

accumulated reward. A Bellman equation defines optimality conditions for the situation when the environment dynamics (i.e. transition probabilities and a reward function) are known. In such a case, the problem of finding the policy becomes a planning problem which can be solved using iterative approaches like policy and value iteration (Bertsekas 2007). These algorithms take $(\mathbb{S}, \mathbb{A}, T, R, \gamma)$ as an input and return a policy which determines which action should be taken in each state so that the long term reward is maximised. Every policy, π, has an associated state value function, $V^\pi : \mathbb{S} \to \mathbb{R}$, which represents the expected long term reward the agent will receive when it starts in state s and will follow the policy π. Another more specific form of the value function is the state-action value function, $Q^\pi : \mathbb{S} \times \mathbb{A} \to \mathbb{R}$, which specifies the long term reward which the agent is expected to receive when executing action a in state s and following policy, π, after that:

$$Q^\pi(s,a) = E\left\{\sum_{t=0}^{\infty} \gamma^t r_{t+1} | \pi, s_t = s, a_t = a\right\}. \tag{2.2}$$

The standard definition of the policy is derived from the value function:

$$\pi(s) = \arg\max_a Q(s,a), \tag{2.3}$$

and the problem of computing the policy can be reduced to the problem of computing an optimal value function, Q^*. In fact, most existing research in the area of RL has focused on methods which represent the policy via the estimation of the value function (Sutton & Barto 1998). Having defined the notion of the value function, we can introduce value iteration which applies the following update rule to find an optimal value function:

$$Q^*_{t+1}(s,a) = \max_a \sum_{s'} T(s,a,s')[R(s,a,s') + \gamma V^*_k(s')], \tag{2.4}$$

where $V^*(s) = \max_a Q^*(s,a)$.

Methods of solving MDPs in the presence of the MDP model of the environment represent another option for tackling RL problems when the state space is sufficiently small. This time, learning and policy representation are implemented indirectly via the value function.

2.1.4 Temporal Difference Learning

Temporal difference learning is closely related to methods for solving MDPs and uses the value function to represent and learn the policy. This family of algorithms has been the most often studied in the RL community to date (Sutton & Barto 1998), and the meaning of reinforcement learning is often associated and limited to temporal-difference methods (see Section 2.1 for clarification).

The policy and value iteration methods which allow solving fully specified MDPs require

access to an explicit, mathematical model of the environment, that is, transition probabilities, T, and the reward function, R, of the controlled process. When such a model is not available, there is a need for algorithms which can learn from experience. Most of the research in the area of RL studies algorithms which learn the policy from the simulation in the absence of the MDP model (Sutton & Barto 1998; Bertsekas & Tsitsiklis 1996). In many practical situations, even if an explicit, mathematical model of the MDP cannot be constructed, the system can be simulated either directly or via a generative model (it is often easier to build a generative mathematical model than an explicit model of system dynamics Tesauro 1994) and this allows the learning methods to estimate the policy.

The first approach to value function based RL is to estimate the missing MDP model of the environment using, e.g., statistical techniques. The repeated simulation is used to approximate or average the model. Once such an estimation of the model is available, standard techniques for solving MDPs, like policy and value iteration, are again applicable. This approach is known as model-based RL (Brafman & Tennenholtz 2002; Sutton 1990).

An alternative approach to RL does not estimate the model of the environment, and because of that it is called model-free RL. Algorithms of this type directly estimate the value function from repeated simulation. The standard examples of this approach constitute Q-learning and SARSA algorithms (Sutton & Barto 1998), but direct policy search methods introduced in Section 2.1.2 which do not learn the MDP model can be classified as model-free as well. Q-learning and SARSA apply temporal-difference learning to learn Q-values from which the policy is naturally derived. Temporal-difference learning was applied for the first time to the game of checkers in (Samuel 1959). These algorithms apply so called temporal-difference updates to propagate information about values of states, $V(s)$, or state-action, $Q(s,a)$, pairs. These updates are based on the difference of the two temporally different estimates of a particular state or state-action value (for this reason this paradigm is called temporal-difference learning). The particular update rules for Q-learning and SARSA are derived from the Bellman optimality equation (Bertsekas 2007) and both are presented here because we are referring to them in the other parts of the thesis. An update rule for Q-learning which is an off-policy method:

$$Q(s,a) \leftarrow Q(s,a) + \alpha[r + \gamma \max_{a'} Q(s',a') - Q(s,a)], \tag{2.5}$$

and for SARSA which is an on-policy method:

$$Q(s,a) \leftarrow Q(s,a) + \alpha[r + \gamma Q(s',a') - Q(s,a)]. \tag{2.6}$$

These methods modify the value of taking action a in state s, when after executing this action the environment returned reward r, moved to a new state s', and action a' was chosen in state s'. The Q-learning rule in Equation 2.5 is an off-policy method because the value of $\max_{a'} Q(s',a')$ does not depend on the action chosen by the exploration policy in state s', instead the maximum

value is always chosen. Whereas, SARSA uses $Q(s', a')$ in this place which means that the value of action a' chosen in state s' by the exploration policy is always used for learning. Exploration is explained in the next section, however it is worth noting here that when the exploration policy is deterministically greedy (e.g., ϵ-greedy with $\epsilon = 0$) with regard to the Q-table, Q-learning and SARSA become equivalent.

Some model-based algorithms like DynaQ have certain properties of model-free algorithms. In the case of DynaQ, temporal-difference updates are performed after real experience and additionally simulated updates are performed using the estimated model of the environment (Sutton 1990). In Rmax however, the value function is computed solely from the estimated model (Brafman & Tennenholtz 2002).

2.2 Exploration in Reinforcement Learning

Reinforcement learning is in its principle about learning from experience and exploration-exploitation is a key problem in this area. In order to learn how to behave optimally, the agent has to attempt many suboptimal actions and the successful exploration-exploitation strategy reduces the number of these actions. Our overview of existing approaches to exploration starts with the classification according to (Asmuth et al. 2009) who distinguish the following classes of exploration techniques: *belief look-ahead*, *myopic*, and *undirected*.

Belief look-ahead These kind of exploration strategies are theoretically the most 'desirable' among existing techniques. This is because they select actions in a way which not only maximises total rewards based on *the current environment state and the belief about the MDP dynamics*, but also the fact that valuable information may be gained (i.e. the uncertainty about MDP dynamics will be reduced in such a way so that the future performance will be improved) when the outcome of the chosen action becomes known. The Bayesian framework was applied to model uncertainly about MDPs in this context. The Bayesian approach was then casted as a Partially Observable Markov Decision Process (POMDP) (Duff 2002). In general, POMDPs model partially observable environments where the information about the current state is not available to the agent. Usually there are some state features observable which allow reasoning about the current state (Doucet et al. 2001). The policy is then not a function from states to actions, but from a probability distribution over possible states (*belief state*) to actions. POMDPs plan in the continuous belief space. They were applied to a situation when the uncertainty in the MDP model has to be taken into account during exploration in MDPs (the MDP state features represent the observable part of the POMDP which is used to reason about exploration and the unknown MDP dynamics constitute hidden part of the POMDP state space). An approach based on POMDPs is applied, for example, in (Poupart et al. 2006) where the BEETLE algorithm is presented. This type of exploration 'solves the exploration problem' in theory, however its prohibitive complexity limits its application to domains with few states only (Poupart et al. 2006). A recent approach to approximate Bayesian reinforcement learning proposed by Kolter & Ng (2009) represents an

interesting contribution to the idea of scaling up this type of exploration. However, the approximate nature of this algorithm makes it lose certain properties which PAC-MDP algorithms (see the next paragraph) possess.

Myopic Myopic approaches do not take into account how knowledge of the outcome of a given action (i.e. another sample of this action) would allow improving future reward. These kinds of algorithms cannot be in theory 'as optimal as' belief lookahead-approaches, however they usually have guarantees on their total regret or on the number of sub-optimal actions made during learning. PAC-MDP techniques belong to this class of algorithms (Brafman & Tennenholtz 2002; Strehl et al. 2009). Their exploration strategy always treats unknown actions as the best possible actions, and it guarantees that with high probability the algorithm performs near optimally for all but a polynomial number of time steps (i.e. polynomial in the relevant parameters of the underlying MDP).

Undirected In the last family of algorithms in this classification, actions are chosen without considering how new samples would decrease the uncertainty of the estimation of the MDP model, that is, which part of the environment model becomes known. Standard exploration strategies which belong to this category are ϵ-greedy and Boltzman. A more sophisticated example is Bayesian dynamic programming of Strens (2000). It builds a Bayesian model of possible MDPs, samples from the posterior of this model, and acts for a certain period optimally according to drawn MDPs. Asmuth et al. (2009) improved this idea by sampling a number of models from such a Bayesian posterior and then combining the obtained models into one 'best of sample set' MDP, which is likely to preserve more optimism than only one MDP as in work of Strens (2000). This allowed Asmuth et al. (2009) to prove that their approach is PAC-MDP and thus belongs to the class of myopic exploration approaches. The idea of applying optimistic model selection was used also in (Wyatt 2001), but it was not analysed theoretically in that paper.

The classification presented above basically applies to the situation when the agent is estimating the MDP model during learning (which is the case in model-based RL) or when it collects some information in addition to the Q-table for making decisions on exploration (Thrun 1992). Model-free algorithms do not learn the MDP model, therefore the above classification of exploration techniques does not apply to model-free RL unless it stores additional information on top of the Q-function. The advantage of model-free RL is that it is particularly useful when the size of the state space is prohibitive to apply model-based algorithms, because model-based algorithms need to store and learn the model. The representation of the Q-function in model-free RL using function approximation (Section 2.5 introduces the idea of function approximation in detail) or direct policy search methods (which are also model-free in most cases) allow solving even huge problems (Tesauro 1994; Crites & Barto 1996). When problems are huge, and function approximation is used, it may be prohibitive (and additionally difficult in case of continuous

state and/or action spaces) to store additional, exploration-related information in model-free algorithms or store the model in model-based RL, and for this reason 'exploration based on the current policy' is desirable and can be the only feasible method. This type of exploration stays close to the current policy (represented by the Q-table) and deviates randomly from it in order to attempt exploratory actions. The agent follows the current best policy and with some small probability tries other actions because they may turn out to be better. Our claim which is motivated by this comment is that *it is worth studying (model-free) algorithms with exploration based on the current policy*.

When considering RL algorithms with exploration based on the current policy, two types of strategies can be distinguished: (1) optimistic and (2) pessimistic. Optimistic makes an initial assumption that all actions have the highest Q-value and this value is decreased during learning. This helps to explore broadly but it may be also difficult to implement it when function approximation with global basis functions is used (see Section 2.5). Pessimistic strategies (usually) assume all actions to be equally bad initially and attempt more often those actions which yielded better outcome, and this helps the agent to focus on the best policy found up to a given time point. An important remark is that both optimistic and pessimistic approaches can be implemented with the sheer use of the Q-table, i.e. they can be based on the current policy represented by the Q-table. For example, in order to implement pessimistic exploration in the domain in which only the final goal reward is higher than 0, it is sufficient to initialise the Q-table with the value of 0. Optimistic exploration would be achieved in this situation by initialising all Q-values to a very high value (e.g., equal to the value of the goal reward). With such initialisations, exploration driven by the Q-table would be pessimistic or optimistic correspondingly.

In the above text, the importance of exploration based on the current policy was emphasised. Here, one more argument is discussed. When not only the Q-function is stored by the algorithm, but more statistics from simulation are collected (like execution counters in Delayed Q-learning Strehl et al. 2006), then with a small increase in the space requirements (space complexity of model-based RL is $\Omega(S^2A)$ and, e.g., of Delayed Q-learning $o(S^2A)$) model-based approaches could be applied, because stored counters may allow with some extension estimating environment dynamics, and model-based techniques could be used instead. Basically, if algorithms like Delayed Q-learning or other model-free which store more data instead of only the Q-table are applicable, our guess is that model-based algorithms would be applicable as well.

2.3 Eligibility Traces

The concept of eligibility traces in reinforcement learning represents one of the methods of dealing more efficiently with the temporal credit assignment problem (i.e. the problem of determining which part of the behaviour deserves the reward, which is an important issue in RL because an action executed now has its long term impact on rewards received in the future Sutton 1984) and with non-Markovian state spaces (Loch & Singh 1998; Sutton 1988; Pérez-Uribe & Sanchez

1999). In standard temporal difference learning, backpropagation is performed on only one state at a time, that is, the temporal difference:

$$\delta = r + \gamma Q(s', a') - Q(s, a) \tag{2.7}$$

is used to update only state s:

$$Q(s, a) = Q(s, a) + \alpha\delta, \tag{2.8}$$

where α is the learning rate, γ the MDP discount factor, r an immediate reward, s' is the current state, a' an action to be taken in state s', s the previous state and a the action taken in state s. The computation of the temporal difference in Equation 2.7 is according to the SARSA algorithm (Sutton & Barto 1998). The idea of eligibility traces is to propagate current temporal difference δ not only to state s but also to states which were recently visited (trace) and the measure of this recency is named eligibility. If we assume $e(s, a)$ to be the eligibility of pair (s, a), the SARSA update takes the form:

$$Q(s, a) = Q(s, a) + \alpha\delta e(s, a) \tag{2.9}$$

and is applied to all state-action pairs for each value of δ. When state s is the most recent state to be updated, eligibility for this state is set to one and after each time step it is reduced by the multiplicative factor $\lambda\gamma$ where λ controls how eligibility decays in time. SARSA with updates of this type is named SARSA(λ). The λ parameter controls how fast the eligibility of the state-action pair decays in time, and in this way it makes the impact on how far the current temporal difference is back-propagated. This implies the impact of λ on exploration.

2.4 Reward Shaping

When the agent is learning from simulation, the immediate reward r, which is in the update rule given by Equation 2.6, represents the (only) feedback from the environment. The idea of reward shaping is to provide an additional reward which will improve the performance of the agent. This improvement can mean either faster learning or a better quality of the final solution, especially in the case of large domains. The shaping reward does not come from the environment. It represents extra information which is incorporated by the designer of the system and estimated on the basis of knowledge of the problem. The concept of reward shaping can be represented by the following formula for the SARSA algorithm:

$$Q(s, a) \leftarrow Q(s, a) + \alpha[r + F(s, a, s') + \gamma Q(s', a') - Q(s, a)], \tag{2.10}$$

where $F(s, a, s')$ is the general form of the shaping reward which in our analysis is a function $F : \mathbb{S} \times \mathbb{S} \rightarrow \mathbb{R}$. A natural example of the potential function in navigation domains is the straight-line distance to the goal at the maximum speed. The shaping reward, $F(s, s')$, is then positive if,

according to such a potential function, state s' is closer to the goal than state s.

Depending on the quality of the shaping reward, it can decrease the time the algorithm spends attempting suboptimal actions, thus it can improve exploration. This decrease is the main aim of applying reward shaping. Ng et al. (1999) defined formal requirements on the shaping reward. In particular, the optimal behaviour of the (model-free) agent is left unchanged if and only if the shaping reward is defined as a difference of some potential function Φ of a source state s and a destination state s' (see Equation 2.11).

$$F(s, s') = \gamma\Phi(s') - \Phi(s) \tag{2.11}$$

This can be further clarified in the following way. When one has certain knowledge about the environment (knowledge which may help decrease the number of suboptimal actions the agent will attempt during learning), this knowledge can be used in different ways. In some cases the Q-table can be simply initialised based on this knowledge. The theoretical work of Ng et al. (1999) proved that if instead of initialising the Q-table, the same knowledge is used as a shaping reward, the final solution of the agent will not be changed. One of the most important implications of this fact is that it allows for a straightforward use of background knowledge in RL with function approximation. It is not an obvious task of how to use existing heuristics to initialise the Q-table which is represented, for example, as a multi-layer neural network (see Section 2.5 which introduces function approximation). The fact that reward shaping can be equivalent allows for a straightforward use of background knowledge in such cases. Heuristic knowledge can be easily given via reward shaping even when the function approximation with multi-layer neural networks is used. In the case of neural networks with global basis functions (Bishop 1996) the use of reward shaping instead of Q-table initialisation (assuming that such an initialisation could be done easily) would have additional advantages. The consistent reward shaping would be given all the time during the learning process, whereas initialised values would change rapidly during temporal-difference learning.

The motivation for the need for potential-based shaping comes substantially from the work of Randløv (2001) who showed a domain in which a wrongly defined reward shaping changed the objective of learning. In the domain which involves learning to ride a bicycle towards a goal which is determined by the environment reward, the agent with the shaping reward was learning to ride in cycles without moving towards the goal, i.e. it was converging to a different policy than the one specified by the environment reward. In order to act optimally according to the environment reward, the agent has to navigate directly to the goal state while avoiding falling down. This example indicated deficiencies of reward shaping and lead to the theoretically grounded work of Ng et al. (1999) and Wiewiora (2003).

The work of Ng et al. (1999) and their requirement of potential-based shaping rewards apply to model-free algorithms like Q-learning or SARSA (Sutton & Barto 1998). Recently

Asmuth et al. (2008) gave theoretically grounded conditions on the potential function Φ for a prototypical model-based algorithm R-max (Brafman & Tennenholtz 2002). In particular, Asmuth et al. (2008) proved that the R-max algorithm with potential-based reward shaping preserves its properties, i.e. is PAC-MDP (Strehl et al. 2006) if the shaping function is admissible. An admissible potential function is a function which gives a guaranteed upper bound on the value function of the optimal policy. It means that the admissible potential function is always optimistic with regard to the actual value, in the same way as admissible heuristic functions in informed search are optimistic with regard to the true distance (Russell & Norvig 2002). The shaping function is said to be admissible in the context of the R-max algorithm if $\Phi(s) \geq \max_a Q(s,a)$, that is, the shaping reward never underestimates the reward (i.e. never overestimates the cost Russell & Norvig 2002).

When potential-based reward shaping is used, heuristics are required to define the potential function, $\Phi(s)$. The reward shaping work of this thesis investigates the issue when there is no knowledge which can be easily translated into the potential function, or when there is no relevant knowledge at all. Defining $\Phi(s)$ is the principal question to answer when applying potential-based reward shaping. Defining $\Phi(s)$ in RL corresponds to the issue of defining admissible heuristics, $h(s)$, in informed search (Russell & Norvig 2002).

2.5 Function Approximation

The most straightforward approach to the representation of the value function is the state space enumeration with a separate value function entry associated with each state. There are several reasons why this approach may not be sufficient.

1. When the state space is huge, memory requirements may be prohibitive to store values for all enumerated states.

2. Neighbouring states usually have similar values of the value function. When learning with enumerated and represented individually states, only one particular state is updated during one Bellman backup. With this in mind it would be desirable if the update of the value function of one state could influence also values of neighbouring states.

3. Some global regularity in the feature space of the state representation may allow for broad generalisations in the representation of the value function (e.g., using multi-layer perceptron or more generally regression methods which use global basis functions Bishop 1996).

Value function approximation methods take advantage of the fact that states with similar values of state features have in most cases a similar value of the value function, or that the global generalisation can be achieved. The idea is to represent the value function, $V(s)$, as a vector of parameters, $\theta \in \mathbb{R}^d$, with d smaller than the number of states. In this way, the update of the value function according to one state is generalised across similar states (Sutton 1996). The general

form of this approach to the SARSA algorithm yields the following update rule:

$$\theta' = \theta + \alpha\delta_t(Q_\theta)\nabla_\theta Q_\theta(s,a). \tag{2.12}$$

When linear function approximation is used, that is, when $Q_\theta = \theta^\top\phi$ where $\phi : \mathbb{S} \times \mathbb{A} \to \mathbb{R}^d$ defines basis functions then $\nabla_\theta Q_\theta(s,a) = \phi(s,a)$. The special case of this type of function approximation is used in our work of Chapter 4. Linear function approximation is commonly used in practice, however little is known about its convergence properties. The only known theoretical results are due to Melo et al. (2008) who prove convergence under rather restrictive conditions (Szepesvari 2009).

An interesting issue is how different regression methods address requirements listed at the beginning of this section. For example, the second issue can be addressed with function approximation based on local basis functions (e.g., radial basis functions Bishop 1996) or linear averagers (Gordon 1995; Szepesvari 2009). Basis functions of this type are robust in preserving initialisation of the approximation and are also required by specific techniques which have tight requirements on used approximation. For example, proofs of convergence of fitted value iteration in (Gordon 1995) require functions which are contraction mappings and linear averagers meet this requirement. The problem with these methods however is the fact that they do not address well the issue of the exponential state space explosion which is due to the Markov property. Approximation with global basis functions (which addresses the third issue from the list at the beginning of this section) is much more robust against the state space explosion due to global generalisation. The learning process is, however, more problematic in this case. The update of one state usually changes the value function of the whole state space (e.g., when linear regression or the multi-layer perceptron is used). This leads to problems with initialisation and exploration, because the current policy may be changing radically during learning. For this reason, methods like neural fitted Q iteration need to store $\langle s, a, s' \rangle$ triples and re-use them during training of the neural network (Riedmiller 2005), i.e. a type of experience replay is applied (Lin 1992).

2.6 Symbolic Planning

Reinforcement learning problems are inherently about learning how to make a sequence of decisions. The same nature of the task appears also in the most fundamental area of artificial intelligence which is heuristic search. Standard search algorithms do not look into the structure of the state space and states are enumerated and treated separately (Russell & Norvig 2002). In many cases however, structural relations, which can be described using first-order logic, can be identified. Such an approach to heuristic search is applied in *symbolic planning* which can be seen as heuristic search with a structural representation (Ghallab et al. 2004). An intensional view of the state space allows dealing with much larger state spaces and allows for additional directions in developing algorithms (e.g., Graphplan is such an algorithm which builds specifically

on the symbolic representation of the state space Blum & Furst 1997; Blum & Langford 1998; also forward search planners use planning graphs to extract heuristics Hoffmann & Nebel 2001).

The existence, development, and growing successes of symbolic planning (Nau 2007) are particularly motivating for RL. Firstly, symbolic planning is inherently knowledge-based (even domain-independent symbolic planners require much more knowledge than standard RL algorithms). It creates ways of incorporating domain knowledge into symbolic planning. Thus, ideas from symbolic planning can and should be transferred to RL which tackles a more difficult problem that requires learning due to incomplete specification of the environment. Secondly, existing symbolic planning techniques operate mostly on the condition that behaviour of the system can be fully specified beforehand. RL can help with this regard via its ability to learn from experience.

While wrapping up this section, we emphasise that inspiration for new and more robust RL methods should be sought in the area of symbolic planning as well. We take this direction in Chapters 3 and 6.

2.7 Domain Knowledge in Reinforcement Learning

When reinforcement learning is defined in the literature, it is sometimes stressed that it can learn without any (or very little) prior knowledge about the environment, and that the agent may successfully learn only from reward and punishment during its interaction with the environment. This may be a desirable property of this paradigm, especially in the context of scientific modelling of dynamic processes (i.e. when the goal is to understand the process), or learning in the area of theoretical neuroscience (Dayan & Niv 2008; Dayan & Abbott 2001). In technological applications of RL however, the goal is substantially to obtain the best possible approximation of the optimal policy after possibly shortest period of training. The requirement is that the policy performs well, and details of how it achieves it may stay hidden. This thesis highlights the fact that domain knowledge is important for speeding up RL algorithms in particular in technological (though not only) applications. We start with the definition of three coarse classes of knowledge in RL:

1. The first type of knowledge is to define the state representation, 'knowledge about the state representation'. It is required to define the state space and representation of the policy in the RL framework. Here, issues of the design of the state and the value function representations such as function approximation are considered (e.g., the choice of basis functions in neural methods, i.e. tile coding Lin & Kim 1991, radial basis functions, multi-layer perceptron, etc. Bishop 1996, or the choice of a flat or hierarchical state representation, e.g., MAXQ Dietterich 2000, HAMQ-learning Parr & Russell 1997, etc.). Namely, it is about the definition of two things: the space of possible policies, $\pi \in \Pi$, and the properties of the chosen space so that the search can be implemented efficiently, and/or the update of the value function can be possible with a given search method. Certainly, conceptual

knowledge from (3) is required to make decisions at this stage.

2. The second type of knowledge includes 'procedural knowledge' which basically helps to guide the exploration process (admissible heuristics in informed search can be seen as an example of such knowledge). This could also include labelled pairs of states and optimal actions for those states. Such labelled pairs normally exist in supervised learning (Mitchell 1997).

3. The broad spectrum of existing RL algorithms contains algorithms which work well in specific situations but fail in others. An important type of knowledge - 'conceptual knowledge' - recognises correlations between properties of algorithms - e.g., their parameters - and domains.

It was explicitly argued by Wilkins & desJardins (2001), that knowledge is necessary for further development of planning techniques. The same claim can be transferred to RL. In our opinion, the success of the knowledge-based approach requires certain considerations, and we basically suggest that knowledge should be used and analysed in a systematic way in this area. The first reason is that most of the published work in the area does not clearly say which kind of knowledge is available to the algorithm, when specific empirical results are presented. Improvements to specific algorithms when obtained via the use of knowledge should be clearly indicated because the same knowledge may be used in more effective ways. It may encourage other researchers to look for such methods. Furthermore, the explicit consideration of knowledge available to the algorithm usually naturally defines the boundary of applicability of a given approach. Also the use of knowledge types distinguished above in 1-3 requires understanding of implications of given knowledge on decisions about other aspects of the system. For example, reward shaping is substantially more helpful in model-free RL, but it may be profitable to look for different knowledge to be used more efficiently in other ways as shown in Chapter 6 of this thesis.

Knowledge is certainly necessary also for successful deployment of RL algorithms in realistic domains. In this thesis, we look in particular at ways of acquiring the procedural knowledge (in the form of the potential function for reward shaping, or admissible MDP models for PAC-MDP learning), and also attempt to deepen the understanding of correlations between properties of algorithms and domains.

2.8 Motivating Remarks

Before this chapter ends and the description of particular contributions starts, several questions are placed together which the inquisitive reader may want to revisit once again.

2.8.1 Why does reward shaping influence exploration?

When the RL agent follows either the greedy policy (e.g., Rmax) or the one which diverts with a certain small probability from the greedy policy (e.g., ϵ-greedy), the content of the Q-table guides the exploration. The environment reward, even though it precisely specifies the objective behaviour which is to be learned, may not be very informative during learning and temporal-difference updates do not yield any differentiation in terms of the value of actions of a given state in the early stages of learning. The shaping reward is exactly to provide this faster indication. When a given action is good according to the potential function, its execution will be rewarded by the shaping reward and it will be preferred next time against other actions which do not get this incentive from the shaping reward. Good actions are basically rewarded earlier. This usually happens much earlier than the environment reward gets back-propagated far enough to give the same indication.

2.8.2 Why do eligibility traces influence exploration?

As in the previous case, we consider here the exploration policy which is based on the current policy. The length of eligibility traces influences the propagation of the value function, and it therefore influences the current policy.

2.8.3 What is reward shaping for?

The purpose of reward shaping is basically to solve the problem more easily and faster. First of all, its role is to improve the learning rate, especially in early stages of learning, by giving guidance on which actions make transitions to better states by explicitly rewarding those actions. The second purpose concerns domains of significant size for which exact or near optimal solutions cannot be found. Reward shaping may help in such domains to obtain better approximation of the optimal policy. Here, its role is the same as in the corresponding case of informed heuristic search. Search problems which are sufficiently large cannot be solved without admissible heuristics. Additionally, when the search domain is huge, even the best admissible heuristics fail in leading best-first search to the solution in realistic time. When search with weighted heuristic is used, the solution may not be optimal but one can still be found significantly faster (Pohl 1970; Thayer & Ruml 2008). The correspondence between informed heuristic search and reward shaping in PAC-MDP algorithms is even closer than in model-free RL. They both rely on admissible heuristics, and they suffer the same problems when heuristics are not admissible. In model-free reward shaping, there is no need for the potential function to be admissible but it ideally should be close to the value function which is sought (Ng et al. 1999).

2.8.4 Can we not do something easier instead of reward shaping?

When the only available knowledge is the potential function, $\Phi(s)$, reward shaping makes the best use of this knowledge. The potential function defines only how good each state is. There is

no knowledge of how to move between states, i.e. what are the MDP transition probabilities in the controlled process. The power of reward shaping is that it does not require any information about action dynamics. When a specific sample is obtained from the simulation, the shaping reward can be easily computed since both the current and the next state are given in such a sample. So, the shaping reward is applied jointly with samples, and in this way it can cope with the problem of the lack of transition probabilities.

An alternative way of using the potential function in RL is to initialise the value function. This is however not easy when function approximation is used as discussed in Section 2.4 and this also justifies why it is important to study reward shaping.

In model-based RL, there are some more opportunities to apply alternative methods of incorporating domain knowledge because the algorithm is doing continuous replanning (e.g., Rmax and other PAC-MDP algorithms do it) when a new state becomes known. We explore this direction in our work in Chapter 6.

2.8.5 Why is exploration according to the current policy necessary?

Exploration which is based on the current policy is either greedy or diverts with some probability from the policy determined by the Q-table. This kind of exploration generally aims at being relatively close to the best known solution. This approach naturally fits the paradigm of self improving on-line learning, when the agent acts in the environment and constantly improves its performance. Actually, *reward shaping which was discussed above makes most sense with this type of exploration. When another external decision process dictates how explorative actions are chosen, reward shaping becomes irrelevant.* It rapidly rewards promising actions which are then attempted more often. The entire theoretical analysis which is behind PAC-MDP algorithms is based on this type of exploration which is about following the current policy greedily in this case (Strehl et al. 2009).

CHAPTER 3

Plan-based Reward Shaping

This chapter focuses on the use of domain knowledge to improve the convergence speed and optimality of RL techniques. Specifically, we propose the use of high-level STRIPS operator knowledge in reward shaping to focus the search for the optimal policy with two main prototypical exploration strategies: pessimistic and optimistic. Empirical results show that in certain situations the plan-based reward shaping approach outperforms other RL techniques, including alternative manual and MDP-based reward shaping. We show that MDP-based reward shaping may fail and successful experiments with STRIPS-based shaping suggest modifications which can overcome encountered problems. The STRIPS-based method we propose allows expressing the same domain knowledge in a different way and the domain expert can choose whether to define an MDP or the STRIPS planning task. We also evaluate the robustness of the proposed STRIPS-based technique to errors in the plan knowledge.

3.1 Introduction

Potential-based reward shaping represents an elegant and theoretically correct way of incorporating domain knowledge into RL algorithms (see Section 2.4 for the introduction). One problem with reward shaping is that often detailed knowledge of the potential function, $\Phi(s)$ in Equation 2.11, is not available or is very difficult to represent directly in the form of a shaped reward. Rather, some high level knowledge of the problem domain exists, that does not lend itself easily to explicit reward shaping.

Section 2.6 introduced the concept of symbolic planning without going into technical details on how planning problems can be specified. STRIPS is a formal language which is used for

specifying these kind of automated planning instances. The description of the planning problem in STRIPS includes the behaviour of the system (operators), and start and goal states (Fikes & Nilsson 1971). Actions have a set of preconditions which have to be satisfied for the action to be executable, and the set of effects which are made true or false by the action. The start state contains a set of conditions which are initially true, and the goal state consists of conditions which have to be true or false for the state to be classified as a goal state.

In this chapter, we focus on the use of high-level STRIPS operators to automatically create a potential-based reward function that improves the ability and speed of the agent to converge towards the optimal policy. The only interface between the basic RL algorithm and the planner is the shaping reward and information about the current state. In related works, where planning operators were also used (Grounds & Kudenko 2005; Ryan 2002), a RL agent learns an explicit policy for these operators. Thus, STRIPS knowledge is at the same level of abstraction as the target RL process. In our approach, symbolic planning is at the higher abstract level and provides additional knowledge to a classical RL agent in a principled way through reward shaping. As a result, our approach does not require frequent re-planning as is for example the case in (Grounds & Kudenko 2005).

We evaluate the proposed method in a flag-collection domain, where there is a goal state (necessary for applying STRIPS) and a number of locally optimal ways to reach the goal. Specifically, we demonstrate the success of our method by comparing it to RL without any reward shaping, RL with manual reward shaping, and an alternative technique for automatic reward shaping based on abstract MDPs (Marthi 2007). Thus, the contribution of this chapter is the following:

1. We propose and evaluate a novel method to use the STRIPS-based planning as an alternative to MDP-based planning for reward shaping
2. We show that MDP-based reward shaping may fail in certain situations and successful experiments with STRIPS-based shaping suggest modifications which can overcome encountered problems
3. The experimental section yields an additional contribution by showing how different reward shaping approaches influence different exploration schemas, i.e. pessimistic and optimistic exploration

The STRIPS-based method we propose allows expressing the same domain knowledge in a different way from MDPs and the domain expert can choose whether to define an MDP or the STRIPS planning task. The STRIPS-based approach brings new merits to reward shaping from abstract/high level planning in domains with the intensional representation at an abstract level (Boutilier et al. 1999) where such representations usually allow for symbolic reasoning.

High-level domain knowledge is often of a heuristic nature and may contain errors. We address this issue by evaluating the robustness of plan-based reward shaping when faced with incorrect high-level knowledge or plans.

One of the principal solutions of dealing with huge state spaces in RL is function approximation. Because reward shaping does not change RL algorithms and only the reward function is modified, the approach proposed in this paper can be easily applied with any kind of function approximation. Related literature (Meuleau et al. 1999) considers additionally model-free RL as a possible way to overcome the curse of dimensionality of the state space, in general, because function approximation can be applied with this type of RL in the most straightforward way (Stone & Sutton 2001; Stone et al. 2005). This was another reason why our empirical evaluation of plan-based reward shaping in this chapter was performed on the model-free algorithm. The abstract level knowledge can span over large state spaces and this makes it particularly useful for learning policies in such huge state spaces which can be effectively represented and learned with model-free algorithms. Model-based algorithms which have much higher representational and computational requirements are investigated in Chapter 6. More discussion on the issue of model-free versus model-based RL with regard to the results of this chapter is in Section 3.7.

3.2 Plan-based Reward Shaping

The class of RL problems is considered in which background knowledge allows defining state and temporal abstractions using the intensional representation of the environment (Boutilier et al. 1999). Abstract states are defined in terms of propositions and first order predicates, and temporally extended actions, also known as *options* (Sutton et al. 1999), can be treated as primitive actions at the abstract level. The function $f_{abs} : \mathbb{S} \mapsto \mathbb{Z}$ maps states $s \in \mathbb{S}$ onto their corresponding abstract states $z \in \mathbb{Z}$.

3.2.1 The Potential Function from the STRIPS Plan

The intensional representation allows for symbolic reasoning at an abstract level when options can be defined in terms of changes to the symbolic representation of the state space, e.g., they can be expressed as STRIPS operators. STRIPS planning can be used to solve the task at the high level by means of symbolic reasoning. When the RL problem is to learn a policy which moves the agent from the start state, s_0, to the goal state, s_g, it can be translated to the high level problem of moving from state $z_0 = f_{abs}(s_0)$ to state $z_g = f_{abs}(s_g)$. Because of the intensional representation at the abstract level, symbolic reasoning can be used to solve the planning problem of moving from state z_0 to goal state z_g. This is a classical planning task which can be solved using standard STRIPS planners. Graphplan of Blum & Furst (1997) is used in our implementation. The trajectory $\xi = (z_0, z_1, ..., z_g)$ of abstract states (obtained from plan execution at the abstract level) can be used to define the potential function for low level states as:

$$\Phi(s) = step(f_{abs}(s))w,$$

where the function $step(z)$ returns the time step on the abstract level when the given abstract state, z, appears during plan execution and w is an optional scaling factor[1]. In other words, the potential function is incremented after the RL agent has successfully completed an abstract action in the plan, and reached a (low-level) state that is subsumed by the corresponding abstract state in the trajectory.

The question remains how to define the potential function to be assigned to those abstract states that do not occur in the plan. One option is to ignore such states and assign a default value of zero. This approach can strongly bias the agent to follow the path determined by the plan. The agent would be strongly discouraged from moving away from the plan. As discussed later, this leads to problems when the plan is wrong and in particular when there is no transition from state z_i to state z_{i+1} in the environment. The agent may not be able to get out of state z_i, because of the negative reward for going to any state other than z_{i+1}. This and other types of knowledge incorrectness are discussed and evaluated in detail in Sections 3.6.1.2 and 3.6.2.2.

We propose a more flexible approach that allows the agent to abandon the plan and look for a better solution when the plan is wrong. Algorithm 1 explains the idea. States which are in trajectory ξ (plan states) have their potential function set to the scaled time step of their occurrence in the plan. Non-plan states that are reachable from any state $z \in \xi$ have their potential function set to the potential function of the last visited plan state (variable $last$). In this way the agent is not discouraged from diverging from the plan. It is also not rewarded for doing so. A problem with

Algorithm 1 Assigning the potential function, $\Phi(s)$, to low level states through corresponding abstract states.

Input: state s, boolean $new\_episode$
Output: the potential function, $\Phi(s)$, of state s

if $new\_episode$ **is true then**
  $last \leftarrow 0$
end if

if $f_{abs}(s) \in \xi$ **then**
  $last = step(f_{abs}(s))w$
else
  if $last > array\_max[f_{abs}(s)]$ **then**
    $array\_max[f_{abs}(s)] = last$
  else
    $last = array\_max[f_{abs}(s)]$
  end if
end if

return $last$

this approach is that some non-plan states can be reached from different levels of the potential function. For this reason, for each non-plan state the highest value of the last potential function

[1]See Chapter 5 for a detailed analysis of reward shaping which investigates also the issue of scaling the potential function and its relation to the environment reward.

is stored in $array\_max$. The main aim of using this array is to prevent continuous changes in the potential function of non-plan states which may be disadvantageous for the convergence of the main RL process which will use this potential function for reward shaping.

3.2.2 Manual Definition of the Potential Function

The abstract goal state in the considered class of RL tasks needs to be defined as a conjunction of propositions. The most straightforward way to define the potential function for such goals manually is to raise it for each goal proposition which appears in a given state. This kind of potential function, even though it gives some hints to the agent about propositions which bring it closer to the goal, does not take into account how the environment is regulated. There may exist a certain sequence of achieving goal conditions that leads to higher rewards. One example is the travelling salesman problem. The value of the potential function raised just for each visited town will strongly bias the nearest neighbour strategy. An admissible heuristic based on, e.g., minimum spanning trees can be used to obtain a correct admissible heuristic in this problem (Russell & Norvig 2002). In our approach, instead of encouraging the agent to obtain just goal propositions, a more informed solution is proposed that takes into account how the environment behaves as it can be expressed in the STRIPS notation.

3.2.3 The Potential Function from Abstract MDP

Marthi (2007) proposed a general framework to learn the potential function by solving an abstract MDP. In this section we show how this idea can be applied with the same kind of knowledge that is given to the STRIPS-based approach. The shaping algorithm of Marthi (2007) obtains the potential function by firstly learning dynamics for options (i.e. actions at the abstract level) and secondly solving an abstract MDP. Options can be defined as policies of low level actions. Because in our class of problems options are assumed to be primitive deterministic actions at an abstract level, computation of their dynamics can be omitted. An abstract MDP can be solved using, e.g., value iteration, before the main RL learning process begins and the obtained value function is used directly as the potential function. The following equation describes this idea:

$$\Phi(s) = V(f_{abs}(s))w, \tag{3.1}$$

where $V(z)$ is the value function over state space $\mathbb{Z}$ and it represents a solution to the corresponding MDP-based planning problem, and w is an optional scaling factor. Because the high level model is deterministic and options make transitions between abstract states, this planning task can be solved using the following formula which represents a special case of value iteration:

$$V_{k+1}(z) = \max_{z'}[R_{zz'} + \gamma V_k(z')]. \tag{3.2}$$

Knowledge equivalent to STRIPS operators can be used to determine the possible next states z' for a given state, z. The reward given upon entering the goal state in the abstract MDP and the discount factor, γ, can be easily tuned to make the difference in the value function between neighbouring abstract states close to one, as it is the case in the plan-based algorithm.

3.3 Experimental Domain

The proposed algorithms are evaluated on an extended version of the navigation maze problem. This problem has been used in many RL investigations, and is representative of RL problems with the following properties:

- There exists an abstract goal state, which can stand for a number of actual goal states. A well-defined goal state is necessary for applying STRIPS planning.

- There are many ways to reach the goal with varying associated rewards. In other words, there are local policy optima that the RL agent can get stuck in.

We use a maze domain to evaluate our algorithm in a way which allows understanding the algorithm's behaviour and comparison with related approaches. Therefore, it will be suitable for any real-world problem with the above properties. Such a domain also can be seen as a proxy of the real world (Pasula et al. 2007). This approach to empirical AI evaluations is commonly adopted in the related literature, e.g., (Ryan 2002).

In the standard navigation maze problem, an agent moves in a maze and has to learn how to navigate to a given goal position. In the extended version of this problem domain, the agent additionally has to collect flags (i.e. visit certain points in the maze) and bring them to the goal position. The reward at the goal is proportional to the number of flags collected. In order to introduce abstraction and demonstrate the use of high-level planning, the maze is additionally partitioned into areas (rooms).

Because an episode ends when the agent reaches the goal position regardless of the number of collected flags, this problem is challenging in terms of exploration and has been used extensively in the existing literature, e.g., (Dearden et al. 1998; Strens 2000). The learning agent can easily get stuck in a local optimum, bringing only a reduced number of flags to the goal position.

An example maze is shown in Figure 3.1. The agent starts in state S and has to reach goal position G after collecting as many flags (labelled A, B, C, D, E, F) as possible. The episode ends when the goal position has been reached and the reward proportional to the number of collected flags is given. Thus, the reward is zero in all states except the goal state in the default configuration which is evaluated with pessimistic exploration. In experiments with optimistic exploration, each action has additionally the cost (negative reward) of -1. The agent can choose from eight movement actions which deterministically lead to one of eight adjacent cells when there are no walls. The move action has no effect when the target cell is separated by a wall.

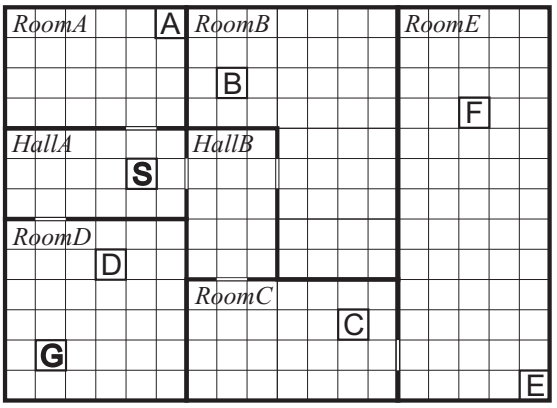


Figure 3.1: The map of the maze problem. S is the start position and G the goal position. Capital letters represent flags which can be collected.

3.4 Evaluated Algorithms and Parameters

In the empirical evaluation, the SARSA algorithm was used. The usage of SARSA aims at investigating the influence of potential-based reward shaping on model-free reinforcement learning, in general. This algorithm was used in its basic form as it is presented in (Sutton & Barto 1998). The following values of parameters were used: $\alpha = 0.1$, $\gamma = 0.99$ with step reward equal to 0 and $\gamma = 1$ with negative reward. The number of episodes per experiment was 10^4 with the first and 10^5 with the second reward type. The second value was higher because exhaustive exploration with the corresponding reward type required more episodes of learning. In all experiments an ϵ-greedy exploration strategy was used where epsilon was decreased linearly from 0.3 in the first episode to 0.01 in the last episode. To achieve pessimistic exploration with the step reward equal to 0, the Q-table is initialised with the value of 0 for all state action pairs. In the optimistic case with the negative step reward, all state-action pairs start with the Q-value equal to 600 (which corresponds to the highest reward from the environment) for all but goal states and 0 for goal states.

Plan-based reward shaping was compared with a non-shaping approach and with three other shaping solutions. This leads to five reward shaping options:

- no reward shaping,
- plan-based (STRIPS-based) reward shaping,
- abstract MDP-based reward shaping,
- flag-based reward shaping,
- plan-flag reward shaping which combines plan-based and flag-based approaches.

STRIPS-based and abstract MDP-based reward shaping are used in the form as they were introduced in Section 3.2. In the above no-shaping case, no shaping reward is given. The flag-based shaping reward is determined by the number of collected flags, and the potential function is the function $\Phi(s) = flags(s)w$, where $flags(s)$ is the number of collected flags in state s (it is worth remembering that the state of the environment is the position of the agent and names of collected flags). It is an instance of the manual shaping approach (discussed in Section 3.2.2) which raises the potential function for each goal proposition achieved in the current state. This kind of reward shaping thus represents the 'nearest flag' heuristic. In plan-flag reward shaping, the potential function is:

$$\Phi(s) = (plan(s) + flags(s))w$$

which means that it is a sum of the STRIPS-based potential function, $plan(s)$, and the number of collected flags in state s, $flags(s)$. Flag-based reward shaping when combined in this way with STRIPS-based shaping may hurt the performance of the pure STRIPS-based approach because the 'nearest flag' heuristic is suboptimal. However the 'nearest flag' bias can help in the case of incorrect planning knowledge. For this reason such a composition of flag- and STRIPS-based shaping named plan-flag is also evaluated.

If not explicitly mentioned otherwise, all experiments were repeated ten times and the average performance is shown in graphs of this chapter. Because of considerable amount of episodes in the presented results, results are also averaged along the x-axis with a window of 100 episodes. This guarantees enough smoothing which makes graphs easier to present and analyse. Presented error bars for standard error of the mean (SEM) are for this reason in some cases displaced from the smoothed curve because they are computed without horizontal averaging. They were added for a more informative presentation of results (Cohen 1995).

3.5 Potential Function for Experimental Domain

This section explains how the evaluated reward shaping approaches were applied to the flag collection domain.

3.5.1 Low Level Model

In our experiments, reinforcement learning is carried out at the low level which is defined by the target MDP ($\mathbb{S}$, $\mathbb{A}$, R, T), where $\mathbb{S}$ is the state space defined by the position of the agent in the 13×18 maze and by the collected flags, and $\mathbb{A}$ is the set of eight primitive actions corresponding to eight movement directions. The reward function R and transition probabilities T are not known to the agent in advance.

3.5.2 High Level Representation

Plan-based reward shaping assumes that there exists a high level structure in the modelled world. The access to two types of knowledge about the abstract level world is required:

1. **State mapping:** The mapping from low level to abstract states in the form of a function $f_{abs}(s) : \mathbb{S} \mapsto \mathbb{Z}$. This function identifies each abstract state as the area in the maze in which the given low level position is located. Hence, the abstract state is determined by the room location of the agent and the set of collected flags. Such a state can be symbolically expressed as:

$$robot\_in(roomB) \wedge taken(flagE) \wedge taken(flagF). \tag{3.3}$$

2. **Transitions:** Possible transitions between high level states. In this case, there are three types of knowledge which allow defining transitions at the abstract level:

   (a) Possible transitions between areas in the maze (i.e. which adjacent rooms have doors between them). This knowledge can be expressed by the following rigid facts:

```
(next-to roomC roomE)
(next-to roomE roomC)
(next-to roomA hallA)
(next-to roomB hallB)
(next-to roomC hallB)
(next-to roomD hallA)
(next-to hallA roomA)
(next-to hallB roomB)
(next-to hallB roomC)
(next-to hallA roomD)
(next-to hallA hallB)
(next-to hallB hallA)
```

   (b) Location of flags: in which room a given flag is located. The initial configuration of fluents which describe the position of flags is as follows:

```
(flag-in flagA roomA)
(flag-in flagB roomB)
(flag-in flagC roomC)
(flag-in flagD roomD)
(flag-in flagE roomE)
(flag-in flagF roomE)
```

   There is one more fluent which determines the position of the robot in the maze:

```
(robot-in hallA)
```

(c) STRIPS operators. In our domain, the following STRIPS operators were used:

```
(TAKE ((<flag> FLAG) (<area> AREA))
    (preconds
        (flag-in <flag> <area>) (robot-in <area>))
    (effects
        (del flag-in <flag> <area>) (taken <flag>)))
(MOVE ((<from> AREA) (<to> AREA))
    (preconds
        (robot-in <from>) (next-to <from> <to>))
    (effects
        (del robot-in <from>) (robot-in <to>)))
```

3.5.3 High Level Planning Problems

Knowledge about the high level structure of the world is used to define high level planning problems. The abstract state representation attributes are used to define state representation for both classical and MDP-based planners. In the case of the STRIPS representation, the location of the robot and symbolic names of collected flags are used for an intensional description of the world. For the MDP-based planner the state space is enumerated and all possible abstract states are collected in the tabular representation which has 448 entries. In both cases the state encoding preserves the Markov property.

Both investigated planning approaches require action models. In this case knowledge about transitions and intensional state representation is used to define high level actions. The first-order STRIPS operators presented in the previous sub-section are grounded and used as actions at the high level. These operators together with knowledge about the possible transitions between areas and the location of flags allow reasoning about the changes in the environment. The same knowledge is used to define possible transitions between abstract states in the abstract MDP, specifically to find for each state the set of reachable states. According to the description of the algorithm, deterministic options are assumed which allow for deterministic transitions between abstract states.

Introduced STRIPS actions allow reasoning symbolically about the changes in the world. The planner has to find a sequence of `MOVE` and `TAKE` actions which can transform the system from the start state in which $robot\_in(hallA)$ to the goal state:

$$robot\_in(roomD) \wedge taken(flagA) \wedge taken(flagB) \wedge taken(flagC) \\ \wedge taken(flagD) \wedge taken(flagE) \wedge taken(flagF). \quad (3.4)$$

Because of the closed-world assumption (everything not mentioned explicitly in the description of the state is assumed to be false) the start state has to define all initial facts, like locations of flags (e.g., $flag\text{-}in(flagA, roomA)$) and connections between rooms (e.g., $next\text{-}to(hallB, roomC)$). Facts from the last group are called rigid facts because they do not change over time (the fact whether rooms are connected or not remains unchanged).

Both the MDP and STRIPS planning problems can be solved in advance before the learning takes place. Once these problems have been solved they can be used to assign the potential function to high level states directly and to low level RL states indirectly via the mapping function, which translates low level states to high level abstract states. The potential function is assigned to abstract states in the manner presented in Section 3.2.

The abstract MDP planning was performed with $\gamma = 0.98$ and the goal reward of 100. These values were chosen to obtain difference of the potential function between neighbouring abstract states close to 1. The scaling factor w was selected as follows: In plan-based and MDP-based reward shaping, $w = 600/18$ because the optimal plan length is 18 and the maximal value function in 600. In experiments with pessimistic exploration, the value of w was much less important and results were the same within a large range of values of w. In experiments with optimistic exploration, $w = 1$ was leading to only small improvement of reward shaping, for this reason higher values were tested and selected. In the flag-based case, $w = 600/6$ because there are 6 flags in the domain, and in the plan-flag version, $w = 600/(6 + 18)$ as it combines plan and flag based potentials. The reader is referred to Chapter 5 which contains a detailed analysis of the issue of scaling the potential function in reward shaping and its influence on exploration in model-free RL.

3.6 Empirical Results

In this section, the empirical results are presented and discussed. The presentation of results is divided according to two dimensions of the analysis. The first dimension is determined by the type of exploration which is considered, i.e. pessimistic versus optimistic. Additionally the correctness of knowledge is considered and its impact on the tested algorithms is evaluated. The knowledge accuracy factor is important because even though the high level plans used for reward shaping are optimal according to the provided high level knowledge, this knowledge may contain errors. Therefore, the plan may not be optimal at the low level where the RL agent operates. For this reason the presentation of experimental results is divided in each of the two following sections. In each section, the results are firstly analysed when the high level plan is optimal. Afterwards, various possible plan deficiencies are defined and their impact is empirically evaluated.

```
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)
```

Figure 3.2: The optimal STRIPS plan.

3.6.1 Pessimistic Exploration

The intuition for applying planning knowledge as the potential function was that it can influence this type of exploration dramatically since the agent is guided towards highest reward goal states in an informed way along long trajectories which can be obtained by the planner at the abstract level. The knowledge-based guidance associated with the greedy (e.g., in the ϵ-greedy sense) pessimistic exploration can guide the agent rapidly to the reasonable solution and in the case of difficult exploration, as in our tested domain, even to solutions of much better quality. For this reason this section evaluates our approach on pessimistic exploration.

3.6.1.1 Results with Optimal Plans

Results presented in this section are for the test domain as shown in Figure 3.1. High level plans generated by STRIPS planning and the abstract MDP are optimal at the lower RL level. The STRIPS plan is shown in Figure 3.2. The MDP-based plan leads to the same sequence of visited abstract states as in the STRIPS plan when the policy determined by the value function is followed from the start to the goal state.

In Figure 3.3 results are presented. They show the difficulty of the investigated maze problem in terms of exploration. In all 10 runs the no-shaping RL version was not able to learn to collect more than one flag. It quickly converged to a sub-optimal solution which takes only flag D and directly moves to position G (the goal position). The only approaches that were able to learn to collect all flags (though not in all runs) are using STRIPS-based and plan-flag reward shaping. At this time it is worth remembering that SARSA, like Q-learning, has asymptotic convergence guarantees. With this in mind, all algorithms (such as those presented in Figure 3.3) should, in

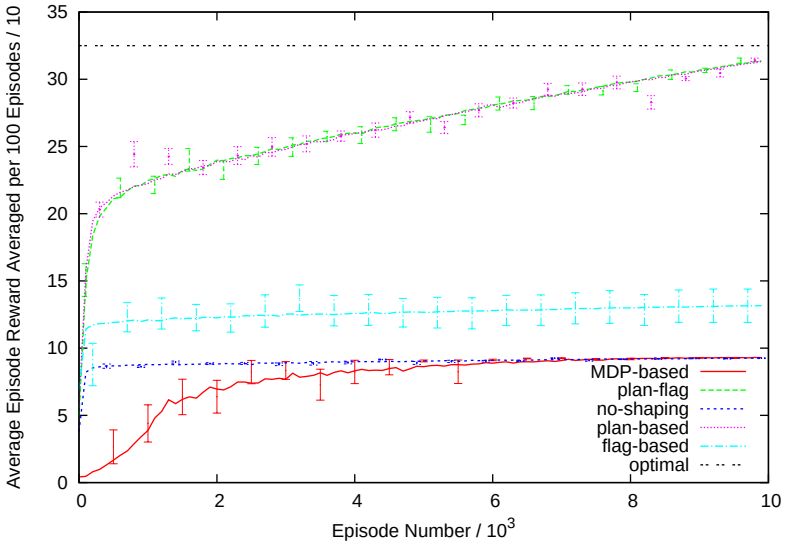


Figure 3.3: SARSA results with all reward shaping types, correct plans and pessimistic exploration.

theory, improve when one would continue the experiment for an infinite number of episodes. This is however infeasible and our experiments were stopped after most algorithms have converged to a relatively constant solution. This is sufficient to compare the exploration performance of different techniques when the interest is in the initial period of learning.

These experimental results show that this problem poses a challenge to model-free methods without background knowledge and is difficult to solve without properly used background knowledge.

In the above results, the MDP-based reward shaping displayed a notably worse performance than not only STRIPS-based but also less informed methods. A more detailed analysis was undertaken to look for the reason of this low performance. Some conclusions can be drawn from the analysis of the histogram (see Figure 3.4) which shows how many times each abstract state was entered in both STRIPS-based and MDP-based approaches. The presented graph is for a single run of SARSA. The first observation from Figure 3.4 is that the algorithm with an MDP-based plan tried many different paths, especially in the first episodes of learning. In the STRIPS-based case there is only one path along which the potential function increases. In the MDP-based case many different paths can be tried because the potential function increases along many paths when moving towards the goal. When the agent moves away from the plan it can still find a rewarded path to the goal because the MDP-based policy defines an optimal path to the goal, not only from the start but from all states. This led to a rather 'inconsistent' behaviour of the algorithm in the early stages of learning. The agent tried many different and advantageous paths, but because different paths were tried, they did not converge quickly enough. In effect, short and

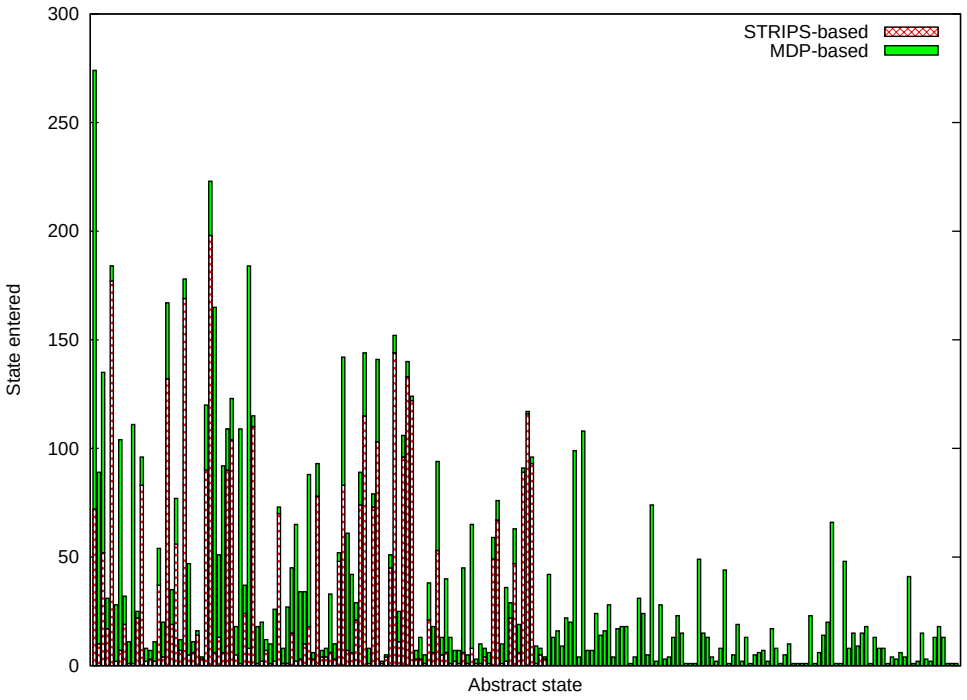


Figure 3.4: The histogram which presents how many times abstract states were entered during first 50 iterations of the single run of the SARSA algorithm.

sub-optimal paths, like, e.g., the one that goes from the start state, S, directly to the goal, G, after taking flag D, quickly dominate because they lead to a better immediate performance than very long paths that collect more flags, because longer paths need more time to back-propagate their high rewards. The histogram shown in Figure 3.4 provides more evidence for this hypothesis. First of all, it can be observed that the number of visited abstract states is almost twice larger in the MDP-based case. The agent considers a higher number of paths to be 'interesting' in this case. In this particular run, the number of visited abstract states was 106 in the STRIPS-based and 202 in the MDP-based case (there were 3141 and 6876 low level states visited respectively during the entire episode). Specifically, in the STRIPS-based case, the states that are visited when the optimal plan is followed are those with the highest number of visits in the histogram. Other abstract states which also have high values in the histogram are adjacent to those which follow the optimal path. It is worth noting that states which follow the optimal path are not visited very often in the MDP-based case.

The main conclusion from this empirical analysis is that in the case of model-free RL algorithms and a difficult problem domain (in terms of exploration), it may be better to assign the potential function according to one particular path which can converge quickly rather than to supply many paths. The latter raises the probability of converging to a sub-optimal solution

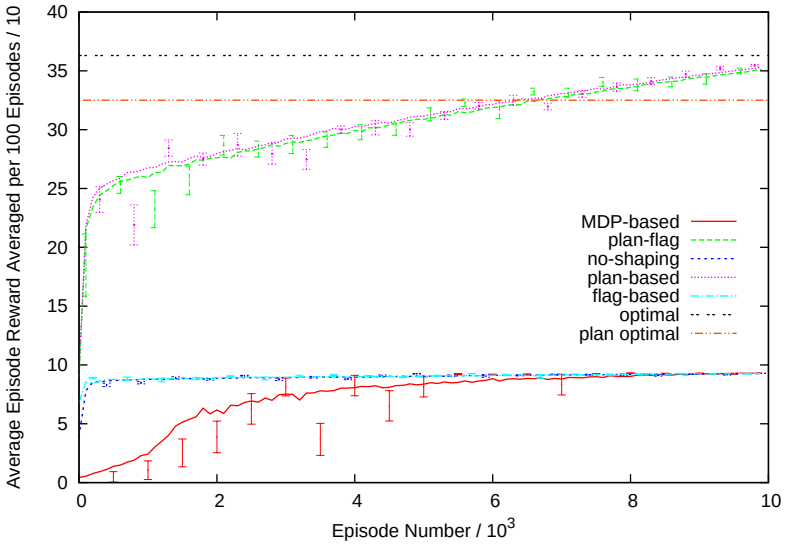


Figure 3.5: SARSA learning with pessimistic exploration when the planner does not know about the rigid fact `(next-to roomE roomC)`.

because many potential options are available.

This observation suggests one potential improvement to MDP-based reward shaping when problems discussed here may arise. Instead of using the value function for the entire state space as the potential function, the best path which corresponds to the STRIPS plan can be extracted. If this path was used with our algorithm (in the same way as the STRIPS plan) to define the potential function, it would direct the agent in a more focused way towards the goal when it can be easily misled by a suboptimal result otherwise.

3.6.1.2 Results with Sub-optimal Plans

In this section we take a closer look at various errors in plans used for reward shaping, which may be caused by incomplete or imprecise knowledge.

3.6.1.2.1 Plan Too Long Incomplete knowledge about the environment can lead to the situation when the planner computes a longer plan than necessary. In the actual environment, direct transitions from state z_i to state z_{i+k} where $k > 1$ may be possible, because the planning knowledge may not contain all facts. In order to test this scenario, we created an additional transition from room E to room B, which has not been taken into account in the computation of the STRIPS-plan. It means that the rigid fact `(next-to roomE roomC)` was not known to the planner. After collecting two flags in room E, the agent wants to collect flag B in room B. According to the suboptimal plan, it has to go through room C and hall B.

Empirical tests show that this kind of plan deficiency did not cause problems for the RL agent with plan-based reward shaping. Results are shown in Figure 3.5 and they show that the plan-

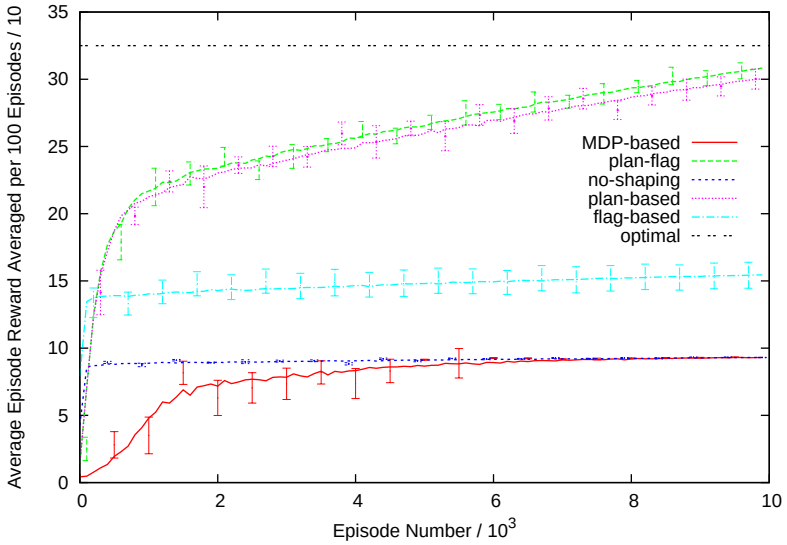


Figure 3.6: SARSA learning with pessimistic exploration when the planner assumes the existence of the transition E to B, i.e. the rigid fact `(next-to roomE roomC)` which does not exist in the actual environment.

based shaping reaches the performance which is higher than it is possible with the faulty plan (plan optimal in Figure 3.5). The transition from E to B, when encountered, is well rewarded because it has a higher difference in the potential function ($6w$ in room E and $9w$ in room B). Other algorithms yield results which are similar to those with correct knowledge.

3.6.1.2.2 Plan Assumes Impossible Transition Incorrect knowledge can cause also the opposite effect: connections between two states that are assumed by the plan knowledge may not exist in the actual environment. In our experiments, we created such a situation where the plan was computed assuming a connection between rooms E and B. That is the planner assumed that the rigid fact `(next-to roomE roomC)` is true, whereas it was not true in the environment. Learning with plan-based reward shaping was also successful in this configuration (see Figure 3.6) and 6 flags were collected in 7 out of 10 episodes.

3.6.1.2.3 Missing Goal Conditions In this experiment the plan was computed with a missing goal condition, thus potentially missing required actions, or including actions that are undoing part of the goal. The information about flag B has not been given to the planner. The question is whether the learning agent is able to find the missing element through exploration. This is principally possible because the proposed schema to assign the potential function to non-plan states does not penalise for moving away from the plan. The evaluation results show that none of the algorithms was able to overcome this difficulty as shown in Figure 3.7.

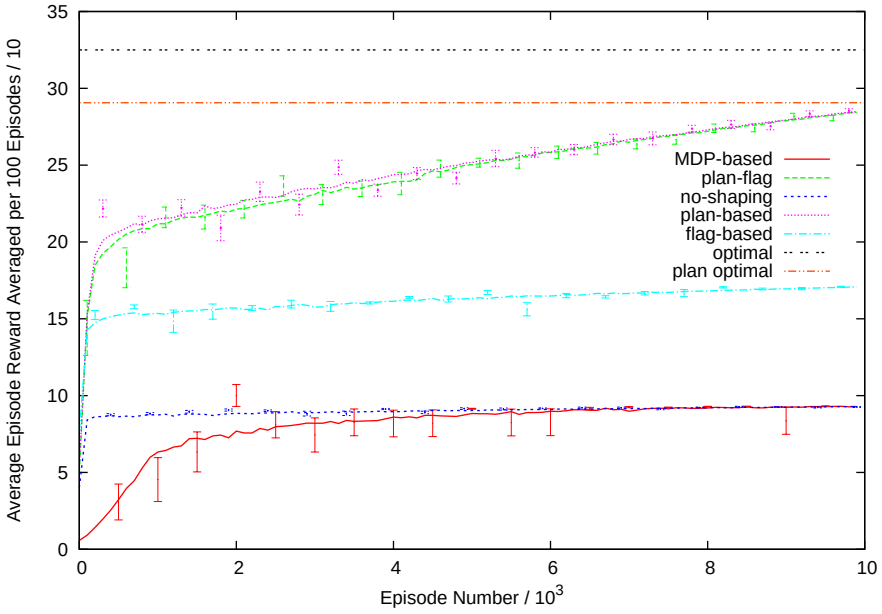


Figure 3.7: SARSA learning with pessimistic exploration when the planner does not know about flag B.

3.6.1.2.4 Wrong Sequence Even when high level knowledge about the domain is complete and the problem is specified correctly, there is one more factor which may lead to a sub-optimal policy at the low RL level. The main goal of classical planning algorithms is to find a plan which can transform the system from the start to the goal state. This achievement of the plan usually is satisfactory and the cost of actions is not taken into account in most STRIPS-based planners (Ghallab et al. 2004). Additionally, the cost of abstract actions would be unknown in a typical RL scenario because the cost of low level actions is also unknown in the standard scenario. In the approach introduced in this chapter, this may lead to sub-optimal plans when high level actions can have different cost when implemented by low level primitive actions. To test our algorithm with this deficiency of the plan, the experimental domain was modified in the following way. Halls A and B were merged into one hall and the high level plan was modified so flags were collected in the following order: B, A, C, E, F, D. This plan is clearly sub-optimal. Even though all flags are in the plan, there is another plan that results in a shorter travelled distance. This scenario was also difficult to tackle by most tested approaches (see Figure 3.8). Only the plan-flag reward shaping which combines two heuristics was significantly better than other algorithms though it reached performance which was close only to the optimal performance according to the faulty plan which was evaluated. The analysis of solutions obtained with the plan-based reward shaping showed that it was not only difficult to overcome deficiencies of this plan but it was also problematic for the algorithm to stick to this plan, since this plan was encouraging

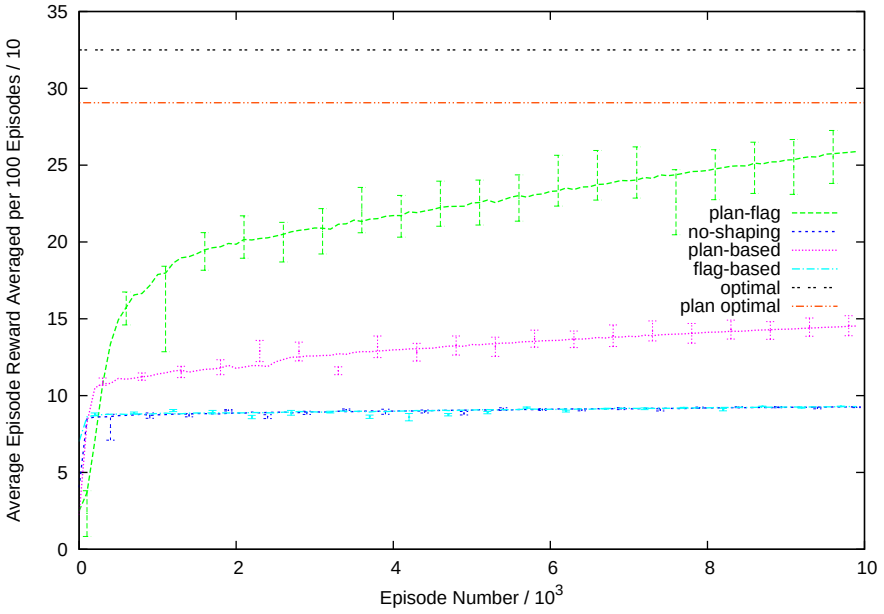


Figure 3.8: SARSA learning with pessimistic exploration when the plan is not correct (wrong sequence of abstract states).

suboptimal behaviour by following the path which was close to another suboptimal path (the agent is encouraged to collect flag A after B and it is much closer from A to D than to C, and this fact encourages wrong exploration). This experiment showed that, in the case of plan-based reward shaping the desirable property of plans is that they avoid such problematic situations by preferring more linear plans.

In summary, our results in this section show that even when plans are not optimal or contain errors, RL algorithms are performing the best when STRIPS-based reward shaping is used, even though they are not always able to converge to the optimum. Nevertheless, this can be satisfactory because the goal (especially with this kind of exploration) is often not to find the optimal solution but an acceptable policy in a reasonable amount of time.

3.6.2 Optimistic Exploration

In Section 3.6.1 the shaping reward was guiding the agent towards higher rewards when the agent was greedily biased towards currently promising solutions. In this section, the evaluation concerns the optimistic exploration, where the agent attempts to search the whole search space and the shaping reward is intended to cut this search space in the same way as good admissible heuristics help informed search methods such as best-first search (Russell & Norvig 2002).

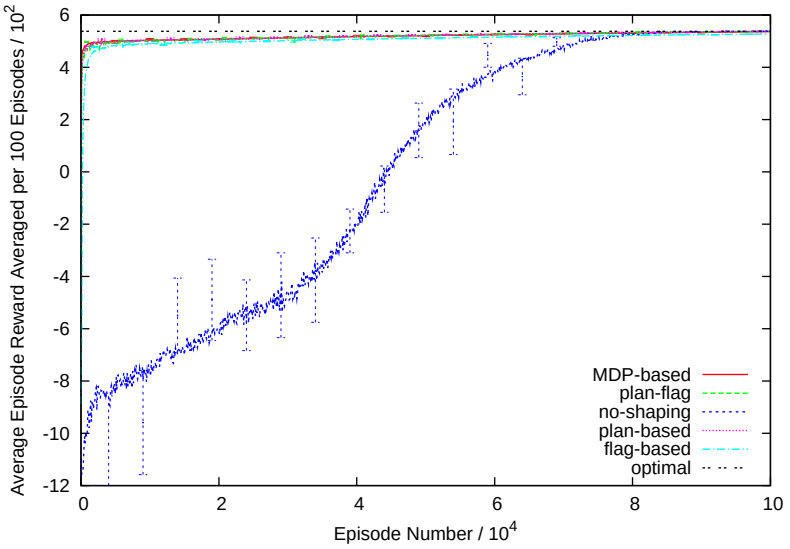


Figure 3.9: SARSA results with optimistic exploration and all reward shaping types.

3.6.2.1 Results with Optimal Plans

Results for the situation when the planning knowledge is correct and generated plans are optimal are presented in Figure 3.9. This experiment shows the standard problem of optimistic exploration. The agent who learns without reward shaping is able to find the optimal solution in terms of the number of collected flags but it requires a huge number of training episodes. It is worth emphasising that graphs in this section span over 10^5 learning episodes, whereas in the previous section this number was 10^4.

The first observation is that reward shaping, in general, yields immense speed-up with this type of exploration. As shown in Figure 3.9, only the non-shaping algorithm is learning very slowly. This difference can be easily explained when again referring RL with reward shaping to informed search. When reward shaping is used in RL, the situation is similar to the informed search problems where guidance given by the heuristic is extremely important to reduce the search space.

For a better comparison of different reward shaping methods, Figure 3.10 shows the zoomed-in graph. It shows that the non-shaping approach reaches finally the performance of the best performing algorithms. The flag-based approach leads to a suboptimal solution as expected. Plan-based approaches perform the best, and MDP-based is slightly worse than plan-based algorithm at the end of learning.

The overall outcome of this experiment is that in the case of optimistic exploration the difference in the quality of the heuristic which is used to shape the reward is less significant when fast learning of a reasonable solution is considered. All shaping rewards improve the performance

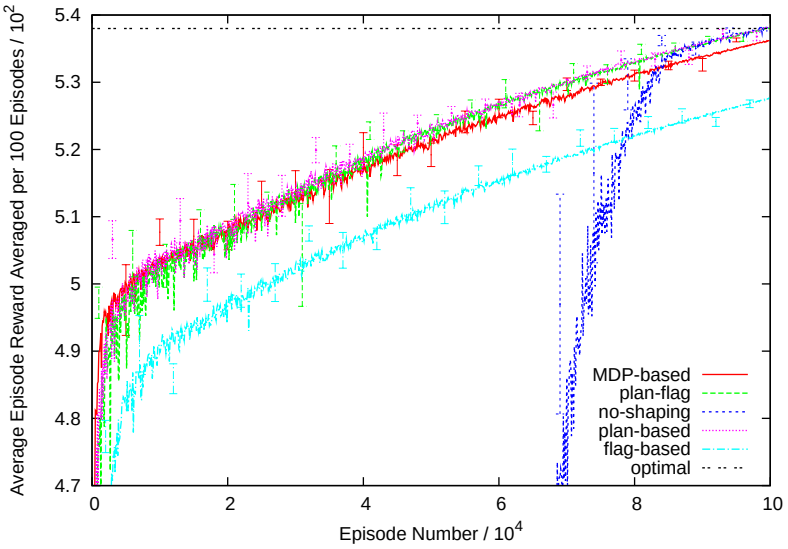


Figure 3.10: SARSA results with optimistic exploration and all reward shaping types.

to a very high extend when comparing with non-shaping learning. Wrong knowledge however which is incorporated in the flag-based reward shaping can be harmful in terms of the asymptotic convergence of the solution and optimistic exploration may not be sufficient to overcome such problems.

3.6.2.2 Results with Sub-optimal Plans

In this section, as well, various errors in plans used for reward shaping, which may be caused by incomplete or imprecise knowledge, are evaluated.

3.6.2.2.1 Plan Too Long As introduced in the corresponding experiment in Section 3.6.1, the situation is considered when planners do not know about the rigid fact `(next-to roomE roomC)`. The zoomed-in graph of this test is in Figure 3.11. The flag-based reward shaping is inferior in the same manner as in the experiment with correct knowledge. Plan-based and plan-flag based reward shaping have the biggest speed-up and gain the best asymptotic performance. The MDP-based approach is inferior to the two former approaches though it overcomes plan deficiencies and learns a slightly better policy than the one defined by the plan. This type of knowledge incorrectness turned out to be more harmful for the MDP-based learning. The plan-based reward shaping is not based on the value function and the fact that the short-cut is discovered may still yield shaping rewards which encourages correct behaviour whereas the MDP-based approach is left with an entirely wrong policy determined by the value function.

3.6.2.2.2 Plan Assumes Impossible Transition In this test, the planner assumed that the rigid fact `(next-to roomE roomC)` is true whereas it is was not true in the environment.

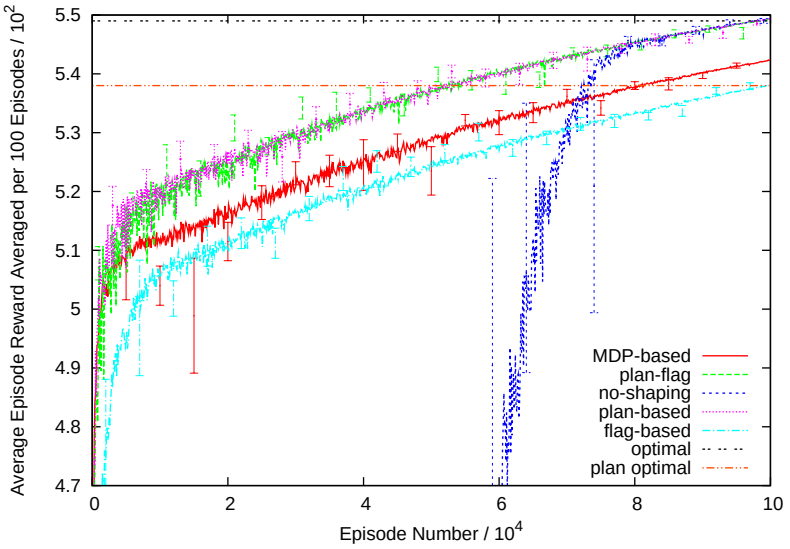


Figure 3.11: SARSA learning with optimistic exploration when the planner does not know about the rigid fact `(next-to roomE roomC)`.

Results of this experiment are in Figures 3.12 and 3.13. The general pattern of the behaviour of tested algorithms is similar as in previous experiments. The specific issue which is interesting with regard to this analysis is the behaviour of plan-based and MDP-based reward shaping methods. Initial episodes of learning with MDP-based shaping found in Figures 3.12, show that the algorithm experienced unstable behaviour due to wrong shaping rewards. Episodes were longer than in other shaping methods. However, after initial period of instability the algorithm improved (corrected inaccuracies which result from the wrong plan) and was systematically improving, though some instability can still be observed even in our smoothed graphs. The plan-based reward shaping was not as robust when facing this type of knowledge incorrectness. The same factor which made this method perform well in the previous test, this time was detrimental. When the plan fails, that is, it is not possible to follow the plan; the algorithm is left without any heuristic guidance. The plan-flag approach was better in this case than pure plan-based learning because the flag-based potential function served as the missing guidance.

3.6.2.2.3 Missing Goal Conditions In this experiment the planner does not know about flag B. Results of this experiment are in Figures 3.14 and 3.15. Non-shaping learning is extremely slow as in the previous experiments however it obtains the best asymptotic performance. MDP-based reward shaping had good initial improvement however it was not able to overcome plan deficiency and was not able to find the missing goal predicate. It converged successfully only to the solution which is optimal according to faulty plan knowledge (plan optimal in Figures 3.14 and 3.15). Its failure can be attributed to the fact that the potential function based on the value

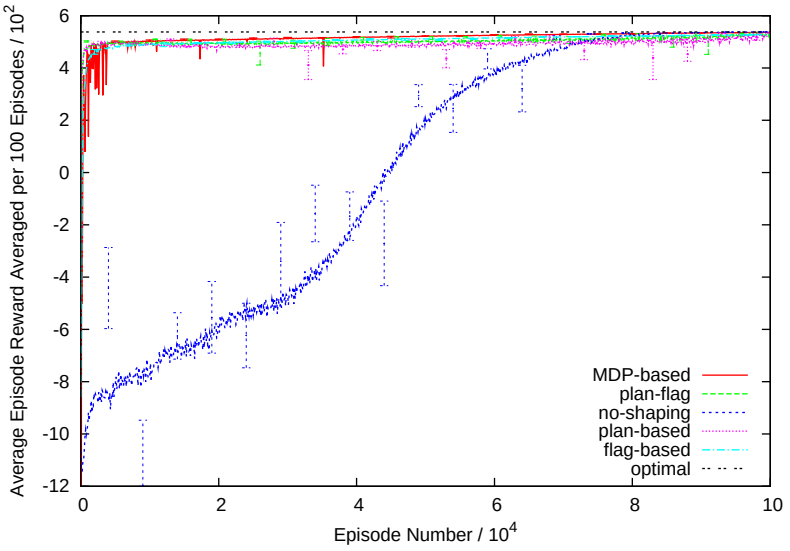


Figure 3.12: SARSA learning with optimistic exploration when the planner assumes the existence of the transition E to B, i.e. the rigid fact `(next-to roomE roomC)` which does not exist in the actual environment.

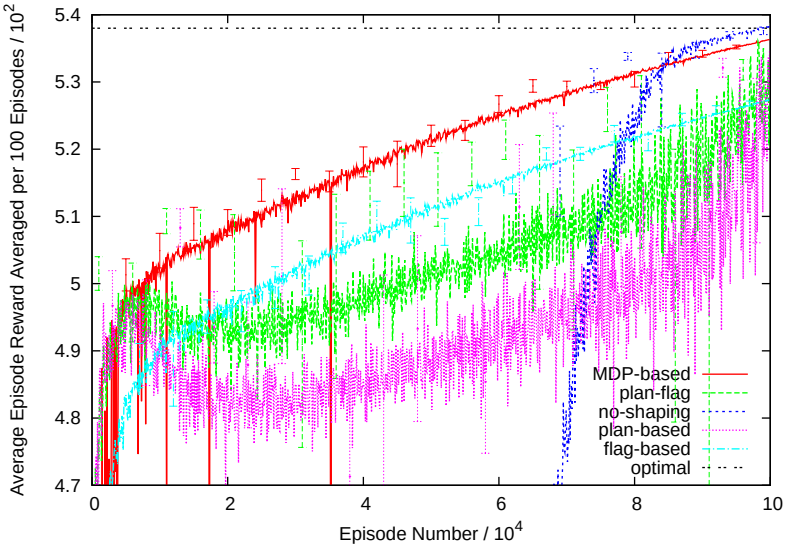


Figure 3.13: SARSA learning with optimistic exploration when the planner assumes the existence of the transition E to B, i.e. the rigid fact `(next-to roomE roomC)` which does not exist in the actual environment.

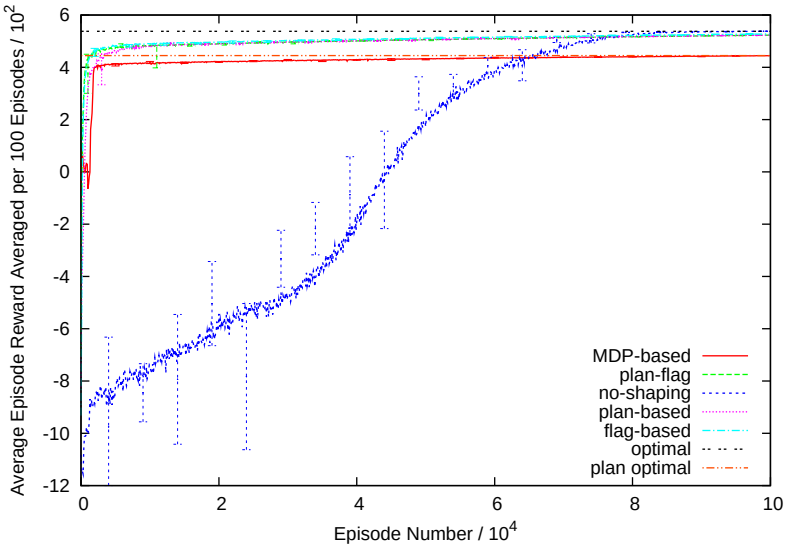


Figure 3.14: SARSA learning with optimistic exploration when the planner does not know about flag B, i.e. when one of goal predicates is missing.

function is consistently encouraging the agent to move towards suboptimal goals. Any attempt of moving away from the plan is penalised by the reward shaping in this case. In the case of plan-based reward shaping, such attempts to explore are neutral, i.e. not penalised by the shaping reward. In this test, the best speed-up was due to flag-based shaping. Plan-based managed to learn about the missing predicate however the wrong plan led to slower convergence than the flag-based learning in this case.

3.6.2.2.4 Wrong Sequence The full description of this type of knowledge incorrectness is given in the corresponding paragraph in Section 3.6.1. The overall problem here is that the planning knowledge is wrong and the plan encourages the agent to follow suboptimal path which collects flags in a wrong order. Results of this test are in Figure 3.16. In this case planning knowledge encourages the suboptimal solution which is equivalent to the flag-based heuristic. In effect, all reward shaping algorithms converge to the same suboptimal quality.

3.7 Plan-based Reward Shaping with Function Approximation

One of the observations from experiments in Section 3.6 is the fact that plan-based reward shaping is particularly encouraging and significantly better than alternative techniques when applied to pessimistic exploration. This kind of exploration is potentially suitable for huge states spaces in model-free RL where one wants to concentrate the computation on the most interesting parts of the state-space, neglecting highly unlikely state-transitions (Meuleau et al. 1999). The plan-based

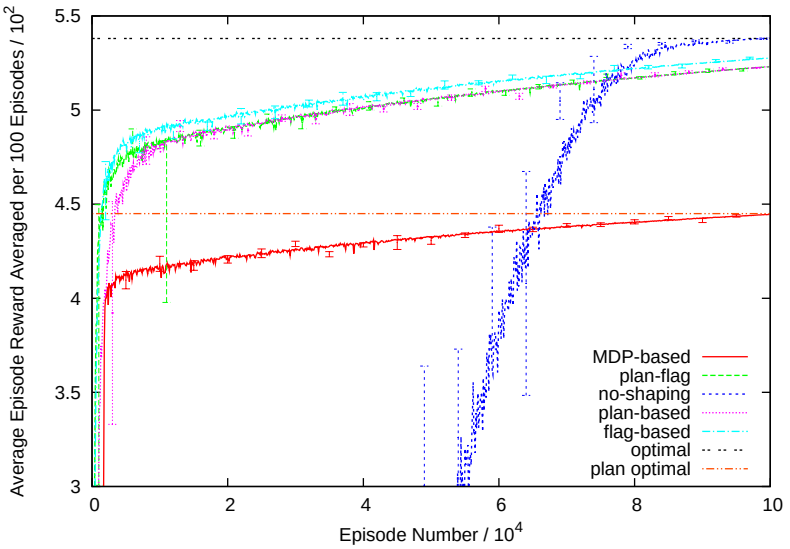


Figure 3.15: SARSA learning with optimistic exploration when the planner does not know about flag B, i.e. when one of goal predicates is missing.

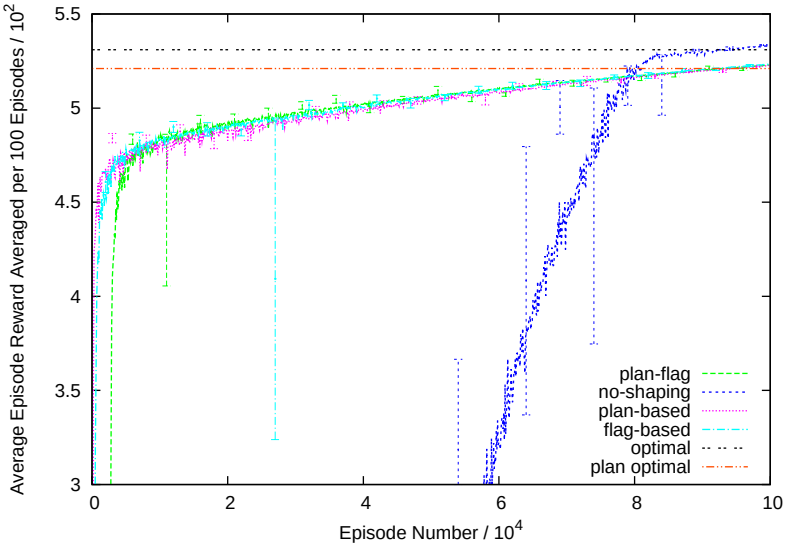


Figure 3.16: SARSA learning with optimistic exploration when the plan is not correct (wrong sequence of abstract states).

reward shaping pushes the agent towards the goal state along one particular path and significant parts of the state space can be omitted (see histogram shown in Figure 3.4, where plan-based reward shaping reduces the number of visited states during learning when compared to the MDP-based method). Thus, the agent can avoid the greedy exploration of the entire state space and learn a specific path in this state space when shaped by the plan-based reward shaping as shown in our experiments. When the Q-table is implemented efficiently, for example, as a hash table, the policy can be found with only a fraction of states stored in the Q-table. Such, a solution was applied in our experiments. Another idea is to use more sophisticated function approximation which allows for generalisation (Sutton & Barto 1998).

Recent research on the use of function approximation in RL (Wu & Meleis 2008, 2009a,b) matches and supports the way we motivate our work on plan-based reward shaping. Wu & Meleis (2008) apply Kanerva coding to approximate the value function in huge (as tested on multi-agent problems) domains where techniques with local basis functions like tile coding or radial basis functions cannot cope with the exponential state space explosion. The Kanerva coding is based on choosing features that correspond to particular states called prototype states. Thus, features are based on selected states in the domain state space instead of state features. In this way, the complexity depends entirely on the number of selected prototype states, which is not necessary related to the dimensionality of the task (see Section 8.3.4 in Sutton & Barto 1998 for a discussion of the problem). The work of Wu & Meleis (2008) aims at optimising the set of prototype states by minimising its cardinality and maximising its expressive power in representing the approximated value function. Their evaluation in (Wu & Meleis 2008) is based on the ϵ-greedy pessimistic exploration because such an exploration guarantees that the algorithm relatively quickly focuses on the solution which is the best at a given time. Wu & Meleis (2008) do not mention this, but their approach may encounter problems when used with optimistic exploration because of exhaustive exploration which would increase frequency of states which can be neglected. Thus, we conjecture that this function approximation technique can be profitable only when exploration can be focused in a way which avoids visiting unnecessary states and exploits those states which are necessary for precise representation of the value function. This is actually provided by our plan-based reward shaping. This explanation is to show that our idea of focusing on pessimistic exploration with plan-based reward shaping can be used to improve related work. Specifically, a joint application of plan-based reward shaping and the adaptive Kanerva coding of Wu & Meleis (2008) could be seen as an interesting future work.

Another important remark comes from the fact that in order to tackle huge problems function approximation with global basis functions is sometimes unavoidable (Tesauro 1994). The problem with this kind of function approximation is that it is difficult or in most cases even impossible to guarantee optimistic exploration. If the approximation is initialised optimistically, its values rapidly change in the entire state space due to global impact of these kinds of basis functions (Bishop 1996). This fact highlights the need for efficient methods of improving pessimistic ex-

ploration. This type of exploration is in principle more suitable for large domains and thus our plan-based reward shaping is promising with this regard as well.

3.8 Plan-based Reward Shaping in Model-based Learning

In Section 3.7, the potential applicability of plan-based reward shaping with function approximation in huge state spaces was emphasised. Related literature (Meuleau et al. 1999) considers model-free RL as a possible solution to overcome the curse of dimensionality of the state space, in general, and function approximation can be applied with this type of RL in the most straightforward way. This was another reason why our empirical evaluation of plan-based reward shaping, in Section 3.6, was performed on the model-free algorithm. The abstract level knowledge can span over large state spaces and this makes it particularly useful for learning policies in such huge state spaces which can be effectively represented and learned with model-free algorithms.

Thus, as already emphasised, the plan-based reward shaping is suitable for larger problems with high level knowledge where approximate solutions are satisfactory and the overall goal of learning is to obtain such a reasonable solution as fast as possible. Standard model-free algorithms like SARSA, or in some situations model-based DynaQ (when not all states are stored in the Q-table, pessimistic or similar exploration is used, and appropriately small number of simulated Bellman planning backups is performed Sutton 1990), are suitable for this task because of the way they balance representational and computational cost. The DynaQ algorithm represents a rather flexible extension which incorporates learning of the MDP model into RL algorithms. When used with small number of planning steps, its behaviour is similar to model-free algorithms like Q-learning or SARSA. On the other hand, when used with properly designed optimistic initialisation and an appropriately large number of planning steps it can also be close in its behaviour to PAC-MDP algorithms (Kakade 2003; Strehl & Littman 2008). Thus, its performance would depend on particular design and the way it is deployed. Our experiments on model-free RL presented in Section 3.6 and additional experiments with the DynaQ algorithm in (Grześ & Kudenko 2008a) indicate that its performance with pessimistic exploration is similar to the model-free learning, where the plan-based learning leads to the best results. Such DynaQ configuration would be the only feasible model-based solution for large state spaces, however the fact that the MDP model has to be learned is already cumbersome enough. More exhaustive exploration incorporated in this method would make its application even more difficult in huge state spaces. Discussion of such exhaustive model-based methods with regard to our reward shaping approach is in the next paragraph.

The PAC-MDP type of model-based learning is about learning a solution which is optimal with high probability, but this approach has very rigorous requirements which make it difficult to apply to domains with huge state spaces (Brafman & Tennenholtz 2002). Additionally, in order to apply reward shaping to such algorithms the potential function has to be strictly admissible. Otherwise the algorithm will not be PAC-MDP any more (Asmuth et al. 2008). Our initial anal-

ysis of the plan-based reward shaping showed that it would be difficult to design an admissible potential function and thus to apply this technique to PAC-MDP algorithms in a way which would preserve the PAC-MDP property. Knowledge which is used in our algorithm is of a high and abstract nature and yields improvements as shown in the experimental results where such heuristic knowledge, even though faulty, may still be very useful for non-PAC-MDP algorithms. This kind of knowledge is potentially available in large realistic domains and this brings higher potential of application of our idea. PAC-MDP algorithms do not deal well with huge state spaces, because of the model which has to be learned and the immense number of planning steps which are required by these types of algorithms (Brafman & Tennenholtz 2002). For this reason the issue of developing a plan-based reward shaping algorithm for PAC-MDP algorithms which would preserve their theoretical properties is left as a future, challenging work. Challenging because one has to tackle abstract knowledge in this case. However, as explained in this section, the proposed plan-based reward shaping technique is more suitable to model-free algorithms, like SARSA, and to standard model-based algorithms, like DynaQ, and this fact underlines its potential of practical applicability to larger domains where existing PAC-MDP algorithms have limited applicability anyway. In order to apply PAC-MDP algorithms to large problems, some theoretical guarantees have to be relaxed, but such relaxation would violate their theoretical requirements and reduce them to special cases of standard model-based algorithms like DynaQ (Sutton 1990).

A knowledge-based approach for PAC-MDP algorithms with appropriate low level knowledge is considered in Chapter 6 where knowledge-based PAC-MDP learning is investigated and proposed extensions meet all theoretical requirements of this type of RL.

3.9 Summary and Discussion

In this chapter we show a new method to define the potential function for potential-based reward shaping using abstract plan knowledge represented in the form of STRIPS operators. We empirically compared the performance of our approach with the performance of RL without any reward shaping, RL with a manually shaped reward, as well as a related automatic shaping approach based on abstract MDPs (Marthi 2007). The results of the experiments demonstrate that STRIPS-based reward shaping improves in certain situations both the quality of the learned policy and the speed of convergence over the alternative techniques. The evaluation was performed with two types of exploration, which yields also a deeper insight into the effect of reward shaping on model-free learning and constitutes an additional contribution of this chapter.

Overall, the results of this chapter can be summarised as follows:

1. Pessimistic exploration in conjunction with model-free RL represents a good approach for solving large RL problems because of the way it deals with huge state spaces (i.e. the pessimistic exploration guarantees that unnecessary state action pairs can be neglected and model-free RL does not have to learn the model). But, such an approach may result in extremely poor results when the domain is difficult in terms of exploration, e.g., when

there are many goals with different rewards. The first part of the experimental section of this chapter showed that this kind of RL can be successfully implemented, even in domains with difficult exploration, when used with plan-based reward shaping.

2. The strong point of the plan-based reward shaping is that it allows agents to avoid exploring and storing many unnecessary states in the Q-table. This was confirmed in our experiments and this property makes it particularly suitable also with more sophisticated function approximation methods which can profit from informative avoidance of unnecessary states, e.g., the Kanerva coding implementation in (Wu & Meleis 2008, 2009a,b).

3. Tests with optimistic exploration showed that already in our artificial test domain this kind of exploration requires a significant number of episodes to converge (this is a standard disadvantage of this type of exploration), though the non-shaping learning can find the optimal solution with regard to the number of collected flags. Any type of reward shaping is extremely helpful in cutting the search space of optimistic exploration, however the quality of the reward shaping plays an important role when asymptotic convergence is considered. The plan-based reward shaping was also very competitive under this criterion.

4. STRIPS-based shaping showed in several cases better results than the MDP-based approach, because the agent was strongly influenced by the plan that guides it towards a good policy. Thus, this observation suggests one potential improvement to MDP-based reward shaping. Instead of using the value function of the entire state space as the potential function, the best path which corresponds to the STRIPS plan can be extracted and used with our algorithm to define the potential function.

5. Evaluation with wrong knowledge showed that specific reward shaping methods are more or less resistant to certain knowledge inaccuracies, and general relationships were identified. For example, the MDP-based reward shaping is not resistant to situations when planning knowledge does not include all goal predicates. Plan-based reward shaping was the most error prone in the situation when inaccurate knowledge assumed high level transitions which do not exist in the actual environment. These observations should be taken into account when deploying reward shaping and when certain predictions can be made on what kind of things may be wrong in the abstract model of the domain.

STRIPS-based approaches can deal with much larger state spaces at an abstract level because states are not explicitly enumerated (Boutilier et al. 1999). Symbolic planners can solve large problems (with huge state spaces) through their compact and highly abstract representations of states. Such planning together with model-free RL (with which STRIPS-based planning works well) can therefore be used with large state spaces and with function approximation at the RL level in particular. It is worth noting that function approximation has been up to now used mainly with model-free RL algorithms and SARSA in particular (Stone et al. 2005).

STRIPS-based reward shaping is easier to scale up than, e.g., MDP-based reward shaping. In MDP-based abstract planning the state space has to be enumerated, which may require stronger abstraction or function approximation when applied to RL domains with very large state spaces. However, a positive feature of MDP-based planning is that it can deal in a natural way with different costs of high level actions (something that is more difficult to achieve with STRIPS).

Overall, STRIPS-based reward shaping can be seen as an alternative to MDP-based reward shaping with the proposed extension as both of these techniques are planning methods. It is up to the domain expert which method to choose, depending on the form of available knowledge. If one has symbolic knowledge, it is very easy to apply it as a way of learning shaping rewards with our plan-based technique.

CHAPTER 4

Reward Shaping and Mixed Resolution Function Approximation

A crucial trade-off is involved in the design process when function approximation is used in reinforcement learning. Ideally the chosen representation should allow representing as closely as possible an approximation of the value function. However, the more expressive the representation the more training data is needed because the space of candidate hypotheses is larger. A less expressive representation has a smaller hypotheses space and a good candidate can be found faster. The core idea of this chapter is the use of a mixed resolution function approximation, that is, the use of a less expressive function approximation to provide useful guidance during learning, and the use of a more expressive function approximation to obtain a final result of high quality. A major question is how to combine the two representations. Two approaches are proposed and evaluated empirically: the use of two resolutions in one function approximation, and a more sophisticated algorithm with the application of reward shaping.

4.1 Introduction

In contrast to supervised learning, RL agents are not given instructive feedback on what the best decision in a particular situation is. This leads to the *temporal credit assignment* problem, that is, the problem of determining which part of the behaviour deserves the reward (Sutton 1984). To address this issue, the iterative approach to RL applies backpropagation of the value function in the state space. Because this is a delayed, iterative technique, it usually leads to a slow convergence, especially when the state space is huge. In fact, the state space grows exponentially with each variable added to the encoding of the environment when the Markov property needs to

be preserved (Sutton & Barto 1998).

When the state space is huge, the tabular representation of the value function with a separate entry for each state or state-action pair becomes infeasible for two reasons. Firstly, memory requirements become prohibitive. Secondly, there is no knowledge transfer between similar states and a vast number of states need to be updated many times. The concept of value function approximation (FA) has been successfully used in reinforcement learning (Sutton 1996) to deal with huge or infinite (e.g., due to continuous variables) state spaces. It is a supervised learning approach which aims at approximating the value function across the entire state space. It maps values of state variables to the value function of the corresponding state.

A crucial trade-off is involved in the design process when function approximation is used. Ideally the chosen representation should allow representing as closely as possible an approximation of the value function. However, the more expressive the representation the more training data is needed because the space of candidate hypotheses is larger (Mitchell 1997). A less expressive representation has a smaller hypotheses space and a good candidate can be found faster. Even though such a solution may not be particularly effective in terms of the asymptotic performance, the fact that it converges faster makes it useful when applied to approximating the value function in RL. Specifically, a less expressive function approximation results in a broader generalisation and more distant states will be treated as similar and the value function in this representation can be propagated faster. The core idea of this chapter is the use of a mixed resolution function approximation, that is, the use of less expressive FA to provide useful guidance during learning and the use of more expressive FA to obtain a final result of high quality. A major question is how to combine the two representations. The most straightforward way is to use two resolutions in one function approximation. A more sophisticated algorithm can be obtained with the application of reward shaping. The shaping reward can be extracted from a less expressive (abstract) layer and used to guide more expressive (ground) learning.

To sum up: in this chapter we propose combining more and less expressive function approximation, and three potential configurations are proposed and evaluated:

- the combination of less and more expressive representations in one approximation of the value function,
- the use of less expressive function approximation to learn the potential function for reward shaping which is used to shape the reward of learning with desired resolution at the ground level,
- the synergy of the previous two, that is, learning the potential function from less expressive approximation and using it to guide learning which combines less and more expressive resolution in one FA at the ground level.

Our analysis of these ideas is based on tile coding (Lin & Kim 1991) which is commonly used for FA in RL. The proposed extensions to RL are however of general applicability and can be used

with different methods of function approximation, especially those which use basis functions with *local* support (Bishop 1996).

The rest of this chapter is organised as follows. In the next section, function approximation with tile coding is introduced. A more general discussion of function approximation is in Section 2.5. Learning with mixed resolution tile coding is presented in Section 4.3 and the algorithm which learns the potential function for reward shaping is discussed in Section 4.4. The experimental validation of the proposed extensions to RL is in Sections 4.5-4.8. Section 4.9 summarises this chapter.

4.2 Background

Tile coding is introduced in this section. In particular, the dependency of the resolution and the generalisation power of tile coding is highlighted and shown as a motivation for this work.

4.2.1 Value Function Approximation with Tile Coding

Value function approximation takes advantage of the fact that states with similar values of state features have in most cases a similar value of the value function. The idea is to represent the value function, V, as a vector of parameters, θ, with the size, N, of this vector smaller than the number of states. In this way the update of the value function according to one state is generalised across similar states (Sutton 1996).

Function approximation should be fast and allow for online learning. Linear functions with updates based on gradient-descent methods meet this requirement. The linear approximation of the value function for action a can be expressed in the following form:

$$V^a(s) = \sum_{i=0}^{N-1} \theta_i^a \phi_i(s), \tag{4.1}$$

where $\phi_i(s)$ is a basis function.The gradient-descent update rule for this approximation takes the form:

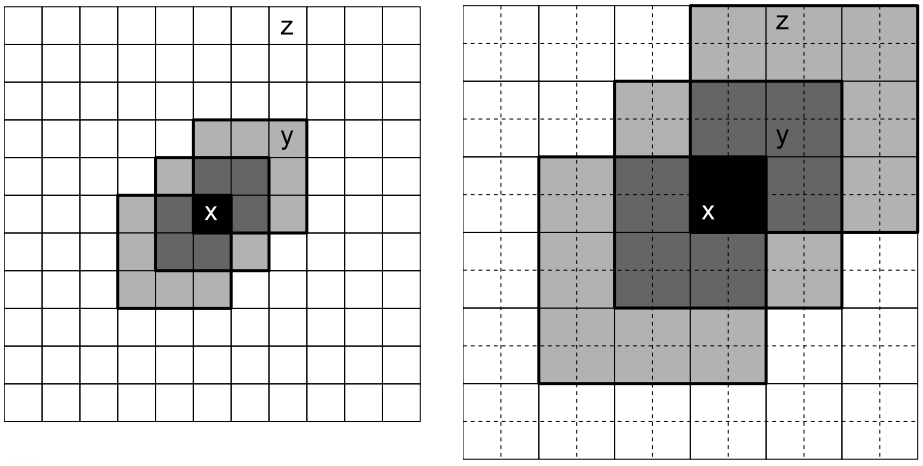
$$\theta' = \theta + \alpha \delta_t \phi(s), \tag{4.2}$$

where α is the learning rate and δ_t is the temporal difference:

$$\delta_t = r + \gamma V^{a'}(s') - V^a(s). \tag{4.3}$$

The immediate reward is represented by r, γ is the discount factor, and s and s' are two consecutive states.

Tile coding (Sutton 1996) is a particular method to define a basis function, $\phi_i(s)$, for states or state-action pairs. This method partitions the input space into several displaced layers (tilings) of overlapping tiles. Each state can be allocated to exactly one tile in each tiling. Thus, $\phi_i(s)$ takes value 1 for tiles it is allocated in and 0 otherwise. Figure 4.1 shows how it can be determined in



a) b)

Figure 4.1: Tile coding examples with a different resolution. Three tilings with tiles of three units in a) and six units in b).

a 2D space. Tiles allow for generalisation to neighbouring positions. For example, an update of the value function in position x has an impact on the value function in position y which may not be visited during the entire period of learning. One of the key motivations to propose the algorithms introduced in Sections 4.3 and 4.4 is the fact that coarser generalisation (see Figure 4.1b) allows for a more rapid propagation of the value function. This coarser generalisation means that the resulting representation is less expressive, but it can be used to guide learning of the value function with a more detailed representation.

4.2.2 Reward Shaping

Potential-based reward shaping has been introduced in detail in Section 2.4 of the thesis. One problem of this idea is that often detailed knowledge of the potential function of states is not available or is very difficult to represent directly in the form of a shaped reward. When the shaping reward is computed as:

$$F(s, s') = \gamma\Phi(s') - \Phi(s), \tag{4.4}$$

the application of reward shaping reduces to the problem of how to learn the potential function, $\Phi(s)$, and in this chapter a method to address this issue is proposed. We suggest learning the potential function online as the value function of a coarse, abstract tile coding. At this time it is worth reconsidering (from Section 2.4) that a particularly convenient potential function would be the one which is equal to the value function, that is, $\Phi(s) = V(s)$, which helps justify why roughly approximating the value function is a promising approach for reward shaping. The algorithm is introduced in Section 4.4.

4.3 Mixed Resolution Tile Coding

In this section we introduce a RL architecture that treats both the fine and coarse tilings as parts of the same function approximator. This straightforward idea can be easily found in Figure 4.1. Basically, two tilings with different resolution are used. The less expressive one with a coarse resolution is intended to allow for broader generalisation early and the more expressive with a fine resolution to yield refinement later on. Now, the value function can be represented as two vectors of parameters θ^c and θ^f for coarse and fine tilings correspondingly. To these tilings correspond also two basis functions ϕ^c and ϕ^f. In this setting the value function is computed as:

$$V(s) = \sum_{i=0}^{N^c-1} \theta_i^c \phi_i^c(s) + \sum_{i=0}^{N^f-1} \theta_i^f \phi_i^f(s), \tag{4.5}$$

where $N^c = |\theta^c|$ and $N^f = |\theta^f|$. For the value function computed in this way, the temporal difference can be evaluated in a standard way according to Equation 4.3 and vectors θ^c and θ^f updated according to the gradient descent rule in Equation 4.2.

This method allows for a natural coexistence of two resolutions in one function approximator. It can be seen as a method of obtaining and using high level knowledge to guide early learning.

The next section shows how to use this knowledge in a different way. Reward shaping is proposed as another way of using knowledge which is provided by the coarse resolution to speed up learning with a more detailed resolution.

4.4 Learning the Potential Function for Reward Shaping

We propose a RL architecture with two levels of tile coding. The first one learns an approximation of the Q-function at the ground RL level. The second, coarser one learns an abstract V-function which is used as the potential function to calculate the shaping reward (see Equation 4.4) for the ground level. The algorithm which is proposed here builds on two techniques existing in the field: 1) multigrid discretization used with MDPs (Chow & Tsitsiklis 1991), and 2) automatic shaping which was recently proposed (Marthi 2007).

4.4.1 Related Work

The multigrid discretization in the MDP setting (Chow & Tsitsiklis 1991) was used to solve an MDP in a coarse-to-fine manner. While this technique is well suited to dynamic programming methods (a coarse problem at a high, abstract level can be solved and used at a more detailed, ground level), there was no easy way of merging layers with a different resolution when applied to RL algorithms. First such attempts were made by Anderson & Crawford-Hines (1994) and this problem was evident in their work. The need for knowledge of the topology of the state space is necessary in their solution to define how multiple levels are related, and this fact made the approach infeasible for RL tasks. It used a multigrid as a way of obtaining knowledge, but

the mechanism to use this knowledge at a ground RL level was missing. We propose potential-based reward shaping as a solution to these problems. The ground RL algorithm does not have to be modified and knowledge can be given in a transparent way via an additional shaping reward. In this work, the idea of multigrid discretization is reflected in two different resolutions in tile coding.

In the automatic shaping approach (Marthi 2007) an abstract MDP is formulated and solved. In the initial phase of learning, the model of an abstract MDP is built and after a defined number of episodes an abstract MDP is solved and its value function used as the value of the potential function for ground states. We propose an algorithm which applies tile coding with different resolutions to create ground and abstract levels. Instead of defining an abstract task as dynamic programming for solving an abstract MDP, we use RL to solve the abstract task online. RL with representation based on tile coding results in a natural translation between ground and abstract levels. Tile coding in itself can be easily applied in a multigrid fashion and because it has been mostly used with model-free RL and SARSA in particular[1], it is sensible to apply RL for solving an abstract level problem. Tile coding is an important and popular function approximation method for model-free learning, and our approach meets requirements of model-free RL with tile coding. Our aim is to have more robust model-free learning with tile coding, while still enjoying all properties of model-free learning. Additionally, knowledge about the environment which is used to define tile coding at the ground level is sufficient to deploy our method in its basic form.

Work on tile coding which is related to this chapter was presented in (Zheng et al. 2006) where two function approximations with tile coding were also applied. In this case, Q- instead of the V-function is used at an abstract level. The high level, abstract Q-values are used to guide the exploration in the initial learning phase. This approach lacks the reference to the potential-based reward shaping as results in (Zheng et al. 2006) do not indicate a clear advantage of that method. Without the robust mechanism of potential-based reward shaping, the Q-function needed to be used at an abstract level. The usage of the V-function would require for example approximating transition probabilities. In our case, it is enough to learn only the V-function which can converge sufficiently faster to be useful for potential-based reward shaping.

The variable resolution discretization has been studied in the field (Munos & Moore 2002). The idea is to split some cells (states) and bring a higher resolution to some areas of the state space in order to represent a better policy. Our approach can be seen as orthogonal to this technique because they could be combined together and bring their distinct merits to the overall solution. We learn the shaping reward which can be used to guide ground learning with a variable resolution discretization. The interesting question arises, whether a variable resolution could improve the process of learning a potential function when applied at an abstract level and focused on fast propagation of guidance. When applied at the ground level it is intended to play the opposite

[1]Empirical results in the literature (Stone et al. 2005) show that SARSA is generally better than Q-learning when tile coding is used. The explanation is justified in the literature by the fact that SARSA is an on-policy method.

role, i.e. to provide a higher resolution where it is necessary (Munos & Moore 2002).

The relationship of the number of tilings and the interval size was studied by Sherstov & Stone (2005). Their results show that a smaller number of tilings with wider intervals speeds up learning in initial episodes but hurts convergence at later stages. In contrast, narrower intervals (with preferably one tiling) slow down initial learning but lead to a higher quality of the final solution. Choosing in our algorithm a fine grained encoding with a small number of tilings at the ground level and coarse generalisation for reward learning can be seen as an easy way to have fast convergence at the beginning and good convergence at the end of learning.

Because in our algorithm learning takes place at two levels of abstraction, it is worth relating this approach to the general concept of hierarchical machine learning. Stone & Veloso (2000) proposed the universal idea of layered learning where the search space of hypotheses can be reduced by a bottom-up, hierarchical task decomposition into independent subtasks. Each local task is solved separately, and tasks are solved in a bottom-up order. The distinguishing feature of this paradigm is that the learning processes at different layers do not interact with each other and different machine learning algorithms can be used at different layers. In particular, RL was applied to learn in this architecture (Stone & Veloso 2000), i.e. to learn at a particular layer. Because tasks are solved independently using results from learning at lower layers, the algorithm proposed in this chapter can be seen as a potential choice for selected subtasks.

When relating our algorithm to hierarchical reinforcement learning it is worth noting how the hierarchy interacts with reinforcement learning in such algorithms. Regardless of the type of abstraction used to create hierarchy (e.g. state abstraction, hierarchical distance to the goal Kaelbling 1993; Moore et al. 1999, feudal reinforcement learning Dayan & Hinton 1993, temporal abstraction Parr & Russell 1997; Sutton et al. 1999, or both state and temporal abstractions Dietterich 2000) the hierarchy exists in the final representation of the solution, i.e. the policy is defined on this hierarchy, and learning may take place at all levels of the hierarchy simultaneously. The value function is a function of not only the ground states and actions but also some elements determined by the hierarchy (e.g., in Parr & Russell 1997 HAMQ-learning maintains an extended Q-table $Q([s, m], a)$ indexed by a pair of states which includes state s and machine state m, and an action a at a choice point). In our algorithm the actual RL is not modified and the abstract level learning provides feedback which is given in a transparent way via reward shaping. There is also no need for knowledge about the hierarchical task decomposition, as in the basic case the knowledge which is used to design the state representation is sufficient to deploy this algorithm. In particular it can be applied to problems without a clear hierarchy.

4.4.2 A Novel Algorithm

Algorithm 2 summarises our approach, showing the key extensions to the standard version of SARSA(λ) with tile coding (Sutton & Barto 1998). In our case learning at the ground level is the same as in standard SARSA(λ). The modification which is crucial for our discussion is the

point where the SARSA(λ) algorithm is given shaping reward $F(s, s')$ in Line 14 of Algorithm 2 where the temporal difference is computed. The way in which $F(s, s')$ is evaluated defines our extension.

The shaping reward $F(s, s')$ is computed in Line 4 as the difference of the value function of current and previous states visited by the agent. Thus, $\Phi(s) = V(s)$ where V is the current estimate of the value function of the abstract RL task. This task is learned using temporal difference updates with tile coding (Lines 8 and 9) and symbols related to this learning process have subscript v in Algorithm 2. The mapping from state s to the set of tiles used at the abstract level is done in a straightforward way without any special knowledge. Basically, a lower resolution of tiles can be applied. However with optional, additional knowledge about the problem such a mapping can remove some state variables and appropriately focus abstract learning. It means that the less expressive representation can apply not only lower resolution but also remove some of the state variables.

RL at the abstract level is treated as a Semi-MDP[2] since due to coarse tile coding an agent can be several time steps within the same position at the abstract level. The resolution of tile coding at the ground level should avoid such situations. For this reason time t is used when temporal difference in Line 8 is evaluated.

The generic function $reward_v(r)$ shows that abstract learning can receive an internally modified reward. According to our empirical evaluations $\frac{r}{10}$ gives good results on different domains where both the positive and negative reward is given. The division by factor 10 guarantees that the shaping reward extracted from an abstract V-function has smaller impact than the environment reward.

4.4.3 Use of Tilings

The algorithm has been shown as a generic approach to use two levels of tile coding. We combine this algorithm with the idea of mixed resolution function approximation which was introduced in Section 4.3. This leads to two versions of Algorithm 2:

1. ground learning (Q-function) with only high resolution (fine tilings) and abstract learning (V-function) with low resolution (coarse tilings),

2. ground learning with both low and high resolution (according to the description in Section 4.3) and abstract learning with low resolution like in the first version.

4.4.4 Properties of the Algorithm

Even though the shaping reward is learned with a separate tile coding and separate vector of parameters, its performance is strictly correlated with relations between the Q- and V-function,

[2]Semi-MDPs are extensions to MDPs in which the time between one decision and the next decision is taken into consideration as a real-valued or an integer-valued random variable (Hu & Yue 2007).

Algorithm 2 SARSA(λ)-RS: Gradient-descent SARSA(λ) with potential-based reward shaping from temporal difference learning of an abstract level value function.

repeat {for each step of episode}
  $V \leftarrow$ the abstract level v-function for state s
  $V' \leftarrow$ the abstract level v-function for state s'; 0 if s' is a goal state
  $F(s, s') = \gamma_v V' - V$
  $r_v = reward_v(r)$
  if $r_v \neq 0$ or tiles for $s \neq$ tiles for s' at the abstract level **then**
    $t \leftarrow$ the number of time steps since the last update
    $\delta_v = r_v + \gamma_v^t V' - V$
    Update approximation $V(s)$ according to temporal difference δ_v
  end if
  $Q \leftarrow$ the ground level state-action value for pair (s, a)
  $Q' \leftarrow$ the ground level state-action value for pair (s', a')
  if s' is not a goal state **then**
    $\delta = r + F(s, s') + \gamma Q' - Q$
  else
    $\delta = r - Q$
  end if
  Update approximation of $Q(s, a)$ according to temporal difference δ
until s' is terminal

in general, and the design of both levels of tiles. The following factors can thus have influence on the performance of Algorithm 2.

- V(s) values learned at the abstract level are a function of only states whereas ground RL learns Q(s,a) values in order to deal with unknown environment dynamics. This difference suggests that the positive influence of the potential function extracted from V(s) should be higher with a larger number of actions $a \in \mathbb{A}(s)$ because V(s) learns only values of states whereas Q(s,a) additionally distinguishes actions (there are more values to converge). Thus, V(s) can converge faster than Q(s,a) in the initial period of learning and can give positive guidance for learning Q(s,a) at the ground level.

- There can exist structural dependencies between features in the state space. Such structural dependencies can be used to define a reduced representation at an abstract level. For example, a reduced number of features can provide a high level guidance (e.g., goal homing). Detailed encoding at the ground level enables the algorithm to take into account other factors and world properties. Abstract learning with properly selected factors can result in a rapidly converging V-function which may improve slower converging ground learning.

- When the RL agent needs to learn on a problem with a wider range of values of state features with the same required granularity of function approximation (when the value function is very diverse and high granularity is necessary), the impact of learned reward shaping can be more significant. When tile coding at the abstract level applies a lower

resolution, it reflects the situation given in Figure 4.1. Particularly in the initial period of learning, an abstract V-function can faster propagate information about highly rewarded areas than abstract Q-function.

Further sections test some of the aforementioned hypotheses on a range of RL tasks.

4.5 Experimental Design

A number of experiments have been performed to evaluate extensions to RL proposed in Sections 4.3 and 4.4. The following configurations are tested. Their acronyms are defined here for the reference in the remainder of this chapter.

1. SARSA(λ): the standard version of the algorithm (Sutton & Barto 1998).

2. Coarse: the standard version of SARSA(λ) with coarse tile coding.

3. Mixed: the standard version of SARSA(λ) with mixed resolution, that is, two tilings in one function approximator (according to Section 4.3).

4. RS: the algorithm introduced in Section 4.4 with coarse resolution at the abstract level and only fine resolution at the ground level.

5. Mixed-RS: like the previous version but with a mixed resolution for ground learning.

The following values of common RL parameters were used: $\lambda = 0.7$ (used at both levels), and $\lambda = 0$ (also at both levels) in the second series of experiments without eligibility traces, $\gamma = 0.99$, $\gamma_v = 0.99$, $\alpha = 0.1$ and $\alpha_v = 0.1$ (in both abstract and ground learning, the learning rate was being linearly decreased with each episode reaching 0.01 in the last episode). Values $\alpha = 0.1$ and $\lambda = 0.7$ were also used in the famous practical application of temporal difference learning: TD-gammon (Tesauro 1992). In all experiments ϵ-greedy exploration strategy was used with ϵ decreasing linearly from 0.3 in the first episode to 0.01 in the last episode. Values of these parameters were chosen arbitrarily and the selection was guided by the most common settings from the relevant literature (Sutton & Barto 1998; Tesauro 1992). This value of ϵ is high enough to provide explorative behaviour, but small enough to ensure that the policy still drives exploration. All runs on all tasks were repeated 30 times and average results are presented in graphs. Following the evaluation process from recent RL competitions, the accumulated reward over all episodes was used as a measure to compare results in a readable way. It is worth noting that also the asymptotic performance can be explained using this type of graphs. Specifically, when two curves are parallel within a given number of episodes, it means that the asymptotic performance of two corresponding algorithms is the same. If one of these curves is steeper, it means that the asymptotic performance of the corresponding algorithm is better in the period

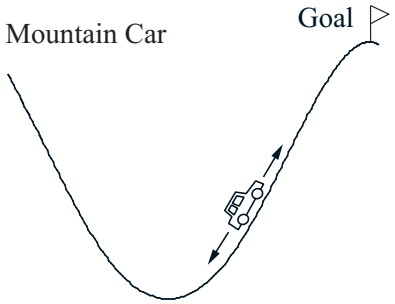


Figure 4.2: The mountain car task (Sutton & Barto 1998).

under consideration. Error bars illustrating the standard error of the mean (SEM) are also presented. Statistical significance was checked with a paired sample Z test by setting the level of significance at $P < 0.05$.

The eligibility traces ($\lambda > 0$) are implemented in an efficient way (Cichosz 1995; Sutton & Barto 1998) at both levels. They are truncated when the eligibility becomes negligible. Specifically, the trace of N most recently visited states or state-action pairs is stored where $(\lambda\gamma)^N \geq 10^{-9}$. The value of eligibility is evaluated as: $e(\phi_i(s,a)) = (\lambda\gamma)^t$ where t is the number of time steps since ϕ_i has been added to the trace. In this way, for all ϕ_i of the most recent pair (s,a), $t = 0$ and it makes $e(\phi_i(s,a)) = 1$ for all ϕ_i of this pair. It means that *replacing eligibility traces* are used (Singh & Sutton 1996). For given values of parameters, $(\lambda\gamma)^N \geq 10^{-9}$, a maximum size of the trace is $N = 56$.

4.6 Experimental Domains

The following set of popular RL tasks were used as test domains in our experiments.

4.6.1 Mountain Car

The first experiments were performed on the mountain car task according to the description of Sutton & Barto (1998). This is one of the most famous RL benchmark problems. The car is situated in a steep-sided valley and its goal is to get out of this valley and ride to one of the hills (see Figure 4.2). Because the car's engine is not powerful enough, the car has to go certain distance up towards the opposite hill to get some momentum, and then accelerate towards the hill which corresponds to its goal.

The state space in this domain is described by the position p_t and velocity v_t of the car. There are three actions: backward, coast and forward. These actions correspond to car's acceleration a_t which has values -1, 0 and 1 correspondingly. The state is updated at each time step according to the following simplified physical model:

$$p_{t+1} = p_t + v_t, \tag{4.6}$$

$$v_{t+1} = v_t + (0.001a_t) + (g\cos(3p_t)), \tag{4.7}$$

where $g = 0.0025$ is gravity. The range of state variables is bounded: $-1.2 \leq p_{t+1} \leq 0.5$ and $-0.07 \leq v_{t+1} \leq 0.07$. The goal state is reached when $p_{t+1} \geq 0.5$ for the main goal on the right hill and $p_{t+1} \leq -1.2$ for the negative goal on the left hill. In both cases, the episode ends and the new episode starts with the agent placed in a random position. An episode was also terminated, and the agent placed in a random position, after 10^3 steps without reaching any of the goal states. In our comparisons all tested algorithms were always evaluated on the same sequence of starting random positions for a fair comparison. It means that the random sequence of starting positions was selected before the experiment and all algorithms were tested on the same set of starting states. The agent received a reward of 1 upon reaching the goal state on the right hill and -1 on the left hill. This type of the reward functions was motivated by experiments of Munos & Moore (2002), as it makes the shape of the V-function more diverse (the car has to learn that it cannot go too much to the left). The goal of learning is to get to the right hill minimising the number of steps. Following Sutton & Barto (1998), 10 tilings with 9×9 tiles were used for fine tilings and 6×6 for coarse tilings.

4.7 Car Parking

The car parking task comes from the existing RL literature (Cichosz 1996, 1995). This is a simulated car parking problem (illustrated in Figure 4.3), where the goal of learning is to navigate the car to the garage so that the car is entirely inside of the garage. The car is represented as a rectangle in Figure 4.3 and cannot move outside of the driving area which is bounded by the solid line. This is an episodic task where the episode ends either when the car is successfully parked in the garage or when the car hits the wall of the bounded area. Each episode starts with the car placed in the same starting location (see below for exact coordinates). A reward of 100 was given upon entering the goal state. At all other time steps, the reward is 0.

The state space in this domain is described by three continuous variables: coordinates of the centre of the car, x_t and y_t, and the angle, θ_t, between the car's axis and the X axis of the coordinate system. There are three actions in the system: drive left, drive straight on, and drive right. These actions correspond to values of -5, 0, and 5 of the turn radius a_r which is used in equations below. These equations specify how state variables are updated after each time step τ.

1. if $r \neq 0$ then

   (a) $\theta_{t+\tau} = \theta_t + \tau v / a_r$

   (b) $x_{t+\tau} = x_t - a_r \sin(\theta_t) + a_r \sin(\theta_{t+\tau})$

   (c) $y_{t+\tau} = y_t + a_r \cos(\theta_t) - a_r \cos(\theta_{t+\tau})$

2. if $r = 0$ then

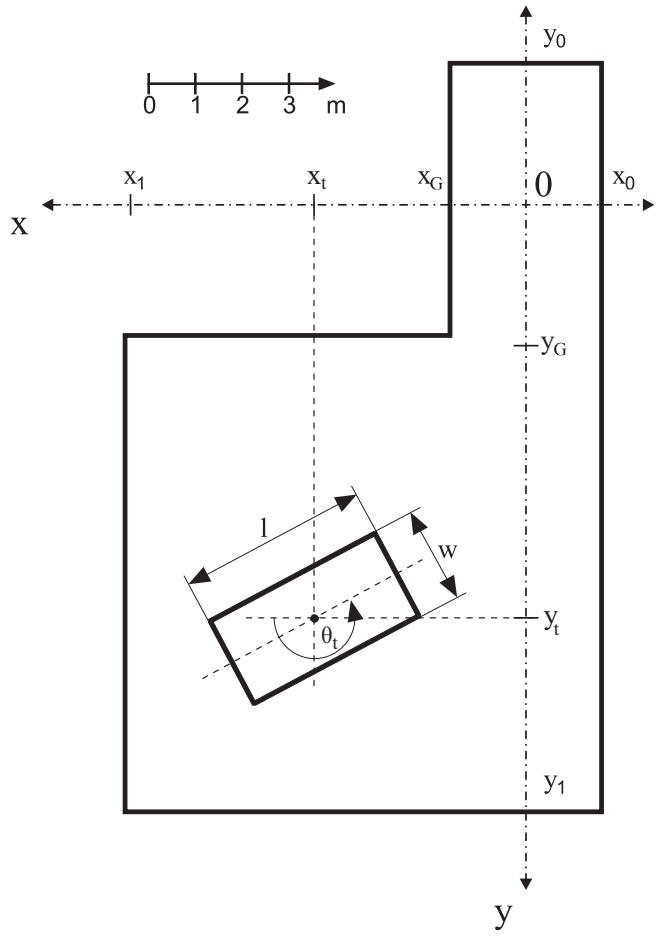


Figure 4.3: The car parking task (Cichosz 1995). The domain state is described by $\langle x_t, y_t, \theta_t \rangle$.

(a) $\theta_{t+\tau} = \theta_t$

(b) $x_{t+\tau} = x_t + \tau v \cos(\theta_t)$

(c) $y_{t+\tau} = y_t + \tau v \sin(\theta_t)$

Velocity v was constant and set to 1 [m/s]. The time step $\tau = 0.5$ [s] was used. The initial location of the car is: $x_t = 6.15$ [m], $y_t = 10.47$ [m], and $\theta_t = 3.7$ [rad].

Two different configurations of the task were analysed in our experiments. The first one is with all geometrical parameters specified by Cichosz (1996). The dimensions are as follows: $w = 2$ [m], $l = 4$ [m], $x_0 = -1.5$ [m], $x_G = 1.5$ [m], $x_1 = 8.5$ [m], $y_0 = -3$ [m], $y_G = 3$ [m], and $y_1 = 13$ [m]. For this configuration, there were 6 tilings over one group of three state variables with 5×5×5 tiles per tiling. The state space is defined in the same way as in (Cichosz 1996). As this version of the problem is relatively small, the same tilings were used also for the V-value at the abstract level. In the second configuration, the size of the driving area was tripled

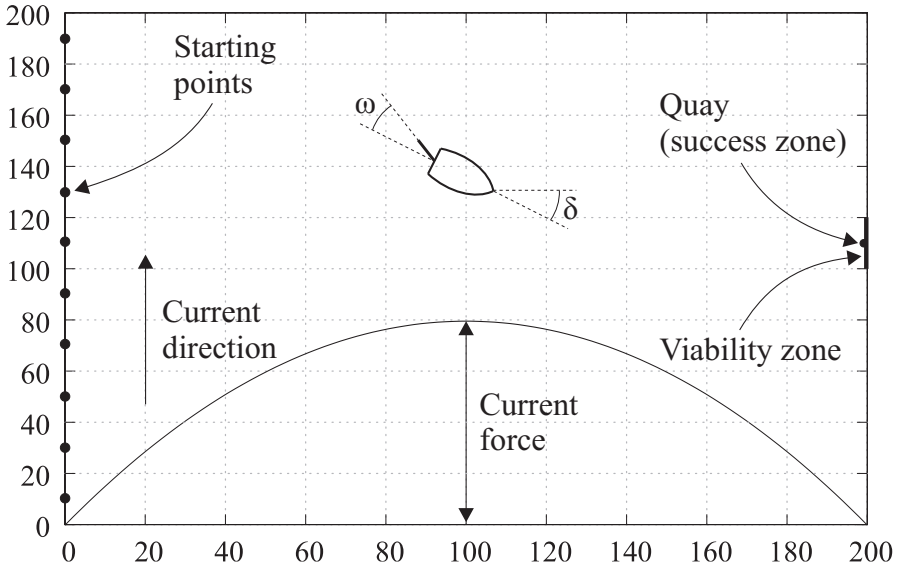


Figure 4.4: The boat task (Jouffe 1998).

with $x_1 = 24.5$ and $y_1 = 37$. Because of the larger size, the number of intervals was also tripled yielding $15 \times 15 \times 15$ tiles per tiling for fine tilings and $10 \times 10 \times 10$ for coarse tilings. The initial location of the car in this larger version of the domain was: $x_t = 22.15$ [m], $y_t = 24.47$ [m], and $\theta_t = 3.7$ [rad].

4.7.1 Boat

The problem is to learn how to navigate a boat from the left bank of the river to the quay on the right bank (see Figure 4.4). There is a strong non-linear current in the river. The boat starts in one of the ten possible starting positions on left bank and navigates to a narrow quay on the right bank (the sequence of random starting positions is the same within compared algorithms for a fair evaluation). The fact that there is strong non-linear current in the river requires precise use of continuous actions in this domain or at least a fine grained discretisation (Lazaric et al. 2007). Our implementation of this domain follows the description of Jouffe (1998) except for a narrower quay with its width set to $Z_s = 0.2$ and the random starting positions used recently by Lazaric et al. (2007) where this task was shown to be challenging for classical RL algorithms. This domain was used in our experiments to check the influence of the number of actions on the performance of our methods, because discretisation into a larger number of actions leads to better final results in this domain but also yields more time consuming learning.

The state of the environment is described by coordinates of the boat's bow, x and y in the range $[0, 200]$, and the angle δ between the boat's axis and the X axis of the coordinate system. The boat is controlled by setting the desired direction which is in the range $[-90°, 90°]$. The

boat's bow coordinates are updated using the following equations:

$$x_{t+1} = \min(200, \max(0, x_t + s_{t+1}\cos(\delta_{t+1})))$$
$$y_{t+1} = \min(200, \max(0, y_t - s_{t-1}\sin(\delta_{t+1}) - E(x_{t+1})))$$

where E stands for the effect of the current and is expressed as: $E(x) = f_c(\frac{x}{50} - (\frac{x}{100})^2)$ where $f_c = 1.25$ is the force of the current. The angle, δ_t, and speed, s_t, are updated according to:

$$\delta_{t+1} = \delta_t + I\Omega_{t+1}$$
$$\Omega_{t+1} = \Omega_t + ((\omega_{t+1} - \Omega_t)(s_{t+1}/s_{MAX}))$$
$$s_{t+1} = s_t + (s_d - s_t)I$$
$$\omega_{t+1} = \min(\max(p(U_{t+1} - \delta_t), -45^\circ), 45^\circ)$$

where $I = 0.1$ is the system inertia, ω the rudder angle, $s_{MAX} = 2.5$ the maximum allowed speed of the boat, $s_d = 1.75$ is the desired speed of the boat, and $p = 0.9$ is the proportional coefficient required to compute the rudder angle according to a given value of the desired direction U_t.

The reward function is defined as follows. If the agent crosses left, top, or bottom boundary of the working area, the reward of -10 is given. If the quay is reached within its boundaries, that is, within the distance $Z_s/2$ from the centre of it (the success zone) where $Z_s = 0.2$, the reward of 10 is always given. There is an additional viability zone defined around the quay. The width Z_v of this zone is 20. If the boat reaches the right bank within this zone (outside the success zone) the reward function is decreasing linearly from 10 to -10 relative to the distance from the success zone. Reaching the right bank outside of the viability zone yields the reward of -10.

4.8 Results

Experimental results are discussed for each domain separately as they were designed to test different properties of the methods proposed in this chapter.

4.8.1 Mountain Car

The obtained results with eligibility traces (Figure 4.5) show that reward shaping with mixed function approximation (Mixed-RS) has the most rapid improvement. Mixed function approximation (Mixed) obtains the second best performance, though the cumulative reward is worse than in Mixed-RS with statistical significance after 2050 episodes. Mixed is better than SARSA(λ) with statistical significance after 230 episodes, than RS after 390 episodes, and better than Coarse after 940 episodes. When comparing other configurations, Coarse speeds up learning at the beginning, but asymptotically loses with a more detailed representation (refer to Section 4.5 to check how to read the asymptotic performance). The need of a more expressive representation becomes evident here. Learning with reward shaping (RS) offers good asymptotic properties,

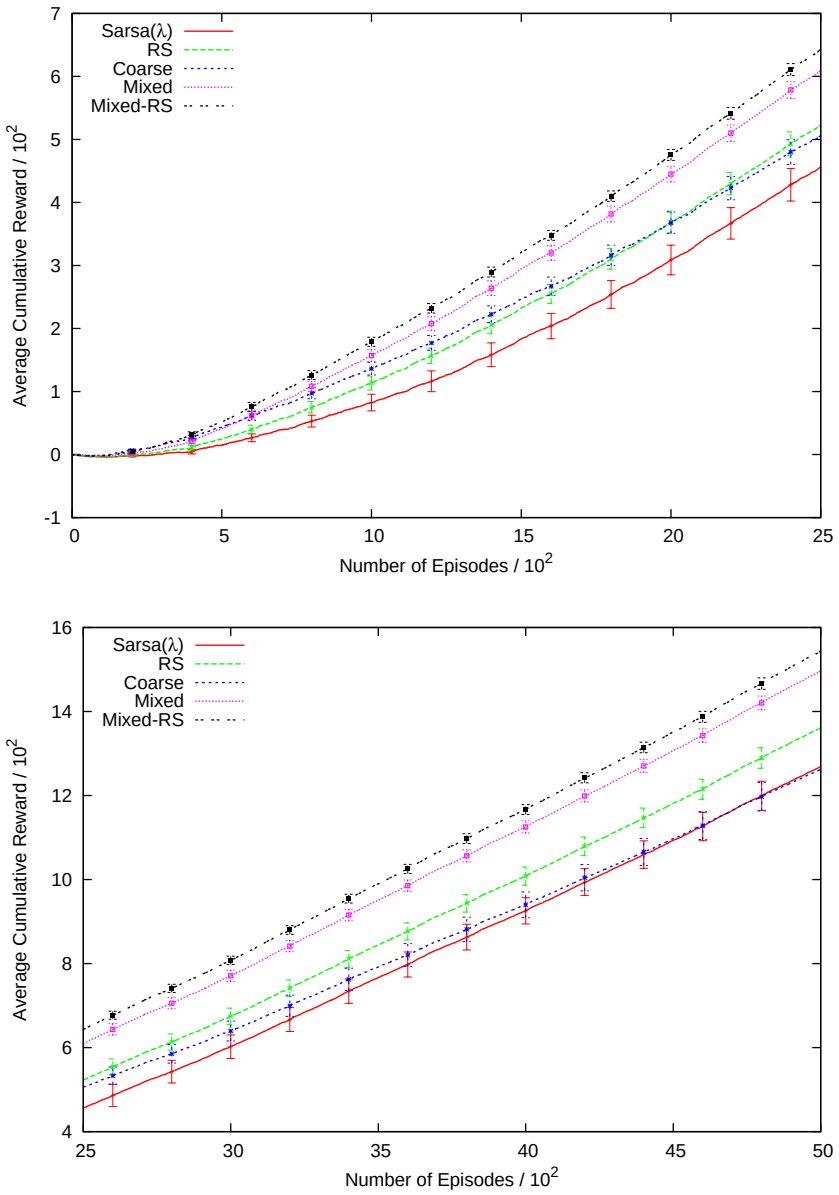


Figure 4.5: Results on the mountain car problem ($\lambda = 0.7$). The top graph shows the first 25×10^2 episodes, and the bottom graph shows the remaining 25×10^2 episodes.

but its improvement is smaller than with mixed versions (Mixed and Mixed-RS). An interesting observation is that mixed representations, both with (Mixed-RS) and without (Mixed) reward shaping, improve learning right from early episodes and gain the best asymptotic performance. These results show, that RL can be boosted in a straightforward way just by combining two representations with different expressiveness in one function approximator (Mixed) and additional use of reward shaping (Mixed-RS) can lead to further improvement.

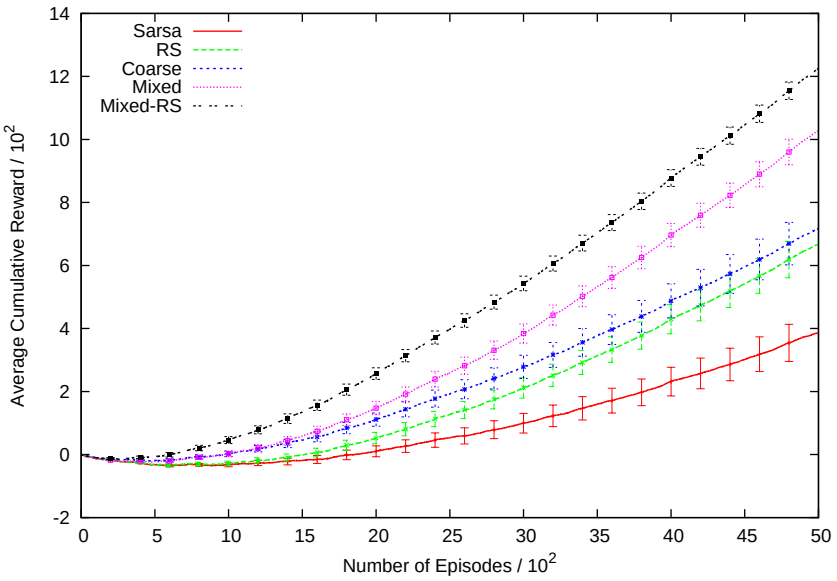


Figure 4.6: Results on the mountain car problem ($\lambda = 0$).

Another experiment on the mountain car task is reported in Figure 4.6 where $\lambda = 0$ is used. In this case the advantage of Mixed and Mixed-RS is more evident. Mixed-RS is better than Mixed with statistical significance after 28 episodes. Mixed is better than Coarse after 2900 episodes and better than RS after 420 episodes. Also in this case, Coarse loses asymptotically with other methods. The overall observation from this experiment is that when learning without eligibility traces, our extensions lead to better absolute improvement.

4.8.2 Car Parking

In the car parking problem with the larger size of the working area, the type of knowledge which is learned from coarse tilings starts playing a more significant role. Figure 4.7 shows results for the original task. The task is relatively small here with a short distance to the goal (see the picture of this configuration in Figure 4.3) and our extensions do not bring improvement in this setting. But, the encouraging observation is that asymptotic convergence is not violated when our methods are used. There is no statistical significance between any two methods in this experiment. In the second configuration, where the distance to the goal is bigger, goal-homing knowledge becomes more important. This is reflected in Figure 4.8. In this case two types of reward shaping yielded the best initial improvement with mixed resolution after them. However, Mixed obtains better final convergence than RS. Statistical tests are more informative here. There is no statistical significance between Mixed and RS, and also between Mixed-RS and Mixed. When comparing Mixed-RS and RS, the difference is statistically significant between episodes 600 and 4800. Mixed-RS is better than SARSA(λ) after 580 episodes and there is no statistical difference between Mixed and SARSA(λ). Even though the differences are less

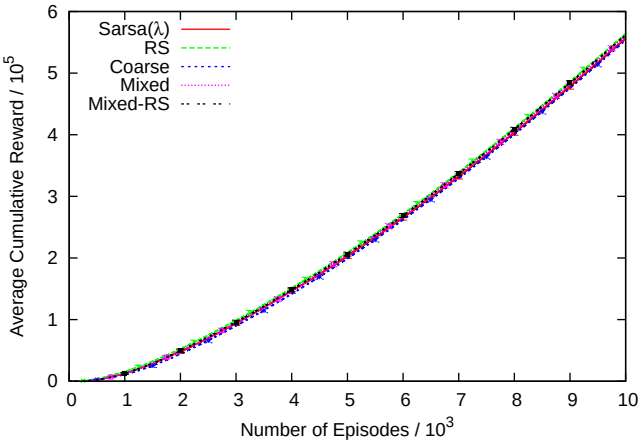


Figure 4.7: The car parking problem with original settings ($\lambda = 0.7$).

significant here, Mixed-RS gains the best performance. It can be noted here that the advantage of our extensions becomes more important on larger instances of problems. We can also try to find an explanation for the fact that reward shaping worked the best here, and RS in the initial period of learning in particular when compared to Mixed. We conjecture that the reason for this is that in order to reach the goal state the car needs to be in a very specific range of positions (it is easy to hit the wall) and learning with only mixed resolution was not able to lead to such an initial improvement because of the strict position to enter the goal. This seems to be a rational explanation when the experiment presented in Figure 4.10 is taken into consideration. In this case, when $\lambda = 0$, methods which use reward shaping, that is, Mixed-RS and RS, work better than all other methods. There is however no statistical difference between Mixed-RS and RS. In Figure 4.9, the experiment on the original task and $\lambda = 0$ is also presented. In contrast to results in Figure 4.7, eligibility traces are not used here and this time the basic version of SARSA(λ) gains better asymptotic performance (no statistical difference between SARSA(λ) and Mixed). This observation shows that on small problems the standard version of the algorithm may be sufficient.

4.8.3 Boat

The agent controls the boat by the desired direction in the range [-90°, 90°]. Experiments with discretization into 5, 20 and 40 values (actions) are reported. The same number of 5 tilings was used with $10\times10\times10$ tiles for fine tilings and $8\times8\times8$ tiles for coarse tilings.

Firstly results with eligibility traces are discussed. Figure 4.11 presents results with 5 actions. Differences, even though small, are statistically significant, particularly for Mixed and Mixed-RS when they are compared to other methods. Mixed-RS has better (with statistical significance)

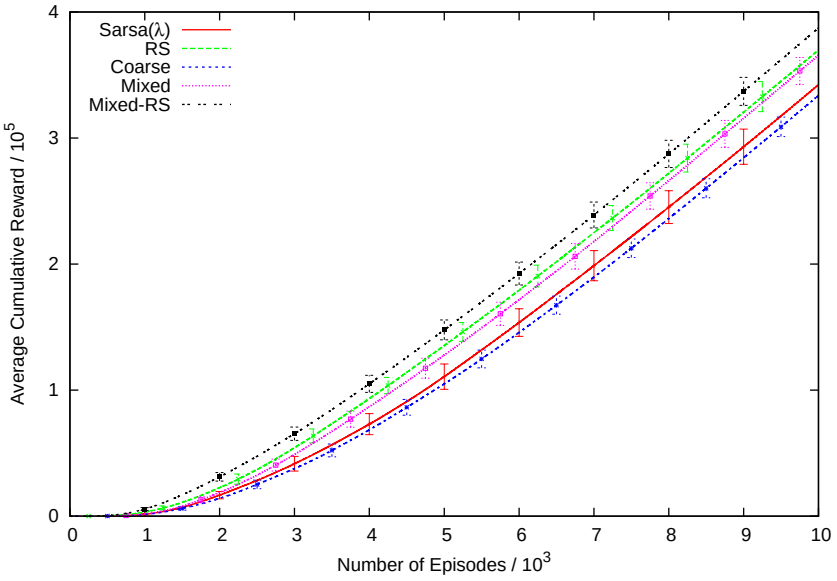


Figure 4.8: The car parking problem with the tripled size of the working area ($\lambda = 0.7$).

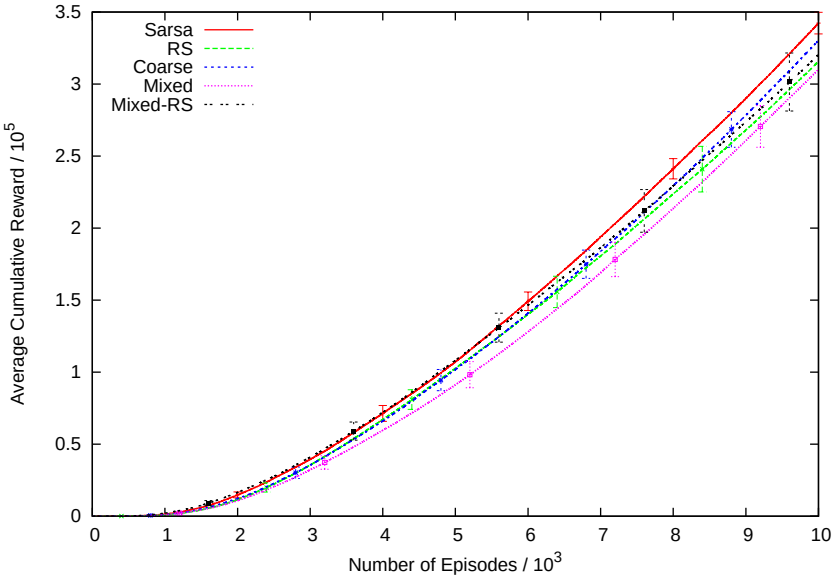


Figure 4.9: The car parking problem with original settings ($\lambda = 0$).

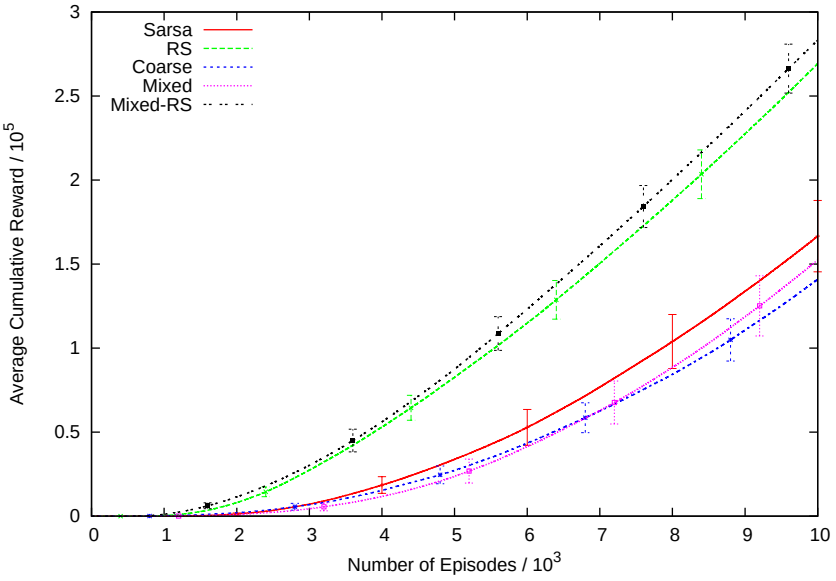


Figure 4.10: The car parking problem with the tripled size of the working area ($\lambda = 0$).

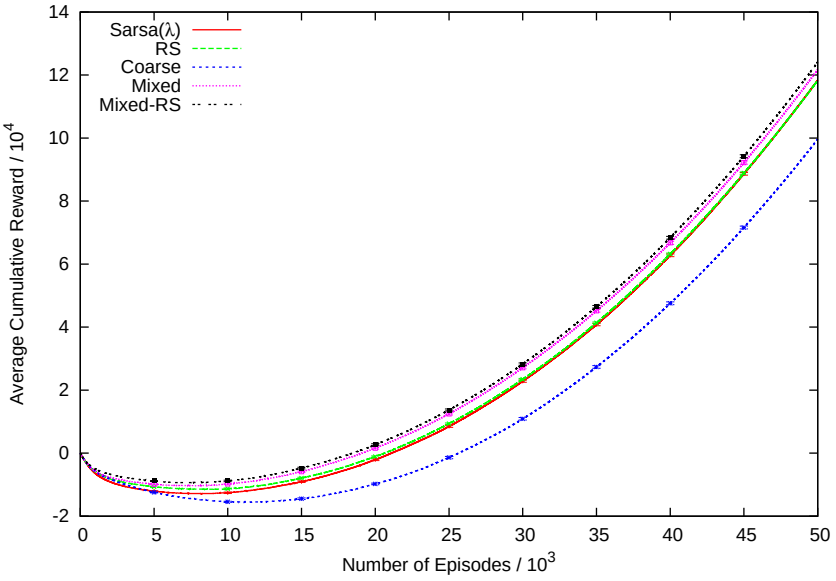


Figure 4.11: The boat problem with 5 actions ($\lambda = 0.7$).

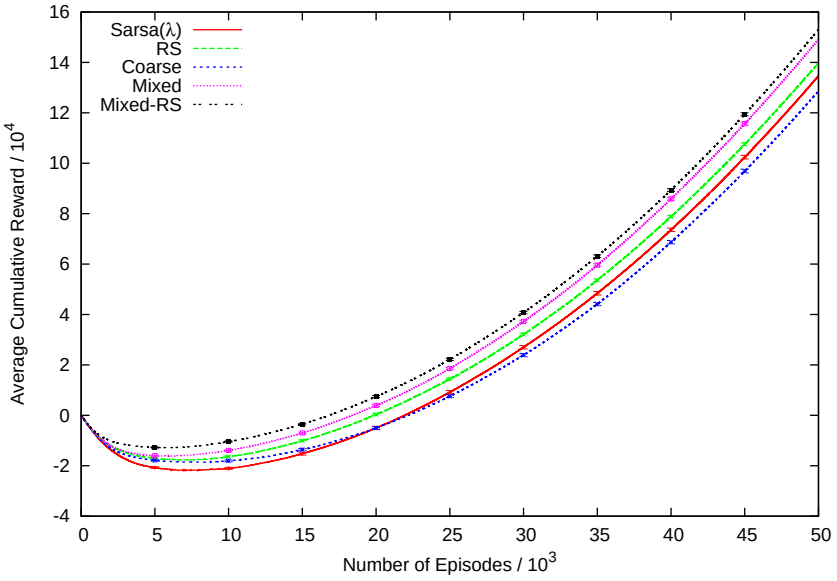


Figure 4.12: The boat problem with 20 actions ($\lambda = 0.7$).

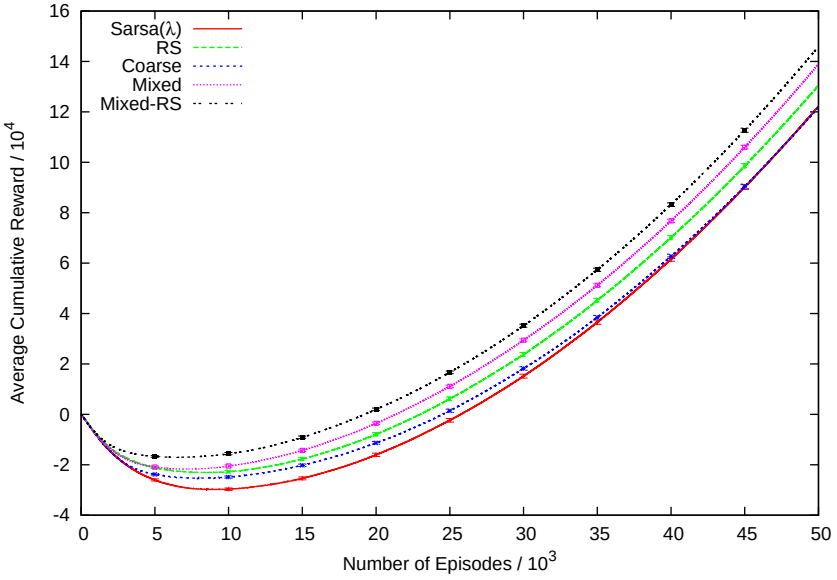


Figure 4.13: The boat problem with 40 actions ($\lambda = 0.7$).

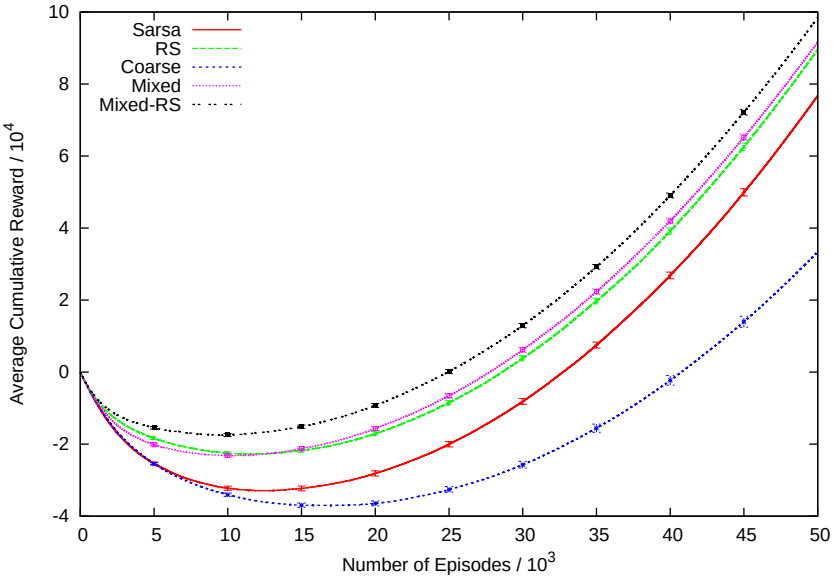


Figure 4.14: The boat problem with 5 actions ($\lambda = 0$).

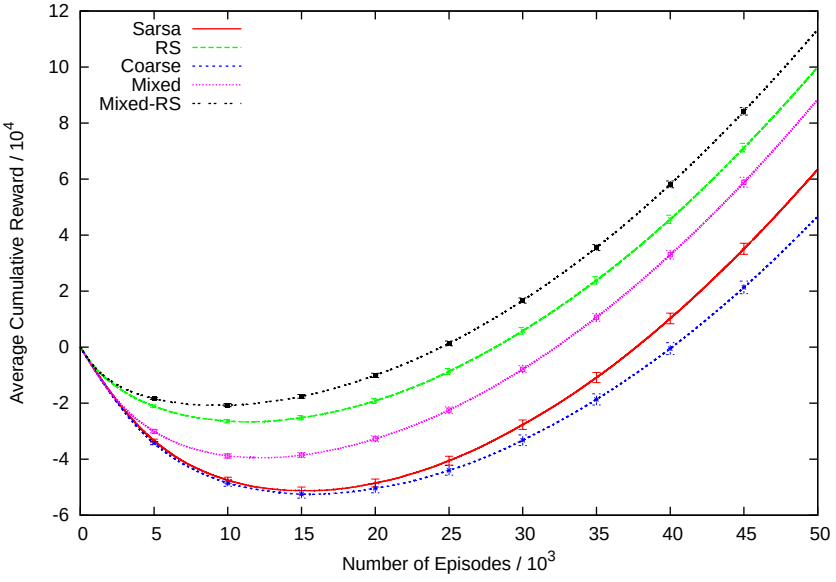


Figure 4.15: The boat problem with 20 actions ($\lambda = 0$).

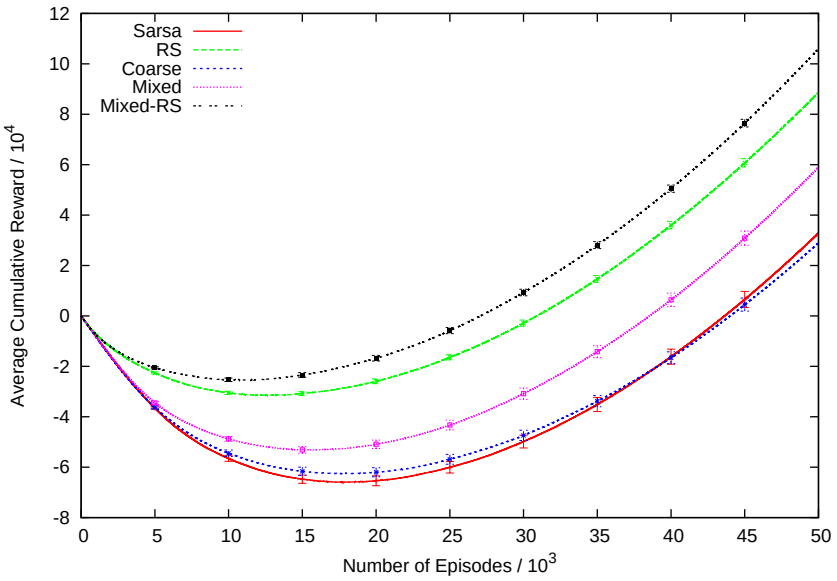


Figure 4.16: The boat problem with 40 actions ($\lambda = 0$).

cumulative reward after 350 episodes than Mixed. And, Mixed is better than RS after 2000 episodes and better than SARSA(λ) after 300 episodes. Learning in this version of the task progresses relatively well and, in effect, the coarse learning loses from early episodes. When 40 actions were used (Figure 4.13), the best performance was also due to reward shaping with mixed function approximation at the ground level (Mixed-RS) followed by learning with only mixed function approximation (Mixed). The difference between Mixed-RS and Mixed is statistically significant after 440 episodes and the absolute improvement is higher here than when 5 actions were used. Mixed is also better than RS after 6800 episodes. Additional experiments with 20 actions (see Figure 4.12) yielded results where reward shaping led to higher improvement than with 5 actions and lower than with 40 actions showing coherence with our hypothesis that our extensions are of particular interest when there are many actions $a \in \mathbb{A}(s)$. The results of RS are between Mixed and pure SARSA(λ) in a similar way as in mountain car. RS with 40 actions converges faster in the initial phase of learning, at a pace similar to SARSA(λ) with only 5 actions, and obtains better results in the long run. The asymptotic performance of our algorithms is also very good. The problem of slow convergence of pure SARSA(λ) with 40 actions (i.e. the number of actions desired for this domain) which was pointed out by Lazaric et al. (2007) can thus be mitigated by our approaches.

The boat domain was also evaluated without eligibility traces, that is, with $\lambda = 0$. Results of this experimentation are in Figures 4.14, 4.15 and 4.16 for 5, 20 and 40 actions respectively. In this case observations are different than in the previous study. Firstly, differences between algo-

rithms are higher in terms of absolute difference in performance, the distances between curves are bigger with a similar size of intervals for the standard error of the mean. In all cases Mixed-RS performs better with statistical significance than other methods. Another important issue in this case is that the basic version of the SARSA(λ) algorithm performed very well in terms of asymptotic convergence. When learning with eligibility traces, the improvement which our methods bring was smaller in terms of the absolute difference, but the asymptotic performance was also very good.

4.9 Summary and Discussion

In this chapter, we propose using two hypotheses spaces, that is, function approximation with different levels of expressiveness in RL. Two approaches to obtain learning with mixed resolution are introduced and empirically evaluated when applied to tile coding. The results show that simultaneous learning at two levels and learning with mixed resolution FA can converge to a stable solution. We conjecture that this is due to the fact that our experiments are based on the SARSA algorithm (on-policy temporal difference learning) which has been shown in the literature (Stone et al. 2005) to work better with function approximation than Q-learning.

Results on tasks selected according to different properties show that the application of our extensions to RL are especially beneficial when: 1) there are many actions in each state; 2) a high resolution of the policy is required (due to details in the environment) with a wide range of values of state variables, i.e. on the larger instance of the domain; 3) a high level guidance can be extracted from a subset of state variables.

Reward shaping with mixed FA at the ground level was the best in all runs on large instances. Actually, only in the car parking task with original size and $\lambda = 0$ our approaches were not the best, and even then it was not statistically significant. Learning with only mixed FA was the second-best on two domains but reward shaping without mixed resolution was better on one domain, that is, when the path to the goal led via states with very constrained values of state variables (entering the parking space in the car parking task). Overall, the results show that reward shaping with mixed resolution FA at the ground level was the most successful.

The contribution of the algorithm is the improved convergence rate, especially in domains satisfying the properties outlined above.

The comparison between learning with $\lambda > 0$ and $\lambda = 0$ showed that our algorithms generally lead to better absolute improvement when $\lambda = 0$, but good asymptotic properties were preserved in both cases in most experiments. Additionally, even with $\lambda = 0$, our algorithms without eligibility traces faster gained a similar performance to SARSA(λ) with eligibility traces, that is, with $\lambda > 0$. Eligibility traces, even when using a more efficient version (truncating is used in our experiments), yield certain computational overhead. With $\lambda = 0$, only one backup is performed after each step and with $\lambda = 0.7$ (and other relevant parameters according to our experimental design) the number of backups is $N = 56$. The computational complexity is significant and was

empirically observed during experimental evaluation. This observation indicates that with our methods applied without eligibility traces, a comparable convergence can be achieved at lower cost, because there is at most one backup of the V-function for each SARSA backup. Eligibility traces require significantly more updates. In contrast to eligibility traces, separate and external representation of knowledge is obtained in our method with reward shaping.

It is important to note that ideas proposed in this chapter do not require any explicit domain knowledge. In its basic form abstract learning can be defined using the same knowledge which is used to design tile coding at the ground level. The most straightforward approach is the use of wider intervals of high level tiles.

CHAPTER 5

Theoretical and Empirical Analysis of Reward Shaping

Reinforcement learning suffers scalability problems due to the state space explosion and the temporal credit assignment problem. Knowledge-based approaches have received a significant attention in the area. Reward shaping is a particular approach to incorporate domain knowledge into reinforcement learning. Theoretical and empirical analysis in this chapter reveals important properties of this principle, especially the influence of the reward type, MDP discount factor, and the way of evaluating the potential function on the performance.

5.1 Introduction

The principal idea to improve the performance of machine learning techniques, in general, is to reduce the hypothesis space (Mitchell 1997). RL is a simulation-based technique where the policy is estimated from samples obtained from the simulated or real environment. The key research challenge in the RL community is how to reduce learning complexity, that is, the number of suboptimal actions in the environment required to estimate the policy. Different directions have been investigated in the area. For example, the representational bias reduces the hypothesis space to the set of solutions which can be learned with the reduced representation (Dietterich 2000). The idea here is to use domain knowledge to impose specific constraints on the policy representation and its expressiveness which are most likely desired in the domain. For example, in classical single-agent RL, deterministic polices are sufficient, but multi-agent (Littman 1994; Boutilier 1999) or constraint (Altman 1995; Dolgov & Durfee 2004; Puterman 1994) domains may require stochastic policies. The procedural bias, on the other hand, focuses the exploration process, i.e. how the agent acts in the world during learning, towards preferred regions of the state

space (Asmuth et al. 2008). It can use explicit knowledge about actions which are likely to be good in a given state (Wiewiora et al. 2003). In this case, the goal is not to impose restrictions on the policy representation, but rather on how the agent explores the environment during learning, it can, e.g. ignore visiting states which are irrelevant in a given RL problem. In both cases, the bias can be in the form of either soft or hard constraints.

In this chapter reward shaping is considered as a way of incorporating the procedural bias into RL algorithms. In standard circumstances RL algorithms learn only from the environment reward which refers only to the last action executed in the world. The idea of reward shaping is to provide an additional external reward which does not change the optimal solution but which guides the agent during learning in a more controlled fashion. Reward shaping constitutes a particular method to incorporate background knowledge into RL. Different types of knowledge obtained in different ways and represented differently can be used with reward shaping (Asmuth et al. 2008; Grześ & Kudenko 2008a). However, the general idea is the same. Reward shaping uses some heuristic assessment of how good or bad particular states in the environment are. Having this in mind, one can see RL with heuristic knowledge given to the agent (e.g., via reward shaping) as *informed reinforcement learning* where the difference between informed RL and uninformed RL is analogous to informed and uninformed search in artificial intelligence (Russell & Norvig 2002). The term *informed reinforcement learning* appeared, for example, in (Croonenborghs et al. 2004) where ideas of model-based RL were investigated in Relational RL (Džeroski et al. 2001).

The underlying mathematical model of RL is the MDP, and one of the elements of the formal definition of MDPs is the discount factor, $0 \leq \gamma \leq 1$, which determines how proximal and distant rewards are weighted against each other. This originally comes from economic models where the same payoff has different utility now than when received in the future (Puterman 1994). The discount factor is thus important and represents a part of the specification of a particular domain.

This chapter conducts theoretical and empirical analysis of potential-based reward shaping. In particular, the influence of different reward models, values of the discount factor, and ways of evaluating the potential function on learning with reward shaping is investigated.

5.2 Reward Shaping

Reward shaping is a promising way of mitigating the negative impact of the temporal credit assignment problem. The idea of modifying the reward has been attempted many times in the past (Gullapalli & Barto 1992; Randløv & Alstrom 1998). However, without theoretically analysed solutions some attempts did not work as expected. A classical example is the RL agent who learns how to ride a bicycle. In this case, wrongly defined reward shaping caused the agent to ride in circles instead of directing it to the goal (Randløv & Alstrom 1998). A significant advancement in formalising reward shaping was the development of the potential-based reward shaping, $F(s, s')$, which is evaluated as the difference of some potential function, Φ, of two consecutive states, i.e.

a source state s and a destination state s' (Ng et al. 1999; Wiewiora 2003):

$$F(s, s') = \gamma\Phi(s') - \Phi(s), \tag{5.1}$$

where γ is a discount factor. Ng et al. (1999) proved that reward shaping defined in this way leaves the optimal behaviour unchanged while the time for attempting suboptimal actions can be reduced. Progress estimators in (Mataric 1994) are very similar to the potential function and represent good early findings about desired properties of reward shaping.

Ng et al. (1999) noted that $\Phi(s) = V(s)$ is a particularly convenient potential function because the value function in the process, M', with such reward shaping is $V_{M'}(s) \equiv 0$ which is a particularly easy V-function to learn. It does not mean however that $\Phi(s) = V(s)$ would instantly yield a solution to the problem. It would indicate only which states are closer to the goal state. Because we are considering a classical RL scenario, the transition probabilities are not known. It means that either transition probabilities have to be learned in order to act according to V(s) or Q-values, which directly indicate which action in a given state should be chosen, learned as well. Thus, in the considered model-free framework the latter approach is applied, and all that would remain to learn, when $\Phi(s) = V(s)$, would be to estimate non-zero Q-values since the model of the world is not available. Thus, learning from simulation is still required. The potential function which satisfies $\Phi(s) = V(s)$ is named a *v-equivalent potential function* in the remainder of the thesis.

The potential function is inherently a heuristic function. Without any loss of generality we focus in this chapter on the potential function estimated as the straight line distance, $d(s)$, from a given state, s, to the goal state. This kind of heuristic function is also common in informed search (Russell & Norvig 2002). The potential function should be higher for states which are closer to the goal according to the heuristic. Thus, one can define the non-decreasing potential function as either a positive, Φ^+, or negative, Φ^- potential function, where $\Phi^+(s) = [\max_{s' \in S} d(s')] - d(s)$ and $\Phi^-(s) = -d(s)$. The way the potential function is evaluated represents one of the dimensions of our analysis.

5.3 Reward and the Discount Factor

Two particular elements of the formal description of MDPs are treated as two additional dimensions of our analysis. The reward model and the discount factor are the intrinsic elements of both the formal definition of MDPs and the specification of the problem being modelled. The reward function, $R(s, a, s')$, is the immediate reward received when action a, taken in state s, results in a transition to state s'. The reward function can have different character. Some reward models can be very sparse (e.g., in games very often the reward is given only at the end of the game with +1 for winning and -1 for losing the game Tesauro 1994) or very dense when the non-zero reward is given for each action. Without loss of generality, in this chapter we consider

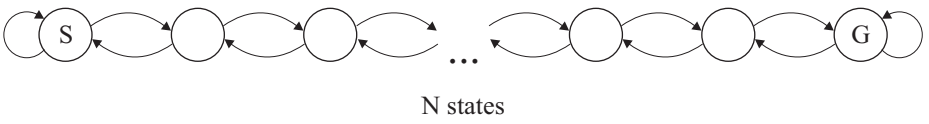


Figure 5.1: The random walk domain.

two general types of the reward function which allow extending our findings to different specific characteristics of the reward model. In the first instance, we assume a sparse reward where the positive reward, $R_g = 1$, is given only upon entering the goal state. The second case, deals with non-positive step reward, $R_s = -1$, which has a meaning of the action cost. The discount factor, γ, is inextricably associated with the reward model. For example, in episodic tasks with R_g the discount factor, γ, should satisfy $\gamma < 1$ because otherwise all state-action values converge to the same $Q(s, a) = R_g$. This would be a useless policy for navigation problems. In real life, it could be compared to a situation when whatever one does, one can reach the final reward which does not depend on the number of steps taken to reach it (in this explanation the fact that from some areas of the state space the goal state may be unreachable is ignored).

5.4 Running Examples and Algorithms

In this section domains and algorithms which are used in the experimental part of our analysis are described. In order to have a controlled impact of domain properties on tested algorithms, three artificial tasks are evaluated.

5.4.1 Random Walk

The first domain is the random walk (RW) task which is worth considering because the straight line distance to the goal is very close to the v-equivalent potential function in this process. Equation $\Phi(s) = V(s)$ is satisfied in RW when actions are deterministic. There are N states in this domain which are connected in a chain-like structure (see Figure 5.1). The agent starts in the left most position (state S) and has to reach the right most position (state G). There are two stochastic actions, left and right, which can fail with probability 0.2 in which case the effect of the opposite action is applied. In this domain, the impact of the path length, N, can be easily analysed without interference of other factors. Furthermore, admissible heuristic functions have different quality between tested domains which yields an additional dimension of our analysis.

5.4.2 Maze

The next domain is the navigation maze task that is shown in Figure 5.2. This is a scaled up (from 15×15 to 25×25 states) version of the domain from (Asmuth et al. 2008), where reward shaping for Rmax was proposed. It is a stochastic domain for which relatively accurate heuristics for potential-based reward shaping can be manually designed. Each action can result in its expected outcome with probability 0.8, and slip into one of two perpendicular directions with probability

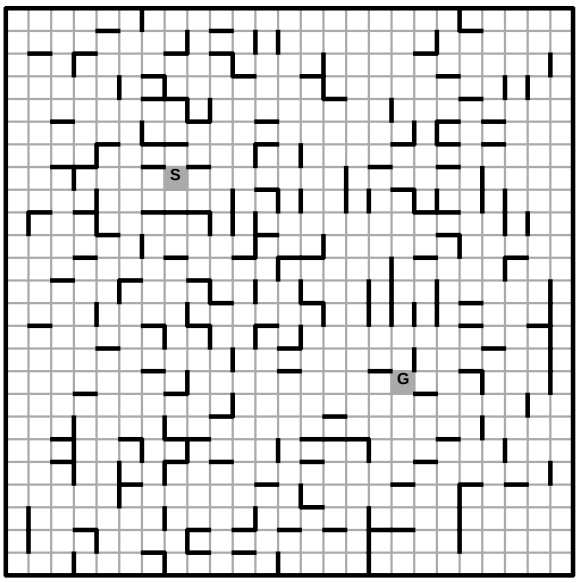


Figure 5.2: The stochastic navigation maze domain (Maze).

0.1 for each of these directions. The start state is marked with letter S. Blocked transitions (walls) between states are marked as solid lines between corresponding states. The RL agent has to learn the highest reward path from the start state S to the goal state G without knowing in advance transition probabilities of the environment. In this domain, which we name Maze in the remainder of this thesis, the heuristic which we are using is of relatively good quality. It is leading the agent towards the goal, however it does not take obstacles into account.

5.4.3 S-maze

The last domain is a maze which has been commonly used in the RL literature (Sutton & Barto 1998: Figure 9.5) and is named S-maze in this thesis. In our case, a larger, scaled up version is used (see Figure 5.3). Each of the 54 states from the base configuration is uniformly divided into 64 squares yielding 3456 states in our scaled-up version. There are 8 actions which lead to an adjacent cell if it is neither the border nor an obstacle. Outcomes which face the border line or an obstacle do not have any effect. Actions are stochastic. With probability 0.2 an action can fail in which case one of the remaining outcomes is chosen with a uniform probability. The straight line distance heuristic is the most inaccurate (when compared to the two previous domains) in this case since backtracking with a sequence of steps is required. By backtracking we mean the sequence of steps which the agent has to perform in the different direction than the one suggested by the heuristic information. For example, the heuristic function may encourage the agent to enter a dead end. In order to get out of it, the agent has to ignore heuristic information and take at least several steps in directions which are not rewarded by the heuristic function.

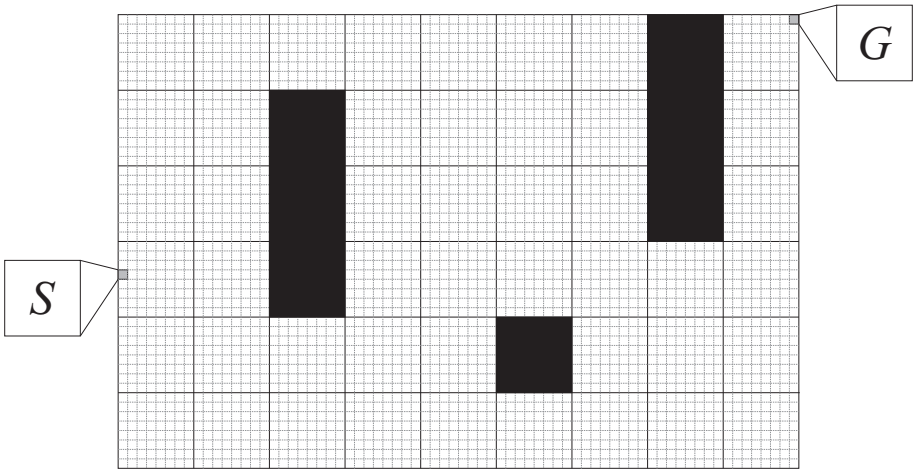


Figure 5.3: The stochastic navigation maze task - S-maze (Sutton & Barto 1998: Figure 9.5).

In all domains, the SARSA algorithm is used as the experimental framework (Sutton & Barto 1998). The ϵ-greedy exploration is used with $\epsilon = 0.3$ in the first episode and decreasing linearly to 0.01 in the last episode. The learning rate, α, starts with 0.1 in the first episode and is decreased linearly to 0.01 in the last episode. Other parameters are given with the description of a particular analysis. Unless explicitly specified, the Q-table is initialised to the value of 0. Values of these parameters were chosen arbitrarily and the selection was guided by the most common settings from the relevant literature (Sutton & Barto 1998).

5.5 Positive and Negative Potential Functions and $\gamma = 1$

The work in this chapter is divided according to the range of values which the γ parameter can take. In the first instance, $\gamma = 1$ is investigated in this section separately, because when $\gamma = 1$ Equation 5.1 takes a special form where the γ parameter is reduced. When $\gamma < 1$ there is no such reduction and later analysis investigates how this fact influences the actual shaping reward which is given to the agent.

The goal of this section is to check the influence of the negative and positive potential functions when $\gamma = 1$ and also to analyse whether modified impact of reward shaping (i.e. $F(s, s') = \tau(\gamma\Phi(s') - \Phi(s))$ where $\tau > 0$) can influence the speed of learning. Results from these experiments serve as an introduction to the theoretical and empirical analysis of Section 5.6. Because $\gamma = 1$ here, only the step reward, R_s, can be used.

The first analysis compares learning rates of the SARSA algorithm with the positive and negative potential functions. Results on three tested domains showed that negative and positive potential functions have exactly the same performance, which is also better than SARSA alone. Results for the RW-64 and S-maze tasks are reported in Figures 5.4 and 5.5 (graphs in this section show the cumulative reward of the agent as a function of the episode number).

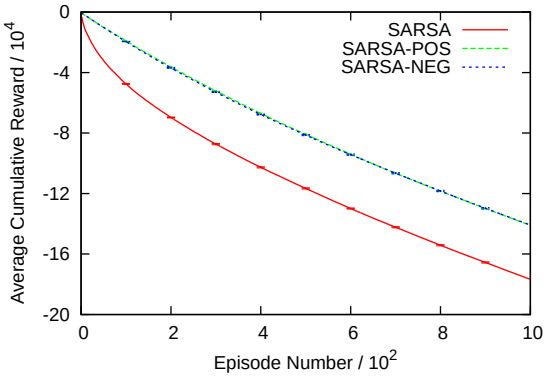


Figure 5.4: Results on RW-64 with positive and negative potential functions.

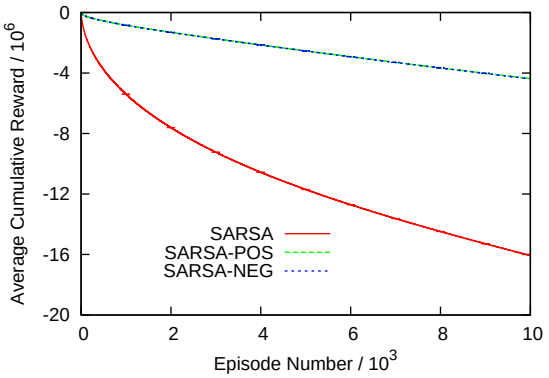


Figure 5.5: Results on S-maze with positive and negative potential functions.

The next question we are asking is whether we can perform better assuming the same potential function as the one used in experiments reported in Figures 5.4 and 5.5 which is very close to the v-equivalent potential function in RW and worse in the two other domains. In this experiment the shaping reward is scaled with the multiplicative factor, τ, in the range of $0.1 - 10^4$. Since two types of potential function work the same in this configuration, for each domain one type of potential function was tested. The value of τ is reported in graphs with results ($\tau = 1$ reflects the standard not scaled shaping reward). Figure 5.6 shows results on RW-64. It can be observed that the relative reduction of the shaping reward decreases performance. Scaling up led to improvement with reference to not scaled shaping. The algorithm reaches saturation point where further increasing of τ did not bring further improvement. The performance was also not decreased by high values of τ. Figure 5.7 shows scaling results on the Maze domain. For lower values of τ, results show the same pattern as in RW-64. The performance with $\tau < 1$ is lower than with

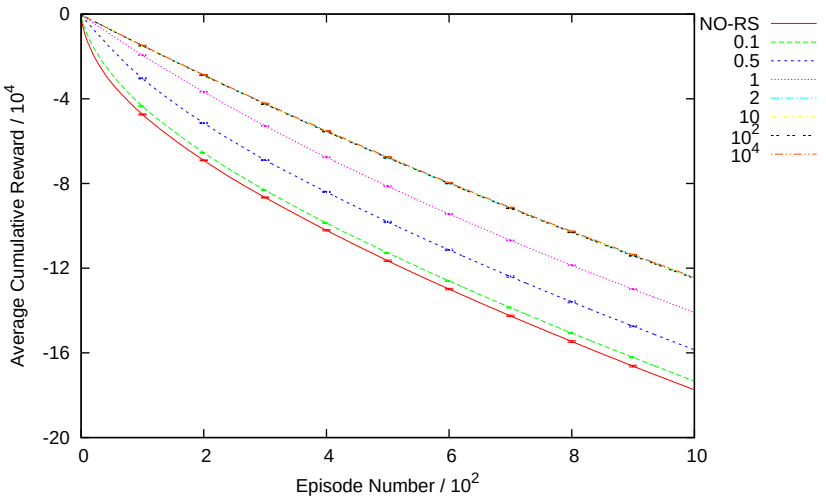


Figure 5.6: Results on RW-64 with a negative potential function and scaling of the shaping reward.

neutral $\tau = 1$. Higher values of τ show improvement, however here the value of $\tau = 2$ is the best whereas all higher tested values reduce performance. This fact can be explained by the quality of the potential function. In RW-64, this function is very close to the v-equivalent function therefore even very high values of τ did not hurt the performance. The potential function is less accurate in the Maze domain than in RW because the heuristic function does not take obstacles into account, thus for very high scaling it leads to lower results. The S-maze heuristic is more inaccurate and high values of τ lead to even worse consequences (see Figure 5.8). Results for $\tau = 50$ and 100 indicate that initial episodes were very long. Furthermore, with $\tau = 10^4$ the agent was not able to reach the goal in a reasonable time. This situation is caused by the faulty heuristic function which leads to dead ends and long backtracking sequences are required to change the direction of search. With very high values of the scaling factor, $\tau > 50$, the influence of the shaping reward becomes very strong. The shaping reward overshadows the reward received from the environment. Overall, the theory of potential-based reward shaping (Ng et al. 1999; Wiewiora 2003) indicates convenient properties of the v-equivalent potential function. However, the RL agent faces also the problem of exploration, and the higher values of the shaping reward (scaled up with $\tau > 1$) have positive influence on exploration as it was reported in experiments discussed in this paragraph. It should be noted that in all three investigated domains the heuristics, even though they may be faulty, still contain useful information. In the case of completely misleading heuristics (e.g., a heuristic which always prefers the longest path in the shortest path problem), any reward shaping would decrease performance.

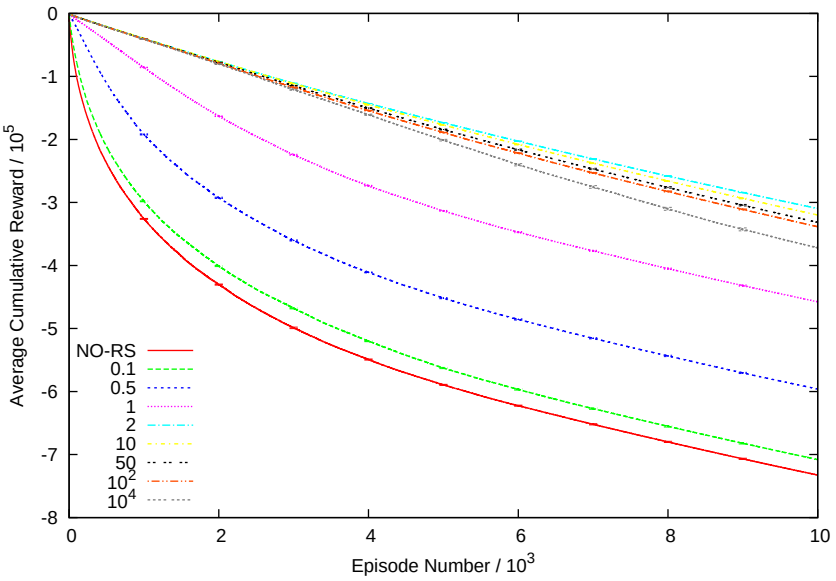


Figure 5.7: Results on Maze with a positive potential function and scaling of the shaping reward.

5.6 Positive and Negative Potential Functions and $\gamma < 1$

This section shows what kind of problems can be encountered when learning with positive and negative potential functions in environments in which $\gamma < 1$.

5.6.1 The Potential Function, Discount Factor, and the Actual Shaping Reward

In this section a detailed analysis is conducted to investigate how a different notion of the potential function (i.e. whether it is positive or negative) influences the actual shaping reward which is given to the agent. Specifically, this analysis is conducted with respect to three types of state transitions in MDPs. When assuming that states s and s' are two states in the environment for which $\Phi(s) < \Phi(s')$, the agent should be rewarded (not penalised) by the shaping reward for transition $s \rightarrow s'$ and not rewarded (penalised) by the shaping reward for $s' \rightarrow s$. Additionally, transitions $s \rightarrow s$ should not be rewarded. Thus, the following cases will be investigated:

$$F(s, s') = \gamma\Phi(s') - \Phi(s) \geq 0, \tag{5.2}$$

$$F(s', s) = \gamma\Phi(s) - \Phi(s') \leq 0, \tag{5.3}$$

$$F(s, s) = \gamma\Phi(s) - \Phi(s) \leq 0. \tag{5.4}$$

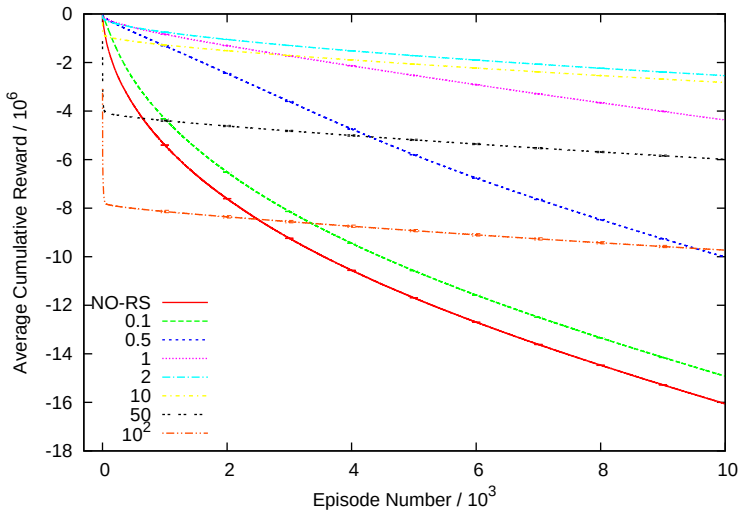


Figure 5.8: Results on S-maze with a positive potential function and scaling of the shaping reward.

Without loss of generality[1] it is enough to assume that the potential function is a linear function with discrete values in $\mathbb{Z}$ and $|\Delta\Phi| = 1$ for any pair of adjacent states. This type of potential function is named an *additive potential function* in our further discussion. The straight line distance to the goal, $d(s)$, can be considered as an additive potential function when $\Phi^{-}(s) = -\lfloor d(s) \rfloor$ for the negative potential function and the positive potential function evaluated in the analogous way. The second type of potential function considered is based on the multiplicative discounting, where $\Phi(s) = \kappa\Phi(s')$ and $\kappa \leq 1$. This case is named a *multiplicative potential function*.

5.6.1.1 Positive Potential Function

In the first instance the additive potential function and its impact on cases shown in Equations 5.2, 5.3 and 5.4 is investigated. In this case, if n represents the potential function of state s, the potential function for state s' is $n + 1$. Thus, for the additive positive potential function the following quantities represent three types of transitions which we consider in our analysis: $F(s, s') = \gamma(n + 1) - n$, $F(s', s) = \gamma n - (n + 1)$ and $F(s, s) = \gamma n - n$, where $n \in \mathbb{N}$. From Equations 5.2, 5.3 and 5.4 and simple algebraic transformations we obtain accordingly:

$$\gamma \geq \frac{n}{n+1}, n \leq \frac{\gamma}{1-\gamma}, \tag{5.5}$$

$$\gamma \leq \frac{n+1}{n}, n \geq \frac{1}{\gamma-1}, \tag{5.6}$$

[1] Our analysis can be naturally extended to the full continuous case.

$$\gamma \leq 1, n \geq 0. \tag{5.7}$$

When the additive positive potential function is used, it is enough to assume with no loss of generality that the minimum value of n is 0 for the most distant state from the goal and n obtains the maximal value in the goal state. In this case, from Equation 5.5 it can be read, that transitions $s \rightarrow s'$ which happen close to the goal state will be negatively rewarded when $n > \gamma/(1-\gamma)$ (n increases when moving towards the goal). For lower values of n, the positive reward will be given as required by Equation 5.2. This relationship shows that in the case of long trajectories (high n) the value of γ should be correspondingly high. If, for example, the maximum value of $n = 1000$, then $\gamma \geq 0.999$. And analogously, for example, for $\gamma = 0.9$ the maximum value of n, which implies the maximum length of the trajectory, is $n \leq 9$. If these conditions are violated, the negative shaping reward will be given for those transitions ($s \rightarrow s'$ in this case) which should be positively rewarded according the the potential function Φ. Transitions $s' \rightarrow s$ and $s \rightarrow s$ do not impose any constraints on n and γ as shown in Equations 5.6 and 5.7 accordingly. Therefore transitions $s' \rightarrow s$ and $s \rightarrow s$ are never positively rewarded. They are always penalised regardless of the value of n and γ as required by Equations 5.3 and 5.4.

Next, the impact of the multiplicative positive potential function on cases shown in Equations 5.2, 5.3 and 5.4 is investigated. If in this case $n(s) = \lfloor d(s) \rfloor$, n for short, then $\Phi(s) = \kappa^{n+1}$ and $\Phi(s') = \kappa^n$. For the multiplicative potential function defined in this way, simple algebraic operations lead to the following constraints:

$$\gamma \geq \kappa, \tag{5.8}$$

$$\gamma \leq \frac{1}{\kappa}, \tag{5.9}$$

$$\gamma \leq 1. \tag{5.10}$$

Furthermore, obtained Equations 5.8, 5.9, and 5.10 lead to the final constraint: $\gamma \geq \kappa$, which is the setting when all three conditions defined by Equations 5.2, 5.3 and 5.4 are met.

5.6.1.2 Negative Potential Function

Firstly, the additive negative potential function and its impact on cases shown in Equations 5.2, 5.3 and 5.4 is investigated. In this case, if $-(n+1)$ represents the potential function of state s, the potential function for state s' is $-n$. Thus, for the additive negative potential function the following quantities represent three types of transitions which are considered in our analysis: $F(s, s') = \gamma(-n) + (n+1)$, $F(s', s) = \gamma(-n-1) + n$ and $F(s, s) = \gamma(-n) + n$, where $n \in \mathbb{N}$. From Equations 5.2, 5.3 and 5.4 and simple algebraic transformations we obtain accordingly:

$$\gamma \leq \frac{n+1}{n}, n \geq \frac{1}{\gamma - 1}, \tag{5.11}$$

$$\gamma \geq \frac{n}{n+1}, n \leq \frac{\gamma}{1-\gamma}, \tag{5.12}$$

$$\gamma \geq 1. \tag{5.13}$$

Without loss of generality it is enough to assume that the maximum value of $-n$ is 0 for the goal state and $-n$ obtains the minimal value for the most distant state from the goal state when the additive negative potential function is considered. In this case, from Equation 5.11 it can be read, that transitions $s \rightarrow s'$ are always positively rewarded as it is required. Here, problems arise with conditions expressed by Equation 5.12. In this case, when moving further from the goal state (i.e. when n grows), transitions $s' \rightarrow s$ start to be positively rewarded whereas they are required to be always non-positively rewarded. For lower values of n, that is those close to the goal state, the negative reward will be appropriately given. But, when moving away from the goal state, those transitions start to be positively rewarded. This relationship and specifically Equation 5.12 show that in the case of long trajectories (high n) the value of γ should be correspondingly high. These conditions mirror what has been found for the additive positive potential function in the previous subsection. Here, $s' \rightarrow s$ start to be positively rewarded when far from the goal state (high n), and in the previous case $s \rightarrow s'$ receive a negative reward when the trajectory is long (high n) and close to the goal state. This section investigates the case when the MDP discount factor is $\gamma < 1$. Since Equation 5.13 requires $\gamma \geq 1$ it means that Equation 5.4 cannot be satisfied and transitions $s \rightarrow s$ are always positively rewarded (see Table 5.1 for the summary).

Next, the impact of the multiplicative negative potential function is investigated. In this case also $n(s) = \lfloor d(s) \rfloor$, n for short. Because the potential function has to be negative here, κ can be raised to odd powers only. Thus, $\Phi(s) = (-\kappa)^{2n+1}$ and $\Phi(s') = (-\kappa)^{2n+3}$. For the multiplicative potential function defined in this way, simple algebraic operations lead to the following constraints:

$$\gamma \leq \frac{1}{\kappa^2}, \tag{5.14}$$

$$\gamma \geq \kappa^2, \tag{5.15}$$

$$\gamma \geq 1. \tag{5.16}$$

Furthermore, obtained Equations 5.8, 5.9, and 5.10 lead to the final constraint: $\gamma = 1$ and $\gamma \geq \kappa^2$, which is the setting when all three conditions defined by Equations 5.2, 5.3 and 5.4 are met. With other values, certain conditions can be violated as indicated by Equations 5.8, 5.9, and 5.10. Here, as in the case of the additive potential function in the previous paragraph, Equation 5.4 cannot be satisfied and transitions $s \rightarrow s$ are always positively rewarded (see Table 5.2 for the summary).

5.6.1.3 Positive and Negative Potential Functions

The theoretical analysis presented in two previous subsections is summarised in Table 5.1 for the additive potential function and Table 5.2 for the multiplicative potential function. The most

		Actual reward	
Transition	Expected reward	Positive potential	Negative potential
$s \to s'$	$F(s, s') \geq 0$	$F(s, s') < 0$ close to the goal	always $F(s, s') \geq 0$
$s' \to s$	$F(s', s) \leq 0$	always $F(s', s) \leq 0$	far from the goal $F(s', s) > 0$
$s \to s$	$F(s, s) \leq 0$	always $F(s, s) \leq 0$	$F(s, s) > 0$ when $\gamma < 1$

Table 5.1: The influence of the type of the additive potential function and of the discount factor, γ, on the actual shaping reward when conditions are violated.

		Actual reward	
Transition	Expected reward	Positive potential	Negative potential
$s \to s'$	$F(s, s') \geq 0$	$F(s, s') < 0$ when $\gamma < \kappa$	always $F(s, s') \geq 0$
$s' \to s$	$F(s', s) \leq 0$	always $F(s', s) \leq 0$	$F(s', s) > 0$ when $\gamma < \kappa^2$
$s \to s$	$F(s, s) \leq 0$	always $F(s, s) \leq 0$	$F(s, s) > 0$ when $\gamma < 1$

Table 5.2: The influence of the type of the multiplicative potential function and of the discount factor, γ, on the actual shaping reward when conditions are violated.

important outcomes of these results can be summarised as follows:

- The additive positive potential function poses problems for transitions $s \to s'$ when close to the goal state (high values of n).
- With the additive negative potential function, transitions $s \to s'$ are always properly rewarded, whereas $s' \to s$ may be positively rewarded when far from the goal state. Additionally, the shaping reward for $s \to s$ is always positive when $\gamma < 1$.
- The multiplicative positive potential function leads to negative reward for transitions $s \to s'$ when $\gamma < \kappa$.
- Transitions $s \to s'$ always receive positive reward as required when the multiplicative negative potential function is used. Transition $s' \to s$ yields incorrect reward when $\gamma < \kappa^2$, and the conditions of non-positive reward are violated for transition $s \to s$ when $\gamma < 1$.

5.6.2 An Empirical Comparison of Positive and Negative Potential Functions

The previous subsection demonstrates analytical results on the properties of reward shaping when $\gamma < 1$ and the potential function can be both positive and negative. Now, an empirical analysis is presented to verify the theoretical findings and further investigate the problem. Because of the discounting in experiments in this section, results in graphs contain the number of episode steps

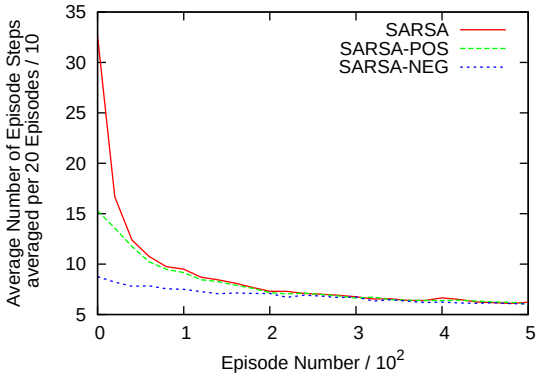


Figure 5.9: Results on RW-32 with $\gamma = 0.95$, R_s, and positive and negative potential functions.

as a function of the episode number. This presentation yielded the most legible charts in this configuration.

5.6.2.1 Evaluation with the Step Reward R_s

In this subsection, the step reward, R_s is used, that is, the negative reward (-1 in our case) is given for each action execution.

In the first instance, the random walk domain (RW) was tested with $\gamma = 0.95$ and with a different number of domain states, N, in the range of $2^3 - 2^7$. Two example runs are reported in Figures 5.9 and 5.10. In the first case, in Figure 5.9 the positive potential function performs worse than negative. On RW-8 both potential functions obtain similar speedup and the growing length of RW showed a decrease in the performance of the positive potential function (Table 5.1 shows that good transitions start to be negatively rewarded when N grows). Thus, with growing N the score of the positive potential function becomes closer to the no shaping baseline. However, the performance of the negative potential function does not remain superior. The experiment with $N = 40$ (see Figure 5.10) captures the situation when learning with the negative potential function (though very good initially) starts going into long trajectories after around 200 episodes. In this run, it was still able to reach the goal state even though trajectories are already significantly longer. For higher values of the RW length, N, the negative potential function did not converge at all (unfinished and very long trajectories with millions of steps).

Two additional questions can, therefore, be asked in this situation to further analyse the problem: 1) why the positive potential function is weaker than negative, and 2) why the negative potential function does not converge on longer RWs ($N > 40$).

The first question is explained by Table 5.1, but it may not be easy to observe this fact there. For this reason Table 5.3 shows shaping rewards for both positive and negative potential functions. When moving towards the goal state (the bottom row in Table 5.3), the shaping reward for

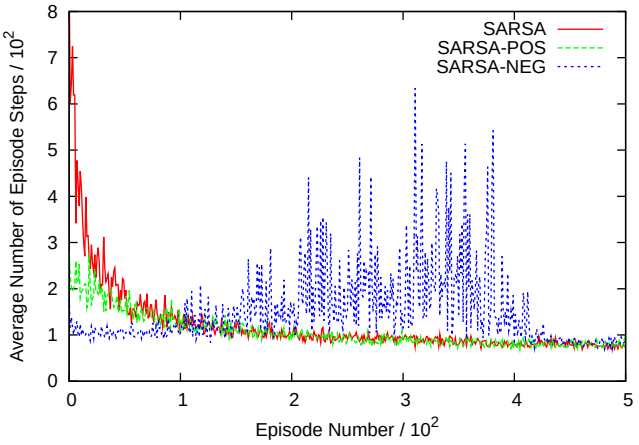


Figure 5.10: Results on RW-40 with $\gamma = 0.95$, R_s, and positive and negative potential functions.

$s \rightarrow s'$ is decreasing when the potential function is positive and its absolute value is additionally lower than in the case of the negative potential function. It means that the shaping reward resulting from the positive potential function is lower. Results were improved when the shaping reward in this case was scaled with $\tau > 1$. For $\tau = 2$ results were significantly improved and with $\tau = 3$ the result was as good as with the negative potential function on RW-16.

Table 5.3 helps also in explaining why the algorithm with the negative potential function does not converge on long RWs. In this case, transitions far from the goal state receive high positive shaping rewards in both directions. This reward is constantly growing when going away from the goal state. It means that for both good transitions, $s \rightarrow s'$, and wrong transitions, $s' \rightarrow s$, Q-values become higher than zero and cause the agent to mistakenly reinforce those values by following loopy paths which involve those transitions far from the goal state. This happens because the initial value of $Q(s, a) = 0$ of all state-action pairs represents the highest possible value only with reward $r < 0$. Anything, which is higher than 0, will be mistakenly preferred.

These configurations of R_s and positive and negative potential functions were also tested on Maze and S-maze domains. Results presented in Figures 5.11 and 5.12 show the same pattern in the performance where the positive potential function is significantly inferior to the negative representation. Tests with different values of γ also show the same pattern as in RW when appropriately changing the length of RW under a constant discount factor. In this case, the negative potential function did not converge when γ was too small (e.g., $\gamma = 0.95$ on S-maze).

5.6.2.2 Evaluation with the Goal Reward R_g

In this subsection, the second general type of the reward function is investigated. The goal reward, R_g, means that the positive reward (1 in our case) is given upon entering the goal state and all other transitions receive the reward of 0.

Φ^+	F(s,s')	F(s',s)	Φ^-	F(s,s')	F(s',s)
0	0.9	-1	-16	2.5	0.6
1	0.8	-1.1	-15	2.4	0.5
2	0.7	-1.2	-14	2.3	0.4
3	0.6	-1.3	-13	2.2	0.3
4	0.5	-1.4	-12	2.1	0.2
5	0.4	-1.5	-11	2	0.1
6	0.3	-1.6	-10	1.9	0
7	0.2	-1.7	-9	1.8	-0.1
8	0.1	-1.8	-8	1.7	-0.2
9	0	-1.9	-7	1.6	-0.3
10	-0.1	-2	-6	1.5	-0.4
11	-0.2	-2.1	-5	1.4	-0.5
12	-0.3	-2.2	-4	1.3	-0.6
13	-0.4	-2.3	-3	1.2	-0.7
14	-0.5	-2.4	-2	1.1	-0.8
15	-0.6	-2.5	-1	1	-0.9
16			0		

Table 5.3: Shaping rewards from positive and negative potential functions on RW-16 with $\gamma = 0.9$.

The first series of experiments is on RW with different values of N. Results with lower values of N are not reported in graphs. For example, on RW-8 two potential functions yield the same speedup. On RW-16 the negative potential function performs worse than both positive and the SARSA baseline, though initially it is better than SARSA. For $N = 32$ and higher, the negative potential function does not allow the agent to reach the goal in a reasonable time. The positive potential function becomes worse as well when N increases. One experiment for $N = 128$ is presented in Figure 5.13. It shows typical behaviour of the positive potential function which is much better initially, when the no-shaping approach performs random exploration, but later on it is significantly worse than learning without shaping.

Experiments on the Maze task show similar properties. Figure 5.14 shows results with $\gamma = 0.95$. The positive potential function is again better only initially and has longer episodes than no-shaping after around 100 episodes. The negative potential function was unstable with this value of γ. A more detailed view of this run is in the internal part of Figure 5.14, which shows the first 300 episodes. The negative potential function, even though good initially, goes into long trajectories and stabilises again after around 3000 episodes. In unreported results, with $\gamma = 0.8$ it did not converge at all and with $\gamma = 0.99$ the graph is similar as in Figure 5.14.

The S-maze task was the most challenging for reward shaping in this configuration (graphs are not included). Learning with the negative potential function did not converge for any of tested

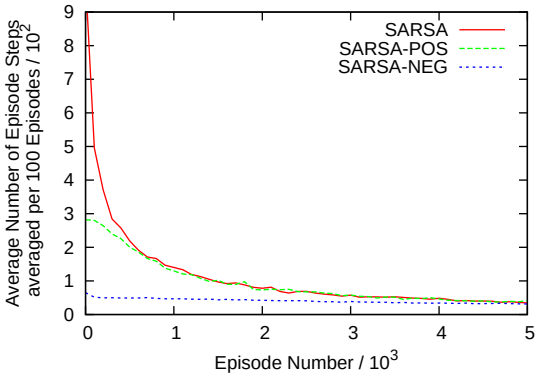


Figure 5.11: Results on Maze with $\gamma = 0.95$, R_s, and positive and negative potential functions.

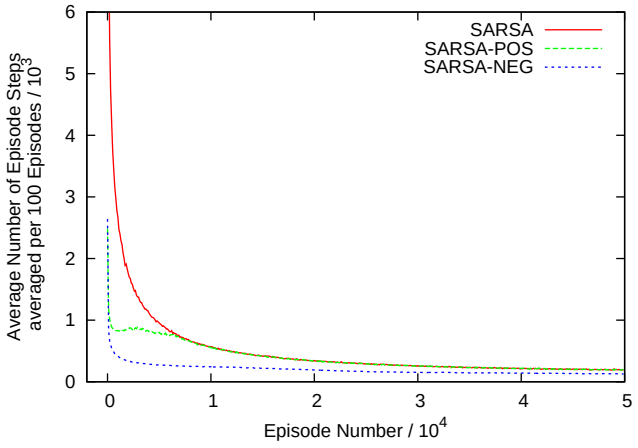


Figure 5.12: Results on S-maze with $\gamma = 0.99$, R_s, and positive and negative potential functions.

γ values, i.e. 0.9, 0.95, and 0.99. The positive potential function performs in a similar way as with other domains. Its performance drops when the discount factor decreases and with $\gamma = 0.8$ it performs significantly worse even in early episodes.

The inability to reach the goal state due to loopy trajectories when learning with the negative potential function was encountered in this section as well. The explanation of this problem which was given in Section 5.6.2.1 applies also to this configuration. The initialisation of the Q-table with a value higher than zero was required and allowed the agent to reach the goal on both RW and S-maze.

One more solution was investigated to solve the problem of the lack of convergence of the negative potential function. The treatment of transitions $s \rightarrow s$ was changed, i.e. $F(s, s)$ was

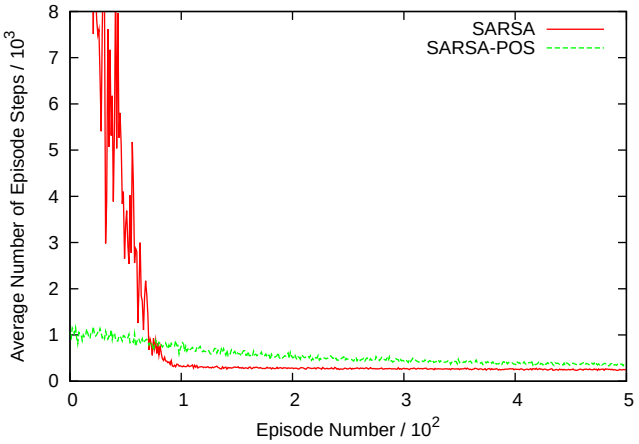


Figure 5.13: Results on RW-128 with $\gamma = 0.95$, R_g, and the positive potential function only (negative does not converge).

manually set to 0 for all states whereas everything else was left unchanged. We expected that this modification would allow the agent to avoid very long, loopy trajectories in the problematic situation. It did not, however. But, the positive potential function encountered problems with $F(s, s) = 0$. For example, on S-maze with $\gamma = 0.95$ very long, loopy trajectories arise after around 900 episodes. Our more detailed analysis revealed that when $F(s, s) = 0$ for each state, then there is no penalty for rebounding to the same state. This can happen when there is a wall in front of the agent and the move forward action will always fail. Transitions in such states can cause negative values of the Q-function. When this happens, the temporal difference for transitions $s \rightarrow s$ will be positive and the agent will wrongly prefer executing actions which cause $s \rightarrow s$. For this reason $F(s, s)$ should not be manually set to 0 but rather left as a negative value given by the potential function. These results in a natural penalty for transitions $s \rightarrow s$ (see Table 5.1).

5.7 Modified Reward Shaping Evaluation

The evaluation presented in the previous sections of this chapter and properties of reward shaping defined according to Equation 5.1 are influenced by the fact that the discount factor, γ, appears in this equation. In this section this analysis continues with a potential solution to encountered problems. The idea which is investigated here comes from the fact that if $\gamma = 1$, then it is reduced in Equation 5.1. The resulting formula to compute the shaping reward from the potential function would then take the form:

$$F(s, s') = \Phi(s') - \Phi(s). \tag{5.17}$$

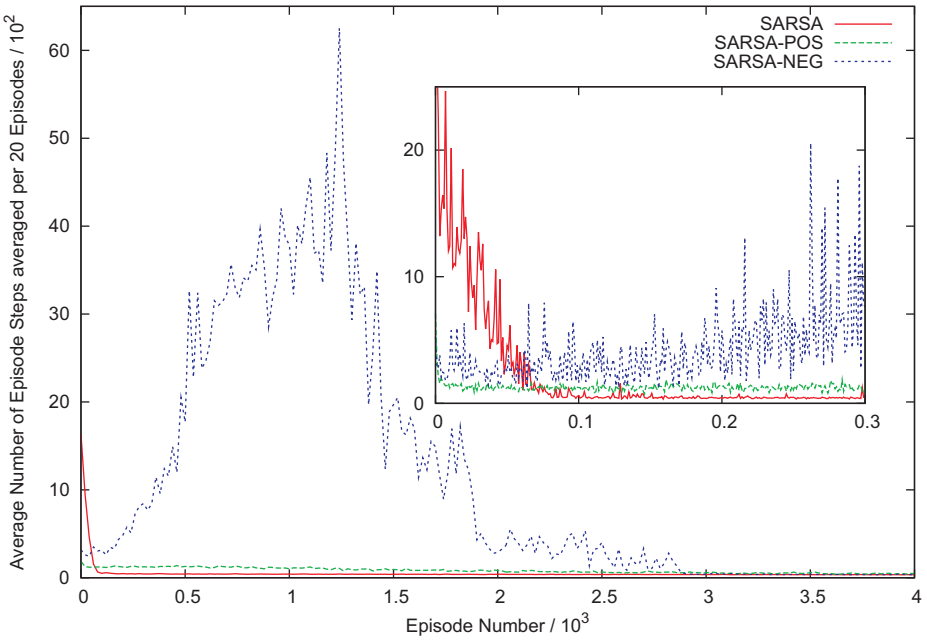


Figure 5.14: Results on Maze with $\gamma = 0.95$, R_g, and positive and negative potential functions.

With such a reformulation of the way the shaping reward is evaluated, the theorem which guarantees that the policy learned with reward shaping is equivalent to the one learned without reward shaping is not valid any more (Ng et al. 1999). Our investigation showed that this cannot be proved for the general case. This fact is explained by Theorem 1:

Theorem 1. *If $F : \mathbb{S} \times \mathbb{S} \mapsto \mathbb{R}$ is the potential-based reward shaping defined according to Equation 5.17, then the policy learned with such a reward shaping may not be equivalent to the one learned without reward shaping.*

Proof. The direct proof shows that it may be the case that $\pi^*_{M'}(s) \neq \pi^*_M(s)$. The optimal value function of M' satisfies the equation:

$$
\begin{aligned}
Q^*_{M'}(s_0,a) &= E\left\{\sum_{t=0}^{\infty}\gamma^t(r_t+\Phi(s_{t+1})-\Phi(s_t))\right\} \\
&= E\left\{\sum_{t=0}^{\infty}\gamma^t r_t\right\} + E\left\{\sum_{t=1}^{\infty}\gamma^{t-1}\Phi(s_t)\right\} - E\left\{\sum_{t=0}^{\infty}\gamma^t\Phi(s_t)\right\} \\
&= Q^*_M(s_0,a) + E\left\{\sum_{t=1}^{\infty}\gamma^{t-1}\Phi(s_t)-\gamma^t\Phi(s_t))\right\} - \Phi(s_0) \\
&= Q^*_M(s_0,a) + (1/\gamma-1)E\left\{\sum_{t=1}^{\infty}\gamma^t\Phi(s_t)\right\} - \Phi(s_0) \\
&= Q^*_M(s_0,a) + A(s_0,a) - \Phi(s_0)
\end{aligned}
\tag{5.18}
$$

where

$$
A(s_0,a) = (1/\gamma-1)E\left\{\sum_{t=1}^{\infty}\gamma^t\Phi(s_t)\right\}. \tag{5.19}
$$

It may be the case that

$$
A(s_0,a_i) \neq A(s_0,a_j) \tag{5.20}
$$

when $i \neq j$, which leads to

$$
\arg\max_{a\in A} Q^*_{M'}(s,a) \neq \arg\max_{a\in A} Q^*_M(s,a). \tag{5.21}
$$

This means that $\pi^*_{M'}(s) = \pi^*_M(s)$ does not always hold. Inequality 5.20 follows from the fact that different actions of a given state, s_0, may lead to different expected sequences of proceeding states. Inequality 5.21 follows from the fact, that if Inequality 5.20 is satisfied, then the difference of values which are compared in Inequality 5.20 may lead to the situation when actions in state s_0 have different rank according to Q^*_M and $Q^*_{M'}$. Which means that an action a_i which has its $Q^*_M(s_0,a_i)$ lower than any arbitrary action a_j may have highest $Q^*_{M'}(s_0,a_i)$ due to its very high $A(s_0,a_i)$. □

Theorem 1 shows that the policy learned with shaping reward defined by Equation 5.17 may be different from the one learned without reward shaping. The proof of this theorem explains additionally where the difference lies when comparing with the standard evaluation of reward shaping as in Equation 5.1. When Equation 5.1 is used, the following relation is satisfied (Ng et al. 1999):

$$
Q^*_{M'}(s_0,a) = Q^*_M(s_0,a) - \Phi(s_0).
$$

The corresponding relation in Equation 5.18 has an additional element of $A(s_0, a)$ defined in Equation 5.19. This factor makes $\pi^*_{M'}(s) = \pi^*_M(s)$ not hold in a general case. When $\gamma \approx 1$, the influence of this factor is small and the resulting policy is very close to the policy of M.

The discussion above showed that when Equation 5.17 is used to compute the shaping reward, this shaping reward modifies the MDP in such a way that its solution does not correspond to the solution of the original MDP. The question one may ask is whether it may still be useful to have reward shaping with such properties. One of the reasons is that reward shaping as such is intended to control exploration in a more informative way and make the agent learn faster and try less useless actions which are far from any reasonable behaviour. As mentioned in Section 2.2, in practical situations often the anticipated result of learning is not an optimal policy but one which is fast to learn and yields performance which is satisfactory. Thus, when one has such requirements, the fact that Equation 5.19 may make $\pi^*_{M'}(s) \neq \pi^*_M(s)$ would be still an acceptable solution. The policy on the modified M' may still be better when executed on M when learning with reward shaping than the policy learned on M without reward shaping.

Another interesting advantage of using Equation 5.17 is that it is robust against cycles during exploration (Ng et al. 1999). Potential-based reward shaping has been shown to be a solution to this problem (Randløv & Alstrom 1998). With Equation 5.17 the sum of shaping rewards for following the loop is always 0, whereas when $\gamma < 1$ with Equation 5.1 the negative potential function still yields some positive reward for cycles and the positive potential function penalises for such a behaviour.

The following experimental analysis was performed with the modified reward shaping according to Equation 5.17.

5.7.1 Empirical Tests with R_s

In the first instance learning with R_s was evaluated. Figure 5.15 shows results for exactly the same settings as in Figure 5.10 and Equation 5.17. This time both positive and negative potential functions gain the same speed-up from reward shaping. In results shown in Figure 5.10 the negative potential function was unstable. Additional tests with a longer version of the task did not converge with the standard reward shaping, but in our case both potential functions converge on $N = 128$ as shown in Figure 5.16. The result on the Maze domain is in Figure 5.17, and with our extension both methods to evaluate the potential function yield the same improvement. In the corresponding experiment presented in Figure 5.11 the positive potential function was inferior.

The last experiment for this reward type was on the S-maze domain. As in the previous case learning with reward shaping was problematic. Both versions of the potential function led to loopy paths which prevented the algorithm from reaching the goal state even in first episodes of learning. Closer analysis revealed that the agent gets trapped in the corner, (55, 47), from which backtracking is necessary to reach the goal.

Experiments presented in this section show that our modification to reward shaping which

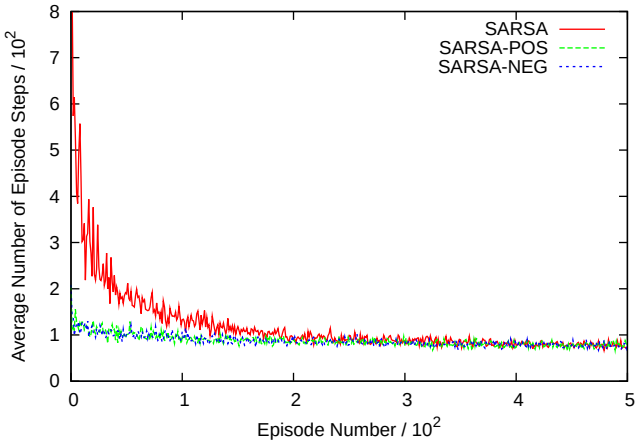


Figure 5.15: Results on RW-40 with $\gamma = 0.95$, Equation 5.17, R_s, and the positive and negative potential functions.

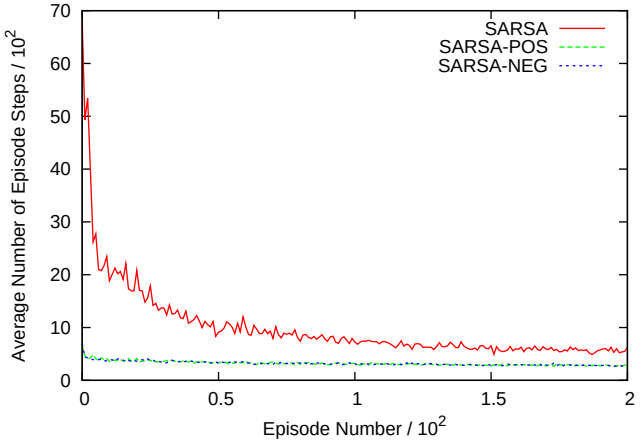


Figure 5.16: Results on RW-128 with $\gamma = 0.95$, Equation 5.17, R_g, and the positive and negative potential functions.

assumes the use of Equation 5.17 to evaluate the shaping reward improves learning on domains in which the agent does not have to backtrack by following long state sequences.

5.7.2 Empirical Tests with R_g

Experiments with the standard evaluation of the shaping reward revealed that R_g was challenging for the negative potential function. Figures 5.18 and 5.19 show that the modified potential evaluation leads to equally successful learning of the two shaping approaches, that is, Φ^+ and Φ^-.

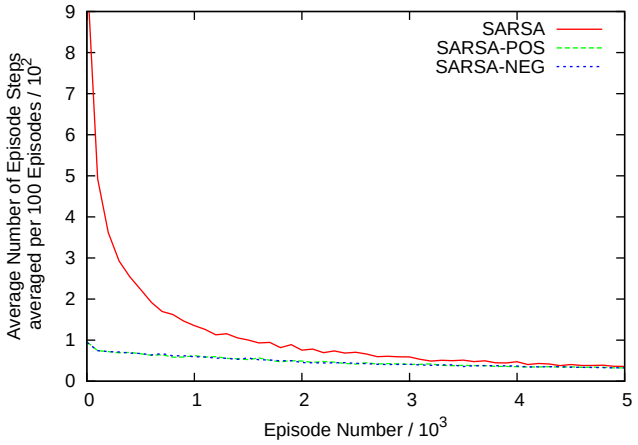


Figure 5.17: Results on Maze with $\gamma = 0.95$, Equation 5.17, R_s, and the positive and negative potential functions.

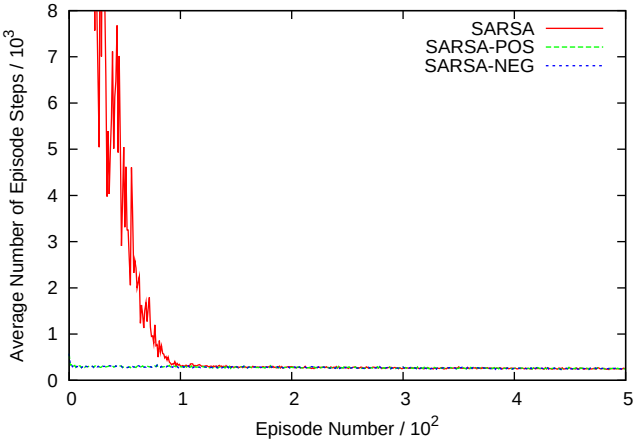


Figure 5.18: Results on RW-128 with $\gamma = 0.95$, Equation 5.17, R_g, and the positive and negative potential functions.

The last experiment on the S-maze task showed similar problems as with R_s and the use of Equation 5.17. The algorithm did learn however successfully in the initial period of a few hundred episodes but the problem of loopy paths appeared later on. The agent was stuck in the the same problematic corner as in the case from the previous sub-section. Again, the behaviour of the algorithm is attributed to the properties of the domain which has a dead end in the corner (55, 47) and the quality of the heuristic function used to evaluate the potential function which rewards the agent for moving towards this state.

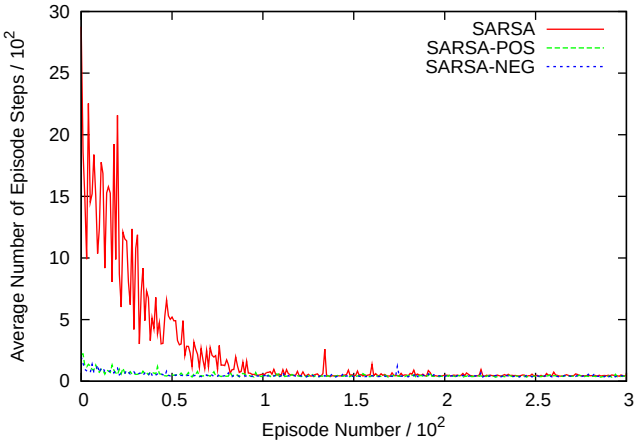


Figure 5.19: Results on Maze with $\gamma = 0.95$, Equation 5.17, R_g, and positive and negative potential functions.

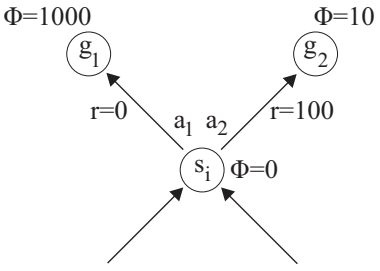


Figure 5.20: A simple MDP where the straightforward application of potential-based reward shaping of Ng et al. (1999) leads to a different policy than non-shaped learning.

Experiments with R_g show similarly that learning with both types of potential function in domains, which do not have properties described in the previous paragraph, is successful and yields improvements over the non-shaping algorithm.

5.8 The Potential Function in Multi-Goal Domains

This section extends the standard formulation of potential-based reward shaping (Ng et al. 1999) by identifying situations when the straightforward application of this approach may fail and by proposing a solution to the encountered problem. This failure can happen in multi-goal tasks where the potential function between goals is different and does not correspond to the quality of those goals. The problem can be easily presented in detail in the following scenario shown in Figure 5.20. There are three states in this domain, two of which are goal states (g_1 and g_2). It can be easily computed that $Q^*_{M'}(s_i, a_1) = 1000$ and $Q^*_M(s_i, a_2) = 110$ whereas the true value of $Q^*_M(s_i, a_1) = 0$ and $Q^*_M(s_i, a_2) = 100$ when $V^*_{M'}(g_i) = 0$, $V^*_M(g_i) = 0$, and $\gamma = 1$. Thus,

in this simple domain $\pi^*_{M'} \neq \pi^*_M$, and also $Q^*_{M'}(s_i, a_i) + \phi(s_i) \neq Q^*_M(s_i, a_i)$. This situation shows that the policy obtained via learning with potential-based reward shaping may be different from the one which is learned without reward shaping (even though reward shaping is based on the potential function and Theorem 1 of Ng et al. 1999 is satisfied).

The first straightforward solution to this problem can be achieved by the explicit modification of the shaping reward with $F(s, a, g_i) = 0$, that is, transitions to any of the goal states should not be rewarded by the shaping reward. When such an extension would be added to the standard definition of the potential-based reward shaping in (Ng et al. 1999), i.e. to Equation 2 in Theorem 1, the modified potential-based reward shaping would overcome problems presented in Figure 5.20 and would maintain all existing properties.

Another solution to this problem is that the potential function of goal states, g_i, can be treated in a special way. The intuition behind this is that since different values of the potential function in goal states influence the value of their predecessor states in M', they should also influence the values of goal states in M'. When solving M with value iteration, the following relation $V^*_M(g_i) = 0$ should be satisfied for all goals. In this way the fixed Q-value of goal states will not influence the policy defined by the reward in the existing environment. In the domain M' with reward shaping, $V^*_{M'}(g_i)$ cannot be set to 0 as in the case of M. To balance the influence of the potential function of the goal state, the Q-table in M' should be initialised as $V^*_{M'}(g_i) = V^*_M(g_i) - \phi(g_i)$. This however would provide the solution to the considered problem only under the assumption that $\gamma = 1$. With $\gamma < 1$ this may still lead to $\pi^*_{M'}(s) \neq \pi^*_M(s)$. In the scenario in Figure 5.20 it is the case when $\gamma < 89/99$. The importance of how the value of goal states is initialised is often omitted when dealing with goal-based domains. In tabular representations values of these states once initialised are not modified by the algorithm but because of backpropagation they influence values of other states. The relevant work of the author of this thesis was presented in (Grześ & Kudenko 2009) where this issue was analysed in model-free learning without reward shaping.

This section shows an exceptional situation when the standard formulation of the potential-based reward shaping of Ng et al. (1999) fails and shows how to extend this standard definition of potential-based reward shaping. It is worth noting that the problem described here also applies to model-based learning with potential-based reward shaping formalised and theoretically proved for the Rmax algorithm in (Asmuth et al. 2008). An analogous solution would be needed in this case as well.

5.9 Summary and Discussion

This chapter presents novel theoretical and empirical insight into RL with reward shaping that every RL practitioner should be aware of. The overall contribution of this chapter can be summarised as follows:

- When $\gamma = 1$, the potential function can be both positive and negative and in both cases the performance is exactly the same.

- Even when $\Phi(s) = V(s)$, the learning algorithm still needs to learn effects of actions and for this reason scaling the shaping reward up ($\tau > 1$) improves the learning rate, because the exploration is improved. When $\tau > 1$, the theoretical requirements of potential-based reward shaping are preserved.

- In domains with faulty heuristics one cannot scale the potential function up too much because the agent may be heavily penalised for diverging from the shaping reward and this may result in failures in reaching the goal state. However the scaling factor, τ, with the value of 2 yielded the best results on all tested domains with different quality of the heuristic function and this value can be considered in practical applications.

- The analysis of the actual shaping reward in domains with $\gamma < 1$ was conducted (summary in Tables 5.1 and 5.2) and results allow explaining the outcomes of the empirical analysis.

- When $\gamma < 1$ and learning with R_s, the positive potential function performs worse than the negative one and the scaling factor $\tau > 1$ improves learning with the positive potential function. The negative potential function is better in the initial configuration but breaks when conditions defined in Equations 5.11-5.13 are significantly violated. This can happen even with a very accurate heuristic function as shown on RW (see Figure 5.10). The re-initialisation of the Q-table to higher values (e.g., 100) allows avoiding very long, loopy episodes which prevent the algorithm from reaching the goal state.

- The goal reward, R_g, seems to be more challenging to reward shaping. Generally, both types of potential function lead to a considerable improvement only at the very beginning of learning, when the no-shaping agent performs random exploration. For higher lengths of RW (e.g., $N = 128$) or generally situations when conditions in Equations 5.2-5.4 are violated to a higher extent, the positive potential function, even though good initially, is significantly worse than no shaping. The negative potential function leads in these cases to loopy episodes, and a different initialisation of the Q-table is required to allow the agent to reach the goal state in a reasonable time.

- Additional analysis of the previous case revealed that transitions $s \rightarrow s$ should be rewarded according to the standard evaluation of the shaping reward, $F(s, s)$, because these transitions should be constantly penalised (Table 5.1).

- A new method to evaluate the shaping reward from the potential function was proposed (see Equation 5.17). It was proven that with such a reward shaping the optimal policy of the shaped MDP may be different from the original one which does not use shaping. However, in large domains, where approximate solutions are satisfactory it may still be more profitable to learn such a shaped policy quickly than the original one slowly, and possibly not reaching the same performance on the main MDP.

- Experimental evaluation of this approach, Equation 5.17, showed that it solves the problem of different learning with the positive and negative potential functions on tasks where the heuristic function which is used as the potential function is relatively accurate. The problems were still encountered on the S-maze task where the heuristic function is the faultiest.
- The final contribution of this chapter shows that even the standard potential-based reward shaping approach may lead to a policy which is not equivalent to the policy of the original MDP on domains with many goal states. The potential function of goal states requires special treatment in such cases.

Our findings do not violate the relevant theory on potential-based reward shaping (Ng et al. 1999; Wiewiora 2003). When Equation 5.1 is used the optimal policy of the shaped MDP is the same as of the original non-shaped MDP. The problem on which our research focuses is how successful exploration with different reward shaping approaches and different algorithm and domain properties is. Since this kind of reward shaping has the equivalent initialisation, the same problems can be encountered with the corresponding initialisation of the value function. The proposed shaping with Equation 5.17 has good potential of applicability in large domains where the optimal solution cannot be found in a reasonable time (or even very long time), but the policy of the shaped MDP can be learned faster and represent a more accurate solution to the target MDP.

CHAPTER 6

PAC-MDP Learning with Knowledge-based Admissible Models

PAC-MDP algorithms approach the exploration-exploitation problem of reinforcement learning agents in an effective way which guarantees that with high probability, the algorithm performs near optimally for all but a polynomial number of steps. The performance of these algorithms can be further improved by incorporating domain knowledge to guide their learning process. In this chapter, we propose a framework to use partial knowledge about effects of actions in a theoretically well-founded way. Our proposed method requires specific knowledge about effects of actions. Empirical evaluation shows that when this knowledge is available, our method outperforms reward shaping when reward shaping is based on admissible heuristics (e.g., the straight-line distance to the goal). Our solution is also very competitive when compared with the Bayesian Exploration Bonus (BEB) algorithm. BEB is not PAC-MDP, however it can exploit domain knowledge via informative priors. We show how to use the same kind of knowledge in the PAC-MDP framework in a way which preserves all theoretical guarantees of PAC-MDP learning.

6.1 Introduction

One of the best approaches to exploration in RL, which has good theoretical properties, is the so called PAC-MDP approach. State-of-the-art examples of this idea are E^3 (Kearns & Singh 2002) and $Rmax$ (Brafman & Tennenholtz 2002). This approach defines the exploration strategy which guarantees that with high probability the algorithm performs near optimally for all but a polynomial number of time steps (i.e. polynomial in the relevant parameters of the underlying

process).

Most of RL research has focused on the situation when knowledge about the mathematical model of the underlying process is very limited. This is however not always the case in practical applications where some domain knowledge may exist. For example, Poupart et al. (2006) argue that some knowledge can be easily available in navigation scenarios. They also give a concrete example from the area of assistive technology where, in the RL-based hand-washing device, the transition dynamics are known except for the behaviour probabilities of people with dementia who use the system (Boger et al. 2005). In this chapter, the use of such partial knowledge about actions of the underlying controlled process is considered to improve the performance of PAC-MDP learning. This is only partial knowledge because it is not sufficient to design the analytical model of the underlying process and (reinforcement) learning is still necessary to solve the problem. The resulting approach is shown to preserve theoretical properties of PAC-MDP learning.

Bayesian techniques can be naturally enhanced with background knowledge through informative priors. A relevant Bayesian approach to the problem of exploration in RL has been recently introduced by Kolter & Ng (2009). This algorithm applies slightly greedier exploitation than the one which is in PAC-MDP algorithms. This may lead to improvements in some practical situations, however this greedier exploitation makes it not PAC-MDP (see the proof of Kolter & Ng 2009). In this chapter, we want to show that knowledge which in Bayesian approaches can be used to define informative priors can also be used in the PAC-MDP framework in a relatively straightforward way and yields a very good empirical improvement of the state-of-the-art algorithms. This allows for the use of background knowledge, obtaining an algorithm which is competitive with the Bayesian approach, and most importantly is still PAC-MDP. The lack of such techniques was one of the points of criticism against PAC-MDP algorithms in (Kolter & Ng 2009).

6.2 PAC-MDP Algorithms

PAC-MDP learning represents one of the approaches to exploration in RL. Such algorithms are based on the technique known as optimism in the face of uncertainty (Kearns & Singh 2002; Brafman & Tennenholtz 2002). Like in standard model-based learning, the dynamics of the underlying MDP are estimated from data. If a certain state-action pair has been experienced enough times (parameter m), then the Hoeffding bound (which is explained in what follows) ensures that the estimated dynamics are close to the true values. Let p be the probability of success in the Binomial distribution, and $\hat{p}$ its empirical estimate. The Hoeffding bound shows that the probability of $|\hat{p} - p| > \epsilon$ for some ϵ is bounded by $\exp(-2m\epsilon^2)$ where m is the number of trials used to estimate $\hat{p}$ (Kearns & Vazirani 1994). It means that for a given value of ϵ and desired probability, it is possible to determine analytically the value of the number of required trials, m. The optimism under uncertainty plays a crucial role when dealing with state-action

pairs which have not been experienced m times. For such pairs, the algorithm assumes that their Q-values have the highest possible value, that is, $Rmax$ when $\gamma = 1$ or $Rmax/(1-\gamma)$ when $\gamma < 1$, where $Rmax$ is the upper bound of the reward function. State-action pairs for which $n(s,a) < m$ are named unknown and known when $n(s,a) \geq m$ where $n(s,a)$ is the number of times the state-action pair was experienced. When a new state action pair becomes known, the existing approximation, $\hat{M}$, of the true model, M^*, is used to compute the corresponding optimal policy for $\hat{M}$ which when executed will encourage the algorithm to try unknown actions and learn their dynamics. Such an exploration strategy guarantees that with high probability the algorithm performs near optimally for all but a polynomial number of time steps (i.e. polynomial in the relevant parameters of the underlying MDP).

The precise implementation of this idea is different in existing algorithms. One of the differences is the way in which the planning step is implemented. The general equation for performing value iteration for computing the policy, $\hat{\pi}$, for the model $\hat{M}$ can be as follows:

$$\hat{Q}(s,a) = \hat{R}(s,a) + B(s,a) + \gamma \sum_{s'} \hat{T}(s,a,s') \max_{a'} \hat{Q}(s',a'), \tag{6.1}$$

where $B(s,a)$ is an algorithm specific exploration bonus. In the Rmax algorithm, $B(s,a) = 0$ for all state-action pairs. In the Model Based Interval Estimation with Exploration Bonus (MBIE-EB) algorithm, which is also PAC-MDP, $B(s,a) = \beta/\sqrt{n(s,a)}$ where β is a constant value provided as an input to the algorithm (Strehl & Littman 2008).

The proofs and theoretical analysis of PAC-MDP algorithms can be found in the relevant literature (Kakade 2003; Strehl & Littman 2008). In our analysis one specific property of such algorithms is advocated, i.e. the optimism under uncertainty, which requires that inequality $\hat{V}(s) \geq V^*(s)$ is always satisfied during learning, where $V^*(s)$ is the optimal value function which corresponds to the true MDP model M^*.

6.3 Near Bayesian Learning

A relevant algorithm which tackles the exploration problem in a way which originates from optimism in the face of uncertainty has been recently proposed by Kolter & Ng (2009). The Bayesian Exploration Bonus (BEB) algorithm implements computationally tractable approximation of the Bayesian exploration. An important fact from our point of view is that this algorithm differs from the PAC-MDP algorithm mainly in a way the exploration bonus, $B(s,a)$, is computed. In the BEB algorithm $B(s,a) = \beta/(1+n(s,a))$. Because n increases faster than $\sqrt{n}$, it has been proven (Kolter & Ng 2009) that BEB is not PAC-MDP. The exploration bonus decays too fast in BEB and does not allow for enough exploration to satisfy the PAC-MDP requirements. Thus, BEB is not guaranteed to converge to the optimal solution. A useful property of the Bayesian approach is however the fact that it can use domain knowledge in a straightforward way through informative priors. The issue of using such knowledge in the PAC-MDP framework has not

been analysed and was considered to be its weakness (Kolter & Ng 2009). In our work we are investigating how to use the same knowledge in PAC-MDP algorithms.

6.4 Domain Knowledge and Admissible Models

The standard scenario when RL is applicable is the situation when dynamics of the controlled stochastic decision process are not available. Actions of such a process can be formally specified with the use of the Probabilistic Planning Domain Description Language (PPDDL) (Younes & Littman 2004). Each action a is specified in this notation by defining the probability, p_i, of each probabilistic effect, e_i, in the following way:

$$(a\ p_1\ e_1\ ...\ p_n\ e_n).$$

When this notation is used to describe the probabilistic planning problem in which the entire model is known beforehand, all p_i and e_i have to be specified for all actions. In the standard RL scenario, neither p_i nor e_i are known. However, it is often the case, that even if the entire model is not available, some elements can be determined beforehand (see reference to Boger et al. 2005; Poupart et al. 2006 in Section 6.1). This chapter aims at improving PAC-MDP learning when partial action knowledge is available. Before going into details on what kind of knowledge will be considered, the definition below introduces the notion of admissible MDP models which will determine theoretical requirements on application of this knowledge.

Definition 1. *A model $\hat{M}$ is admissible iff the corresponding value function, $\hat{V}$, satisfies inequality $\hat{V}(s) \geq V^*(s)$, that is, $\hat{V}(s)$ is admissible.*

An initial model, $\hat{M}$, has to be admissible in order to preserve PAC-MDP properties when used with existing algorithms which are PAC-MDP (Brafman & Tennenholtz 2002; Strehl & Littman 2008).

6.4.1 Optimistic Determinization

In this section we are dealing with RL problems for which all possible outcomes, e_i, of each action, $a \in \mathbb{A}$, can be determined by the designer of the system whereas probabilities of outcomes, p_i, still remain unknown as in the classical RL scenario (such a situation exists, e.g., in the area of assistive technology Boger et al. 2005; Poupart et al. 2006). Under this condition, learning from simulation is still one of the potential solutions and RL is applicable to solve sequential decision making problems of this kind. The question now is, how to use PAC-MDP algorithms with such knowledge about the possible effects of actions. Our solution is motivated by probabilistic planners which apply determinization of stochastic domains (Blum & Langford 1998; Yoon et al. 2007). One of the approaches in (Yoon et al. 2007) yields an admissible deterministic model and is based on all-outcomes determinization (AO). In this case, the determinization process creates a new deterministic action $a_d \in \mathbb{A}_d(a)$ for each possible effect, e_i, of a given action a. The

new set of deterministic actions replaces the original action in the new MDP model. It is worth remembering that these are fictitious actions which are exclusively used to create our admissible model. The agent, while acting in the environment, executes real actions which remain probabilistic. The obtained model is in this case admissible with regard to the original probabilistic one. When this type of determinization is applied in the FF-Replan algorithm (Yoon et al. 2007), probabilities of action outcomes are ignored. In our case this situation is ideal, since we do not have those probabilities in our RL settings anyway. The fact that the AO model is admissible can be easily proven in the same manner as Lemma 6 in (Strehl & Littman 2008) proves that models with the upper bound of the estimated interval guarantee admissibility of the value function with high probability. The proof for the case with the AO model is similar and proves that the corresponding value function is admissible with probability one.

Lemma 1. *For any state s and action a, the condition $\hat{Q}(s,a) \geq Q^*(s,a)$ is satisfied after value iteration on the MDP $\hat{M}$ which is obtained from AO knowledge.*

Proof. Value iteration solves the MDP $\hat{M}$ defined according to AO knowledge. We prove the claim by induction on the number of steps of value iteration which is stopped after a finite number of iterations. For the base case, assume that the Q values are initialised to $Rmax$ when $\gamma = 1$ or $Rmax/(1-\gamma)$ otherwise, for all s. Now, for the induction, suppose that the claim holds for the current value function $\hat{Q}(s,a)$. By assumption, the reward $R(s,a)$ is known exactly and $\hat{T}(s,a,s') = \hat{T}(s, a_d \in \mathbb{A}_d(a), s') = 1$ and $\hat{T}(s,a,s') = 0$ only when $T(s,a,s') = 0$ for sure (according to AO knowledge). The term $Q(s',a')$ on the right-hand side of Equation 6.1 is the result of the previous iteration and is used to compute the new Q-value $\hat{Q}(s,a)$ on the left-hand side of the equation. By our assumption we know $R(s,a)$ exactly and:

$$\begin{aligned}\sum_{s'} \hat{T}(s,a,s') \max_{a'} \hat{Q}(s',a') &= \max_{a_d} \left\{ T(s,a_d,s') \max_{a'} \hat{Q}(s',a') \right\} \\ &= \max_{a_d} \max_{a'} \hat{Q}(s',a') \geq \sum_{s'} T(s,a,s') \max_{a'} \hat{Q}(s',a') \\ &\geq \sum_{s'} T(s,a,s') \max_{a'} Q^*(s',a')\end{aligned}$$

The first step is from the definition how to use a_d to determine values of a. The second and the third steps follow from the assumption that $\hat{T}(s,a,s') = 1 \geq \max_{s'} T(s,a,s')$ or $\hat{T}(s,a,s') = 0$ only when $T(s,a,s') = 0$ for sure, and the fourth from the induction assumption. □

Normally, probabilistic effects reduce the value function via transitions to lower value states and AO determinization leads to higher values in such situations because only the state with the highest value (which is achieved with probability 1.0 in the modified model) is used in the Bellman update.

6.4.2 Free Space Assumption

The free space assumption (FSA) is an approach to define an initial model of the environment which assumes that all transitions in the environment are possible (in robotic navigation environments it would assume, e.g., that there are no walls between all adjacent states or, in the case of the hand washing device of Boger et al. 2005, the person with dementia always behaves like a rational healthy person), and that all actions are deterministic and always lead to a corresponding expected state (Rayner et al. 2007). In the PPDDL notation, it would mean that a real action a is replaced by outcome e_i for which $e_i = \arg\max_{e_i} p_i$, whereas all p_i may stay unknown. The best outcome selected in this way may fail in the real environment (with $p_i = 0$) however by our assumption other outcomes of this action have their highest p_i in a different action. Thus, without much loss of generality and for the sake of theoretical properties of our solution, we require that for each possible outcome which does not correspond to the most expected outcome of a given action, there is another action which has this outcome as the most expected one. This requirement is necessary to guarantee the admissibility of such a model because all outcomes have to be tested during learning (unless they are blocked). In the hypothetical robotic environment, the formulation of the FSA model would mean, e.g., that an action move forward, always moves the robot from a given state to the state in front of the robot, ignoring any existing walls and probabilistic effects of actions like, e.g., slippery surface which would slow down robot's movement or change the direction of its motion. As in AO determinization above, actions in the FSA model are not genuine and are solely used to create our admissible model. The robot, while moving in the environment, executes real actions which remain probabilistic.

The fact that the FSA model is admissible with probability one can also be proved in a similar way as Lemma 1. Admissibility of $\hat{V}(s) = \max_a \hat{Q}(s,a)$ guarantees optimistic behaviour when the highest Q-value is used greedily.

Lemma 2. *For any state s and action a, the condition $\hat{V}(s) \geq V^*(s)$ is satisfied after value iteration on the MDP $\hat{M}$ which is obtained from FSA knowledge.*

Proof. Value iteration solves the MDP $\hat{M}$ defined according to FSA knowledge. We prove the claim by induction on the number of steps of value iteration which is stopped after finite number of iterations. For the base case, assume that the Q values are initialised to $Rmax$ when $\gamma = 1$ or $Rmax/(1-\gamma)$ otherwise, for all s. Now, for the induction, suppose that the claim holds for the current value function $\hat{V}(s)$. By assumption, the reward $R(s,a)$ is known exactly and $\hat{T}(s, a_{FSA}, s') = 1 \geq \max_{s'} T(s,a,s')$. If $T(s, a_{FSA}, s') = 0$, then another effect is better which is FSA of another action, so another action will be better for such an effect.

Equation 6.1 can be expressed also in terms of the value function V. The term $\hat{V}(s')$ on the right-hand side of such an equation is the result of the previous iteration and is used to compute the new V-value $\hat{V}(s)$ on the left-hand side of the equation. By our assumption we know $R(s,a)$

exactly and:

$$\begin{aligned}\max_a \left\{\gamma \sum_{s'} \hat{T}(s,a,s')\hat{V}(s')\right\} &= \max_a \left\{\gamma \hat{T}(s,a_{FSA},s')\hat{V}(s')\right\} \\ &= \max_a \left\{\gamma \hat{V}(s')\right\} \geq \max_a \left\{\gamma \sum_{s'} T(s,a,s')\hat{V}(s')\right\} \\ &\geq \max_a \left\{\gamma \sum_{s'} T(s,a,s')V^*(s')\right\} \geq V^*(s)\end{aligned}$$

The first step removes summation because each FSA action of a given action a is deterministic. The second step follows from this property $\hat{T}(s,a,s'_{FSA}) = 1$, the third from $\max_{s'} T(s,a,s') \leq 1$, and the fourth is from the induction assumption. □

Informally, the value function computed with the FSA model is always at least as high as the true value because the FSA model will utilise shorter optimistic paths (shorter because of unblocked transitions in the FSA model) and assume that corresponding transitions have always probability 1 (in the real model $p_i \leq 1$).

6.4.3 Maximal Probability

The two previous sections rely on knowledge about e_i in the PPDDL specification without information about exact values of probabilities p_i. Useful information may be available however in the form of $\varrho = \max p_i$, where max is over the entire state-action space. So, ϱ represents the maximal possible p_i which can occur in a given MDP. The value of ϱ is useful when $\varrho < 1$ as it can be used, e.g., for more accurate evaluation of the admissible potential function for reward shaping in PAC-MDP algorithms (Asmuth et al. 2008). In our analysis it will be shown how to use knowledge about ϱ in our algorithm and how to enhance existing algorithms with which we are comparing our solution in order to obtain more fair comparison and gain better insight into the problem.

There is one more noteworthy issue about domain knowledge in RL. One well established existing way of incorporating domain knowledge into RL is reward shaping (Asmuth et al. 2008; Ng et al. 1999; Randløv & Alstrom 1998). It requires however knowledge about a sufficiently accurate admissible heuristic in order to preserve PAC-MDP properties of the algorithm. The problem is that in many practical applications it is difficult to define such a heuristic manually, which is the case in domains with symbolic PPDDL-like representations. The use of knowledge which is discussed in this section aims also at dealing with situations when such heuristics cannot be designed. The aim is to use knowledge from this section in an alternative way to reward shaping which is not applicable when there is no admissible heuristic. For better understanding of the problem, our solution will be compared with the reward shaping technique on a domain

where the required heuristics can be easily defined.

The main contribution of this chapter is a method to apply AO (Section 6.4.1) and FSA (Section 6.4.2) knowledge in PAC-MDP algorithms while preserving the PAC-MDP property. We also show how to effectively use knowledge about ϱ in various RL algorithms.

6.5 PAC-MDP Learning with Admissible Models

The extension to PAC-MDP learning which is proposed in this section can be applied to any PAC-MDP algorithm. In this chapter we are focusing on the Rmax (Brafman & Tennenholtz 2002) and MBIE-EB (Strehl & Littman 2008) algorithms. These algorithms apply the standard procedure of PAC-MDP learning, that is, model estimation, optimism in the face of uncertainty when dealing with unknown state-action pairs and planning according to Equation 6.1. Our modification is associated mainly with how the process of estimating the MDP model is handled. The use of the AO model requires also specification on how actions are selected for acting and how updates in Equation 6.1 are performed when AO actions are in the model.

The special treatment of the model used during learning is required because background knowledge needs to be incorporated. In standard Rmax learning, one can distinguish two stages of learning the dynamics of a particular state-action pair (s, a). Initially, when there are no previous executions of (s, a) or when the number of executions does not exceed the value of m, optimism under uncertainty is applied. The second stage is about (s, a) pairs which have been executed at least m times. The use of background knowledge which is considered in this chapter improves the way unknown state-action pairs are dealt with (the first stage). Instead of using standard optimism under uncertainty which uniformly rewards each state-action pair with the analytically highest value function, we are proposing using domain knowledge to deal with this particular detail of PAC-MDP learning in a more informative way. The solution which we propose is based on combining two MDP models during learning and using them to estimate one Q-function. The first model is the knowledge-based admissible model which can be designed before the learning process is executed. The second model is the standard model used for estimating transition probabilities in PAC-MDP algorithms. The key idea is to use the knowledge-based model for all state-action pairs which are still not known and the true estimation from experience for all state-action pairs for which $n(s, a) \geq m$. This procedure is summarised in Table 6.1. At this time it is worth remembering that $m = 1$ in MBIE-EB. Summarising our idea and explaining Table 6.1, our approach is to always use 1) either the knowledge-based model (AO or FSA) for unknown state-action pairs, or 2) the estimated model for known state-action pairs (i.e. those for which $n(s, a) \geq m$). Because both of these models are admissible, the overall model will also be admissible and it does not violate properties of PAC-MDP learning. The description below explains how to implement this idea with AO and FSA knowledge respectively.

	Admissible Model	Estimated Model
$n(s,a) < m$	✓	
$n(s,a) \geq m$		✓

Table 6.1: The use of knowledge-based admissible models.

6.5.1 The AO Model

The application of the AO model in the general approach presented above requires some specific changes. The first modification is that the Q-table has to contain all real actions and all AO actions for each state. Q-values of real actions and their AO determinizations are used exclusively for both planning and acting (according to Table 6.1), that is, action a in a given state is used when it is known, or is substituted for its determinizations when not known. The second issue concerns the management of actions in the model. Initially the AO model contains each separate action corresponding to every outcome of each real action. When a corresponding real action becomes known during learning in the estimated model, every AO action which corresponds to the learned action is removed from the model (unless there are other unknown real actions which share a given outcome). The second issue is the planning part of the algorithm with the AO model. When value iteration is performed according to Equation 6.1, real actions are used in the evaluation in the right part of this equation. The AO model is however composed also of actions obtained from determinization. Each real action a has a corresponding set of deterministic actions, $a_d \in \mathbb{A}_d(a)$. Our solution to planning with the AO model, is to determine the state-action value of the unknown action a using $Q(s,a) = \max_{a_d \in \mathbb{A}_d(s)} Q(s,a_d)$. Each $Q(s,a)$ of a real action is equal to maximal $Q(s,a_d)$ of all its determinizations, $a_d \in \mathbb{A}_d(a)$. A similar approach is necessary also for selecting the best action to execute. When the highest Q-value is due to AO action a_d, the real action a for which $a_d \in \mathbb{A}_d(a)$ is executed (ties are broken randomly).

6.5.2 The FSA Model

The application of the FSA model requires fewer modifications. The handling of the model is easier, because there is exactly one action in the FSA model which corresponds to the real action. Thus, the computation of Q-values does not require any modifications as well as the selection of actions to execute, and these elements may remain unchanged. The handling of the FSA model can be encapsulated entirely in procedures for learning and representing the overall model. The Q-table represents values for real actions only and FSA actions are used until a corresponding real action becomes known in the estimated model.

6.5.3 Maximal Probability Knowledge

Knowledge about the maximal probability, ϱ, can be used both with AO and FSA models. By default, these two models are deterministic and the blocked transitions are not only assumed to be open, but also they can be successful with probability 1. When the value of ϱ is known, these

models can be made more accurate while their admissibility will be preserved. The information about ϱ can be incorporated via the modification of the reward, r, in the model which becomes $r = r/\varrho$ when $r < 0$ and $r = r\varrho$ when $r > 0$. This modification is performed for updates of all non-real actions. A similar trick was applied in (Asmuth et al. 2008) for more accurate evaluation of the potential function for reward shaping where it was shown that it leads to an optimistic value function. This knowledge will be also incorporated into other algorithms for a fair comparison in the experimental section.

6.6 Experimental Validation

The proposed technique to incorporate background knowledge into the considered family of RL algorithms was evaluated empirically on the navigation maze task which was introduced in Section 5.4.2 and shown in Figure 5.2. Without much loss of generality the reward model is assumed be known to the agent, which is commonly assumed in the relevant literature (Asmuth et al. 2008). The MDP discount factor γ is 1 in this task.

Experiments were conducted on a number of algorithms. The details of their settings and parameters are as follows:

- Rmax: The Rmax algorithm with $m = 5$. The versions of this algorithm with AO, Rmax-AO, and FSA, Rmax-FSA, models comprise the major contribution of this chapter.
- Rmax with reward shaping (Asmuth et al. 2008): The Manhattan, *RS(Manhattan)*, and straight line, *RS(Line)*, heuristics are used. The potential function was evaluated as $\Phi(s) = r_s \times h(s)/\varrho + r_g$, where $r_s \leq 0$ is the step reward, $h(s)$ is the heuristic estimation of the distance from state s to the goal G, and r_g the reward given when the goal state is reached. The Manhattan heuristic, which is more accurate, was evaluated also with AO and FSA knowledge in the following way: Instead of using uniform optimistic values for Q-values of unknown state-action pairs, with the use of AO and FSA knowledge they can be assigned values which differ within a given state and are more informative. Thus, with AO knowledge $Q(s,a) = V_{max} + max_{a_d \in \mathbb{A}(a)} F(s, a_d, s')$, where $a_d \in \mathbb{A}(a)$ are all determinizations of action a, and with FSA knowledge $Q(s,a) = V_{max} + F(s, a_{FSA}, s')$, where a_{FSA} is the FSA action of a. Algorithms, Rmax-AO and Rmax-FSA, which comprise the major contribution of this chapter use the same knowledge, and for fair comparison reward shaping was also enhanced with this knowledge.
- MBIE-EB (Strehl & Littman 2008): Before evaluating, this algorithm was tuned for optimal values of the β parameter and the best value was selected for comparisons (such a methodology was also used in Kolter & Ng 2009; Strehl & Littman 2008). This parameter, β, was evaluated for each configuration separately. This algorithm is also evaluated with AO, MBIE-EB-AO, and FSA, MBIE-EB-FSA, knowledge-based models as proposed in this chapter. The algorithm was also tested against one additional improvement which we propose in this chapter. In

particular, the use of the bonus, $B(s, a)$, can be discarded once the pair has been visited enough times. In our case it was $m_B = 5$. All parameter tuning experiments were performed with the standard version of MBIE and with $m_B = 5$ and for each case the best configuration was selected for comparisons.

- BEB (Kolter & Ng 2009): As in the previous case, the β and the m_B parameters were tuned in the same way. Additionally for fair comparisons with knowledge-based approaches this algorithm was enhanced with informative priors based on the AO model knowledge, BEB-AO, and the FSA model knowledge, BEB-FSA. In the first case, $n(s, a) = 1$ for all effects of action a, and $n(s, a) = 3$ when FSA knowledge is available.
- Greedy Optimistic (GO): This algorithm when used without domain specific knowledge corresponds to BEB and MBIE with $\beta = 0$. It can also be considered as a special case of Asynchronous Real Time Dynamic Programming (ARTDP) when used with greedy optimistic exploration (Barto et al. 1995). This is not a PAC-MDP algorithm. In our experiments GO-AO (GO-FSA) corresponds to the BEB version of this algorithm with AO (FSA) knowledge.

All algorithms for which knowledge about ϱ is relevant were tested with and without this knowledge. When ϱ is known, its value is 0.8 in our case (because it is the highest possible p_i in the tested domain), otherwise its default value of 1 is used. In all graphs all evaluations were computed for 10 runs of all algorithms. The cumulative score of each algorithm is reported as a function of the number of learning episodes. The error bars of the standard error of the mean (SEM) are also presented (Cohen 1995).

6.7 Results

The goal of the first series of experiments is to compare our Rmax-based approaches (i.e. Rmax-AO and Rmax-FSA) with basic Rmax and related reward shaping algorithms when they use the same knowledge. Rmax is the most 'cautious' PAC-MDP algorithm in its exploration strategy and our proposed technique should be especially suitable for Rmax. Figure 6.1 shows results with AO knowledge and $\varrho = 1$. The standard Rmax is the slowest to learn. Reward shaping obtains a better learning ratio which is further improved by a more accurate heuristic in RS(Manhattan). RS(Manhattan)-AO is better initially however it does not yield better results than pure RS(Manhattan). This is due to the fact that AO knowledge does not give almost any differentiation of initial Q-values in this case. Our approach, Rmax-AO, showed the best performance. The fact that its performance is better than learning with reward shaping can be explained as follows: In the case of informative models (AO in this experiment and FSA below), *knowledge is injected into our Rmax-AO algorithm starting from the very early stages of learning when all state-action pairs are still unknown. Reward shaping yields improvements only when planning takes place, that is, the improvement has an impact only on those state-action pairs which are known*. Figure 6.2 shows the same experiment with $\varrho = 0.8$. Results of all knowledge-based al-

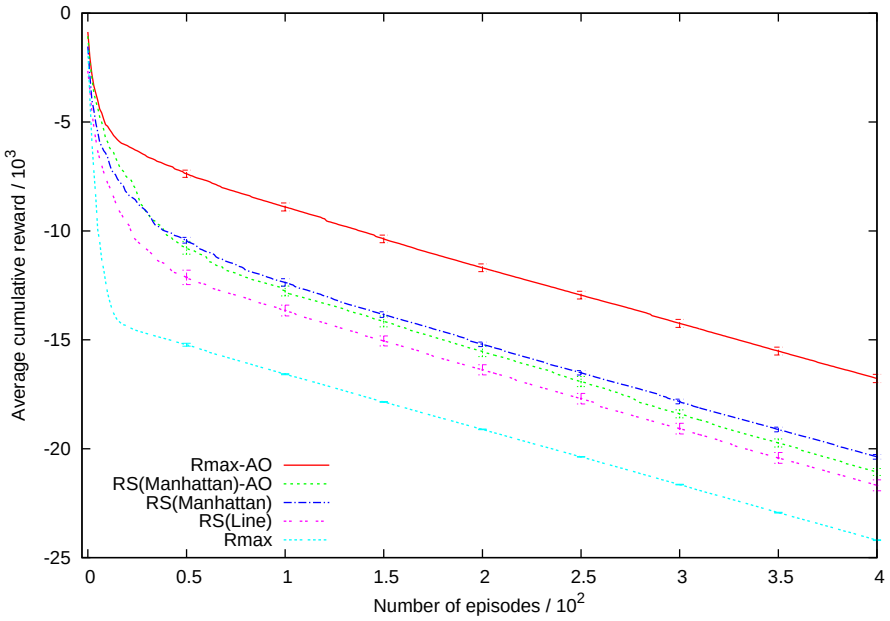


Figure 6.1: AO knowledge and $\varrho = 1$.

gorithms are better than in Figure 6.1 and Rmax-AO is still the best. RS(Manhattan)-AO showed slightly better initial learning than RS(Manhattan).

Results with FSA knowledge are shown in Figures 6.3 and 6.4. The improvement of the tested algorithms is similar as in the case of AO knowledge. This time however the reward shaping with FSA knowledge, RS(Manhattan)-FSA, is much better than pure reward shaping RS(Manhattan). In this case, the FSA knowledge yields very good differentiation of initial Q-values, because only one outcome for each action is considered and, in effect, in most cases there is exactly one state-action pair with $\max Q(s, a)$ in a given state. The use of $\varrho = 0.8$ led to further improvement of RS(Manhattan). Though other algorithms improve with the use of FSA knowledge, our Rmax-FSA algorithm was the most efficient in all cases.

A crucial outcome of the experiments discussed so far is the encouraging performance of our Rmax-based technique (Rmax-AO and Rmax-FSA). The overall advantage of the Rmax algorithm, in general, is that there is only one parameter m and the setting of this parameter is rather straightforward: the higher the value of m the more accurate the estimated model is (where this accuracy is formally specified by the Hoeffding bound introduced in Section 6.2). In Rmax the planning step is also performed at easily identified milestones (i.e. after a new state-action pair becomes known) whereas MBIE and BEB, which are evaluated below, require constant replanning. In the presented results, the Rmax algorithm worked very well with all kinds of knowledge, and reached the same best asymptotic performance in the long term. Obtained results were stable and appropriately improved by provided knowledge.

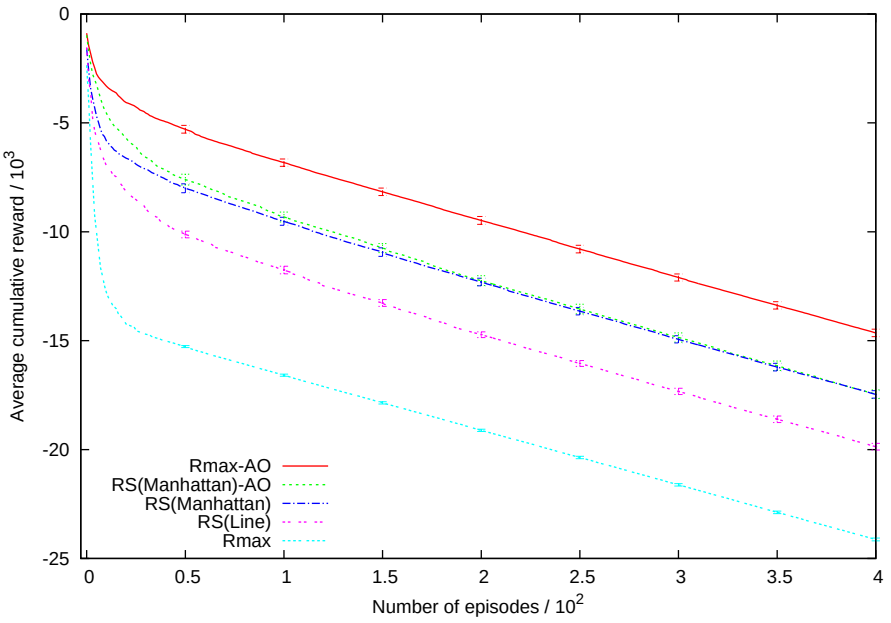


Figure 6.2: AO knowledge and $\varrho = 0.8$.

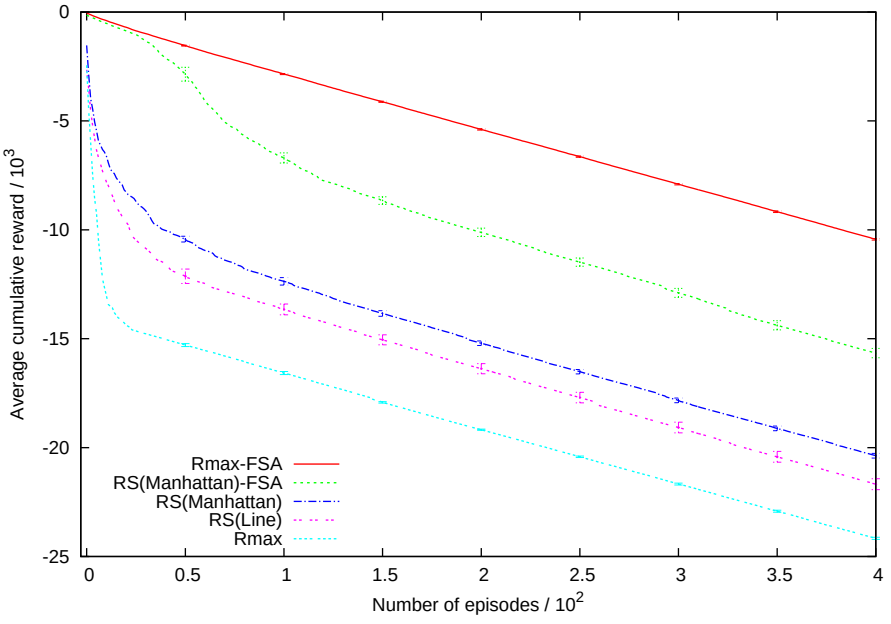


Figure 6.3: FSA knowledge and $\varrho = 1$.

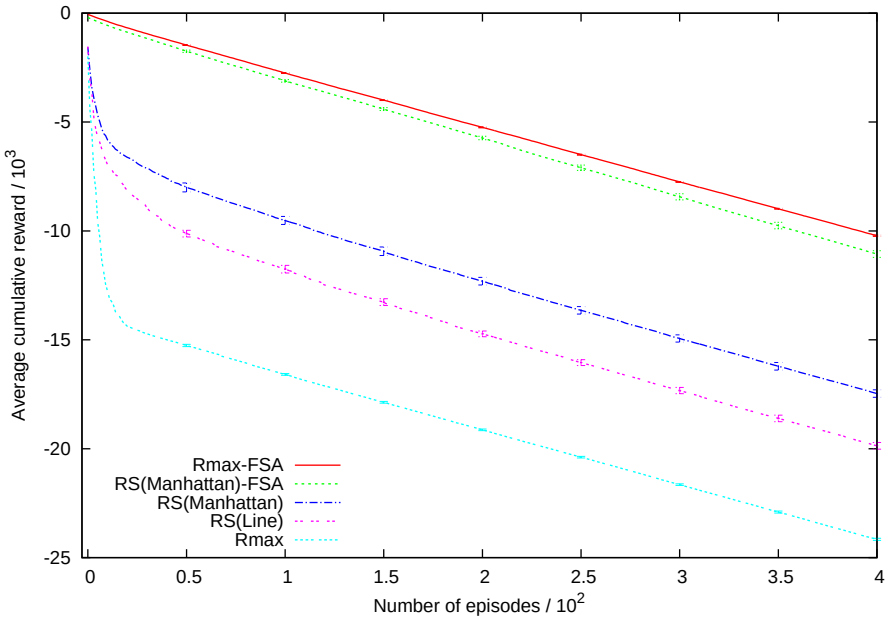


Figure 6.4: FSA knowledge and $\varrho = 0.8$.

Our technique introduced in this chapter can also be applied to the MBIE-EB algorithm. The next series of experiments is to compare the performance of Rmax and MBIE-EB when these two algorithms apply our technique to use knowledge and to compare them with the BEB algorithm. MBIE-EB and BEB are relatively new algorithms and there are no systematic evaluations and comparisons neither between these two approaches nor against Rmax. Thus, before doing a final analysis of these algorithms with our extension to incorporate domain knowledge, we are evaluating the performance of basic Rmax, MBIE-EB and BEB on our test domain (see Figure 6.5). With the best parameter configurations for BEB ($\beta = 0.3$ and $m_B = 5$) and MBIE-EB ($\beta = 0.4$ and $m_B = 5$) these two algorithms learned significantly faster than Rmax (BEB faster than MBIE-EB but the difference was not statistically significant), but were not able to reach the same asymptotic convergence as Rmax. The GO algorithm, though performing well initially, obtained the worst asymptotic performance. In comparisons in (Kolter & Ng 2009) both BEB and MBIE-EB do not converge to the optimum as well, but there are no results on Rmax in that comparison. Our results indicate that Rmax is very competitive in terms of finding the optimal solution.

Figures 6.6 and 6.7 present the target comparison of BEB, MBIE-EB, GO and Rmax algorithms when they have access to the same knowledge, i.e. AO or FSA knowledge. Results on basic Rmax are also included for reference. As already mentioned, AO knowledge is generally weaker than FSA knowledge because it leads to more optimistic and thus less informative MDP models. This fact is reflected in the performance of Rmax-AO in Figure 6.6. Rmax is the most

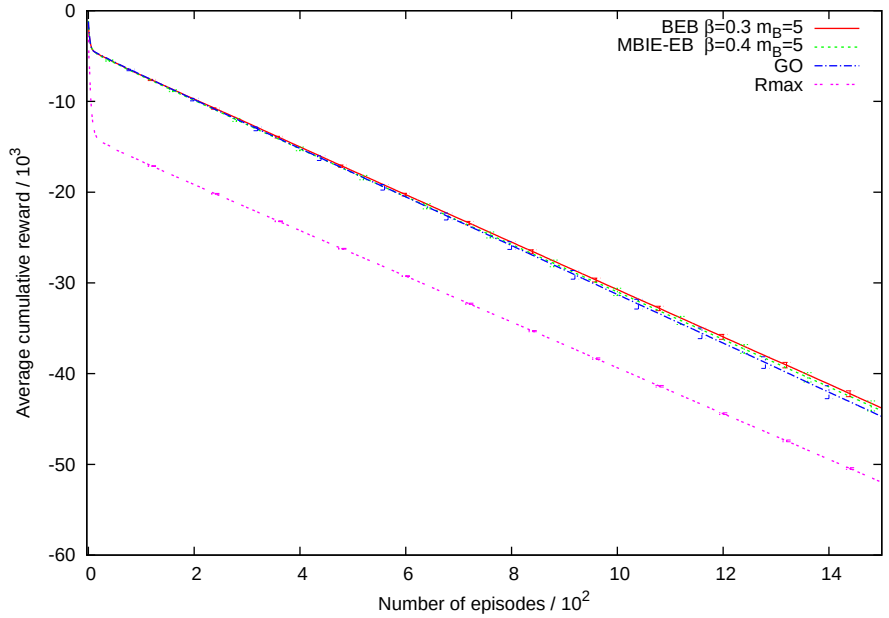


Figure 6.5: Default versions of tested algorithms without AO and FSA knowledge.

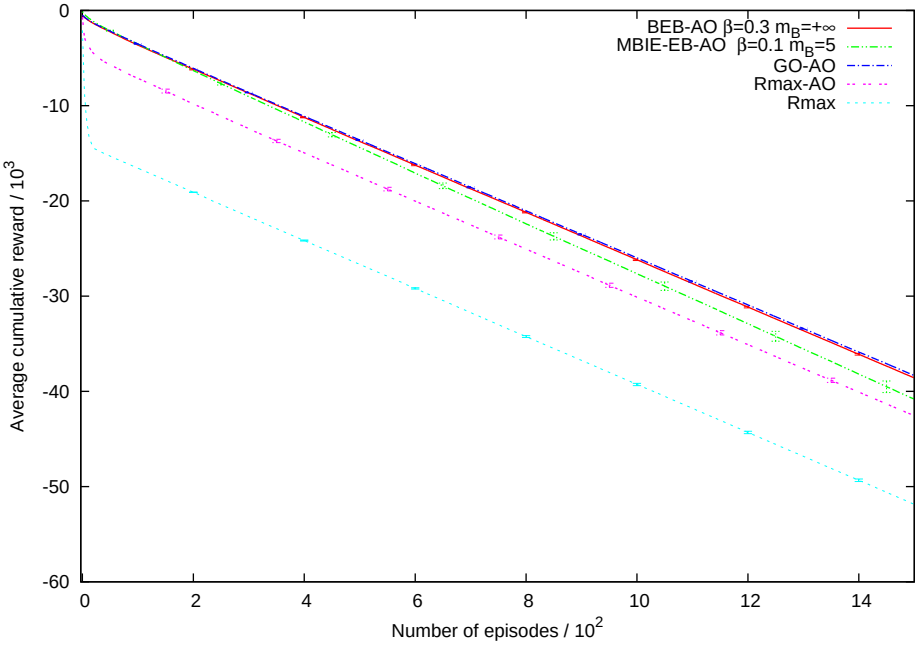


Figure 6.6: AO knowledge and $\varrho = 0.8$.

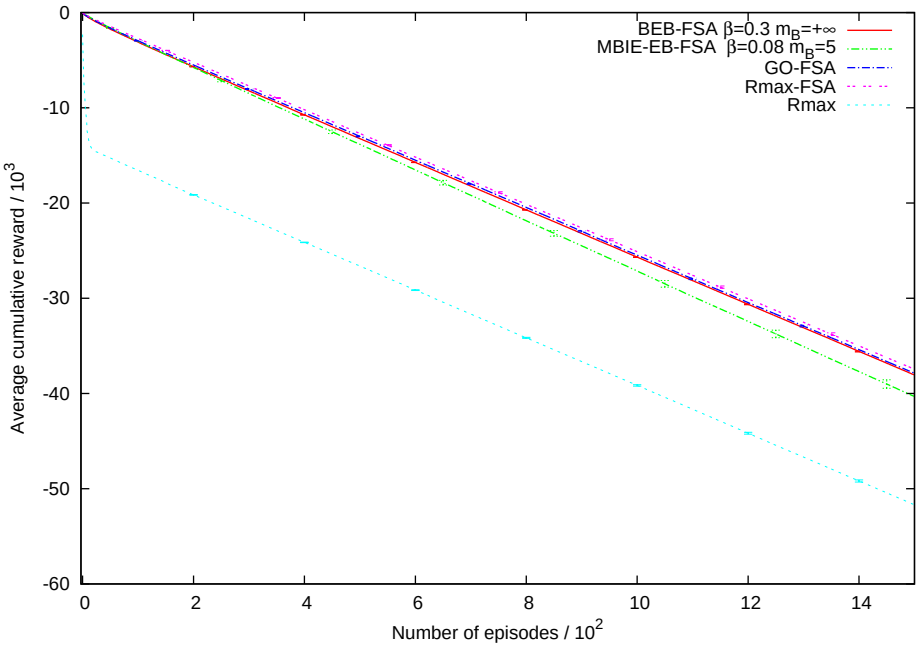


Figure 6.7: FSA knowledge and $\varrho = 0.8$.

'cautious' (i.e. it exploits less and spends more time on systematic exploration) and when using less informative knowledge it still has to explore more than other greedier algorithms. The best performance was due to BEB and GO. The use of knowledge allowed these two algorithms to reach an asymptotic result as good as Rmax. When comparing all algorithms with the use of FSA knowledge (Figure 6.7), this knowledge yields a much more informative model for the Rmax-FSA algorithm and its performance is as good (slightly better in this experiment) as BEB and GO. The MBIE-EB algorithm in this case as well reaches lower asymptotic performance (we present here the results with the best configuration of the parameters of this algorithm). These results are very promising for our extension because the Rmax algorithm is PAC-MDP, it is easy to tune, and it can reach the learning speed of much greedier learning algorithms which are not PAC-MDP. The last claim can be explained as follows: Due to its rigorous requirements, Rmax has to explore more than other greedier non-PAC-MDP algorithms like BEB, which means that it learns more slowly in many domains as is the case in our experiments. This chapter shows that a competitive performance can be obtained under all rigorous requirements of Rmax when it is used with our knowledge-based extension. Another practical remark is that FSA knowledge should be preferred over AO knowledge when our approach is used with the Rmax algorithm.

6.8 Summary and Discussion

Exploration-exploitation is the crucial challenge for autonomous agents which learn from environment feedback using reinforcement learning. PAC-MDP algorithms are particularly effective in practice, and interesting from an analytical point of view because they approach the exploration-exploitation problem in a way which guarantees that with high probability the algorithm performs near optimally for all but a polynomial number of steps. The performance of these algorithms can be further improved by incorporating domain knowledge to guide the learning process. The lack of such methods was shown to be a weak point of PAC-MDP algorithms and alternative non-PAC-MDP methods were proposed and shown to be competitive (Kolter & Ng 2009). In this chapter we propose a framework to use partial knowledge about effects of actions in a theoretically well-founded way. The contribution of this chapter can be summarised as follows:

- With the use of a symbolic specification of the MDP action in the PPDDL formulation, potentially available domain knowledge was distinguished and it was shown how to use this knowledge with PAC-MDP algorithms in a way which preserves theoretical properties of these algorithms.
- The empirical evaluation shows that our proposed method is more efficient than reward shaping which represents an alternative approach to incorporate background knowledge. Reward shaping requires an admissible heuristic which has to be designed manually but, in domains represented symbolically via PPDDL, it is difficult to design such admissible heuristics. Our solution uses only local action knowledge and can be applied when such heuristics cannot be designed or when those designed are not accurate. Our results show that even if such heuristics exist, our approach can be more efficient. Informally it can be argued that, in most cases our approach will be better than reward shaping because in our algorithm domain knowledge is used when state action pairs are still unknown. Knowledge injected via reward shaping is used only with known state-action pairs. This remark applies only to the situation when PAC-MDP algorithms are considered, when our method has access to AO or FSA knowledge, and when reward shaping is based on admissible heuristics such as the Manhattan or straight-line distance. We do not aim at discrediting reward shaping for the general case here because this result applies to a particular family of algorithms. In this comparison, this chapter gives also a good insight into the significance of different kinds of knowledge on the learning performance of various PAC-MDP algorithms.
- Our solution is also very competitive when compared with the Bayesian Exploration Bonus (BEB) algorithm. BEB is not PAC-MDP, however it can exploit domain knowledge via informative priors. We show how to use the same kind of knowledge in the PAC-MDP framework in a way which preserves all theoretical guarantees of PAC-MDP learning.

- The presented results indicate also that FSA knowledge leads to more informative admissible models and should be preferred to AO knowledge when applied to PAC-MDP algorithms such as Rmax.

This work was motivated by our goals of advancing further the use of symbolic representations (e.g., PPDDL) in RL. The technique presented in this chapter will be considered in our future work on such representations. Applicability of our approach is relatively straightforward in domains with a PPDDL description. This representation is behind the theoretical explanation of our solution and exists in many practical RL/planning domains (Ghallab et al. 2004; Russell & Norvig 2002). In this chapter, we focused experiments on the Maze domain because it allowed for detailed comparison with reward shaping approaches, which rely on admissible heuristics. For the Maze domain, we could design such heuristics and obtain detailed comparisons. PPDDL search spaces are massively broader than those in mazes, and we expect more significant improvements on these kinds of problems in the future work. Additionally, it is extremely difficult to design admissible heuristics for such domains, so reward shaping cannot guarantee PAC-MDP properties. Our solution is proven to be PAC-MDP for this broad family of domains.

As shown in this chapter, AO and FSA knowledge displayed encouraging speed-up of PAC-MDP learning. This kind of knowledge is directly based on the PPDDL representation, therefore it is easy to acquire and understand. It would be interesting to see if in future work it could be shown that different (either more general or more specific) types of domain knowledge would meet requirements of PAC-MDP learning. This could be, e.g., 'feature-based heuristics' which indicate that certain actions are more promising than other actions.

As pointed out in the empirical part of this chapter, Rmax, MBIE and BEB algorithms have not been systematically analysed in the existing literature (even in their basic form without the use of special domain knowledge), and from our experience we conjecture that such an analysis could be significant and represents an interesting direction of future work. This could also lead to the theoretical advancement in the understanding of these algorithms and their relationships and properties.

CHAPTER 7

Analysis of Exploration with Eligibility Traces

In this chapter, the robustness and exploration efficiency of SARSA(λ), the reinforcement learning algorithm with eligibility traces, is analysed. The type of reward function and the initialisation of the Q-table are taken into account in this analysis because they naturally influence the nature of the exploration when exploration is based on the current content of the Q-table, i.e. on the current policy. Existing analyses of eligibility traces in the literature focus on the predictive error of the value function after a small number of steps (Singh & Sutton 1996; Sutton & Barto 1998) and are also in most cases limited to variations of Markov Reward Processes (Sutton 1988; Sutton & Singh 1994). Markov Reward Processes (MRPs) can be seen as MDPs with a fixed policy (see Szepesvari 2009 for a formal definition of MRPs) in which the problem of action selection does not exist. MRPs do not consider actions, as is also the case in Markov chains, and the application of eligibility traces helps in learning values of states under a given fixed transition function of the corresponding Markov chain (the prediction problem). In the full MDP case, exploration has to be also dealt with because the algorithm decides which action to execute. This work investigates how the exploration-exploitation trade-off is balanced when learning with eligibility traces of different length (in contrast to the prediction problem in MRPs, the full MDP scenario is considered and the quality of the policy is compared instead of the error of the value function estimation). The analysis is focused on both the initial learning rate and the long term convergence which also have been neglected in empirical evaluations of eligibility traces in existing literature. The additional contribution of our analysis is that for the first time different types of reward function and the initialisation of the Q-table are considered in this context. It is important to investigate their influence because they naturally determine the resulting exploration strategy. Also most of the

empirical analyses or extensions to eligibility traces in the literature have focused on one type of the reward function at a time. In some cases, the step reward (R_s) was evaluated (Cichosz 1995; Framling 2007; Leng et al. 2009), and in others the final goal reward (R_g) (Preux 2002; Wyatt et al. 1999; Cichosz 1996; Zhu & Levinson 2002). We analyse two general types of rewards (final goal, R_g, and step rewards, R_s) and show situations and identify certain generalisations when long traces, i.e. high values of λ, can lead to suboptimal solutions, and when they are more likely to speed up the convergence. Problems are identified and discussed. Specifically, obtained results show that exploration of SARSA(λ) learning is sensitive to different types of reward function and the initialisation of the Q-table. In some cases the asymptotic performance can be significantly reduced. The overall analysis is supported and significantly enhanced by a detailed evaluation of eligibility traces under different learning rates, α, and exploration rates, ϵ, (in the ϵ-greedy exploration) which was not included in our previous presentation of this work in (Grześ & Kudenko 2008b).

7.1 Introduction

In contrast to supervised learning, RL agents are not given instructive feedback on what is the best decision in a particular situation. Agents have to learn which action should be chosen in a given state using numerical feedback which is named the reward function. This type of learning suffers from the *temporal credit assignment* problem, that is, the problem of determining which part of the behaviour deserves the reward (Sutton 1984). To tackle this problem, the iterative approach to RL applies backpropagation of the value function in the state space. Because this is a delayed, iterative approach, it usually leads to a slow convergence, especially when the state space is huge. In fact, the state space grows exponentially with each variable added to the encoding of the environment when the Markov property needs to be preserved (Puterman 1994).

7.1.1 Eligibility Traces

The concept of eligibility traces in reinforcement learning represents one of the methods of dealing more efficiently with the temporal credit assignment problem and with non-Markovian state spaces (Loch & Singh 1998; Sutton 1988; Pérez-Uribe & Sanchez 1999). In standard temporal difference learning, the backpropagation is performed on only one state at a time, that is, the temporal difference:

$$\delta = r + \gamma Q(s', a') - Q(s, a) \tag{7.1}$$

is used to update only state s:

$$Q(s, a) = Q(s, a) + \alpha\delta, \tag{7.2}$$

where α is the learning rate, γ the MDP discount factor, r an immediate reward, s' is the current state, a' an action to be taken in state s', s the previous state, and a the action taken in state s. The particular computation of the temporal difference in Equation 7.1 is according to the

SARSA algorithm (Sutton & Barto 1998). The idea of eligibility traces is to propagate current temporal difference δ not only to state s but also to states which were recently visited (trace) and the measure of this recency is named eligibility. If we assume $e(s, a)$ to be the eligibility of pair (s, a), the SARSA update takes the form:

$$Q(s, a) = Q(s, a) + \alpha\delta e(s, a) \tag{7.3}$$

and is applied to all state-action pairs for each value of δ. When state s is the most recent state to be updated, eligibility for this state is set to one and after each time step it is reduced by the multiplicative factor $\lambda\gamma$ where λ controls how eligibility decays in time. SARSA with updates of this type is named SARSA(λ). For the clarity of presentation, the standard version of this algorithm with the aforementioned type of eligibility traces is shown in Algorithm 3. In this type of trace, the eligibility of the most recent state to be updated is reset to the value of one (Line 9 in Algorithm 3). These traces are named *replacing eligibility traces* and are used in this work since they were shown in the literature to work better than accumulating traces (Singh & Sutton 1996; Sutton & Barto 1998) and are generally used more often in the field.

The analysis of eligibility traces in this chapter is based on SARSA because (in contrast to Q-learning) it is an on-policy method and backpropagation can be performed along the entire trace of visited states. In this way, we try to avoid obscuration in our analysis and results which may be caused by problems of Q-learning when eligibility traces are used. A detailed discussion of this issue is due to Sutton & Barto (1998) and Peng & Williams (1996). The problem generally stems from the fact that Q-learning is an off-policy method, and the policy being learned need not be the same as the one used to select actions. The eligibility trace is governed by the latter one and it is not clear how to use it to update Q-values of the former policy. Specifically, if the last two actions were greedy actions according to the policy learned by Q-learning and the third action is a non-greedy one then the use of the last temporal difference (which corresponds to a non-greedy action) for updating two former Q-values has no relationship to Q-learning. This is the case in Q-learning even with ϵ-greedy exploration, because when a non-greedy action is executed this is already a non-policy action since it does not correspond to the current policy determined by the Q-table. Bruske et al. (1996) suggested, for example, that piecewise constancy of the policy leaves the traces valid in Q-learning for a given period of time.

7.1.2 Motivation

The motivation for this research, which comprises also the opening paragraph of this chapter, can be summarised as follows. Firstly, in the existing literature most evaluations of RL algorithms with different values of λ display results for a given, small number of learning episodes, e.g., 10 (Singh & Sutton 1996; Sutton & Barto 1998). Secondly, such experiments focus mostly on the prediction of the value function whereas exploration is not taken into consideration (prediction

Algorithm 3 The SARSA(λ) algorithm with replacing eligibility traces (Sutton & Barto 1998; Singh & Sutton 1996).

1: Initialise $Q(s, a)$ arbitrarily and $e(s, a) = 0$ for all s, a
2: **repeat** {for each episode}
3: Initialise s
4: Choose a from s using policy derived from Q
5: **repeat** {for each step of episode}
6: Take action a, observe reward r, s'
7: Choose a' from s' using policy derived from Q
8: $\delta \leftarrow r + \gamma Q(s', a') - Q(s, a)$
9: $e(s, a) = 1$ {replacing traces}
10: **for all** s, a **do**
11: $Q(s, a) = Q(s, a) + \alpha\delta e(s, a)$
12: $e(s, a) \leftarrow \gamma\lambda e(s, a)$
13: **end for**
14: $s \leftarrow s'$, $a \leftarrow a'$
15: **until** state s is terminal
16: **until**

in Markov Reward Processes where there is no exploration due to lack of choice of actions). In this work, the special attention is paid to exploration. The λ parameter, the reward type, and the initialisation of the Q-table play significant role in the exploration strategy based on current Q-values when actions are selected using these Q-values as in Line 7 in Algorithm 3. Additionally, sensitivity of RL with $\lambda > 0$ has never been analysed under different types of the reward function, i.e. whether the positive reward is given only upon reaching the goal state, R_g, or negative penalty, R_s, is given after each step. If for all pairs (s, a) zero is initially assigned to $Q(s, a)$, the modification of the reward type leads to a straightforward change of the character of exploration - i.e. optimistic or pessimistic (see Section 2.2). This leads to an additional dimension of the analysis, i.e. different initialisations for each reward type can be considered. As shown in our results, the fact whether the reward function is sparse (R_g) or dense (R_s), and whether exploration is optimistic or not, has significant influence on the performance when used with different values of λ. The influence of the learning rate, α, and the exploration rate, ϵ, on the tested configurations is also considered, evaluated and discussed. The thorough analysis of this chapter yields an interesting and novel insight into the issue of the influence of eligibility traces on exploration in model-free reinforcement learning.

7.2 Experimental Design

In this section, the evaluated algorithm with its parameters and the problem domain are discussed. Some characteristics of the experimental domain are also identified. We use these characteristics in our attempt to generalise our findings to other domains which have similar properties.

7.2.1 Parameters

In our experiments eligibility traces are implemented in an efficient way by truncating them when the eligibility becomes negligible (Cichosz 1995; Sutton & Barto 1998). Specifically, the trace of N most recently visited state-action pairs is stored where $(\lambda\gamma)^N \geq 10^{-9}$. The value of eligibility is evaluated as: $e(s,a) = (\lambda\gamma)^{t(s,a)}$ where $t(s,a)$ is the number of time steps which elapsed since pair (s,a) has been added to the trace. In this way, for the most recent pair $t(s,a) = 0$ which makes $e(s,a) = 1$. It means that in accordance with the previous section, the replacing eligibility traces are used in our implementation (Line 9 in Algorithm 3).

ϵ-greedy exploration is used. It is an instance of value function-based exploration strategies which are based on the current content of the Q-table. Since eligibility traces influence the content of the Q-table significantly, it enforces their influence on the resulting exploration. Thus, ϵ-greedy exploration allows us to gain better insight into the influence of the investigated characteristics of RL. Another type of action selection which is based on the current policy is Boltzmann exploration (Sutton & Barto 1998). This exploration strategy uses Boltzmann distribution for probabilistic action selection. The Boltzmann distribution is a function from values of actions to probabilities of selecting those actions. We are focusing in our analysis on the ϵ-greedy exploration because it is easier to apply than Boltzmann exploration and also due to our detailed and rigorous analysis of this chapter, it would be infeasible to report results for another action selection method.

Our analysis of eligibility traces corresponds also to the analysis of the impact of λ on different exploration strategies which are different with regard to the initialisation of the Q-table. We are using ϵ-greedy exploration here, but it has diametrically opposed properties when used with optimistic or pessimistic initialisation. Thus, we are interested here in ϵ-greedy exploration with different initialisations of the Q-table (pessimistic and optimistic), and this is the initialisation which distinguishes between exploration strategies which we are taking into account. Experiments with a range of values of the λ parameter are reported. A high value of the learning rate, α, can have a negative impact on the asymptotic performance with higher values of λ as indicated by Cichosz (1995). For this reason in our leading experimental settings, α starts with a relatively small 0.1 value in the first episode and is further decreased linearly to 0.01 in the last episode. This is a rather standard value of this parameter which is commonly used in the reinforcement learning literature (Sutton & Barto 1998). The value of 0.1 was also used in the famous practical application of temporal difference learning: TD-gammon (Tesauro 1992). This relatively small value should not obscure our main results. Such a low value of the learning rate is also advised (Sutton & Barto 1998) when the environment is stochastic like in our case because the agent needs more time to adjust to stochastic effects of actions. For a more detailed and deeper analysis, a comparison with various values of the learning rate, α, was also undertaken in additional experiments and this parameter is also considered as one of the dimensions of this research.

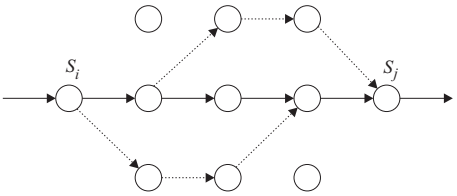


Figure 7.1: Domain properties.

The ϵ-greedy exploration starts with $\epsilon = 0.3$, and this value is also decreased linearly so that it reaches 0.01 in the last episode. We are focusing in this analysis on exploration; in particular on exploration which is based on the current content of the Q-table and which diverts with some probability ϵ from the policy determined by the Q-table. Starting from the value of $\epsilon = 0.3$ guarantees (as mentioned in the literature Sutton & Barto 1998) a reasonable amount of explorative actions in early episodes and also staying close to the current policy. This kind of exploration generally aims at being relatively close to the best known solution. This approach naturally fits the paradigm of self improving on-line learning but is also mentioned to be one of the major limitations of RL algorithms (Jong & Stone 2007). For more detailed analysis and better understanding of processes which govern results which are obtained in this work, a detailed experimentation was also conducted with different values of ϵ and this parameter also represents one of the dimensions of our analysis.

The discount factor is $\gamma = 0.99$ for the goal-based reward, R_g, and $\gamma = 1.0$ when the step reward, R_s, is given. Goal-based reward is $R_g = 100$, i.e. all rewards are 0 and only upon entering the goal state 100 is given[1]. In the second configuration with the step reward, the reward $R_s = -1$ is given after each step.

All runs of evaluated configurations were repeated 10 times and average results are presented in graphs where in most cases the reward for getting to the goal state from the start state is used. For the goal-based reward this is the discounted reward, and for the step penalty reward this is the sum of penalties to reach the goal state from the start state. In some cases, more legible plots were obtained by presenting the overall cumulative reward instead of the episode reward. In each graph, a number of curves for different values of λ is shown. To make the interpretation of these graphs easier, in some cases, selected curves are plotted with a thicker line style.

7.2.2 Experimental Domain

In this experimental analysis, we are interested in the class of reinforcement learning problems in which certain properties can be identified. The problem of learning a policy to reach goal state

[1]For this reason, even though the task is episodic the discount factor γ has to be less than 1. Otherwise, all Q-values would converge to the value 100 which is the value of the goal reward. With $\gamma < 1$ they are appropriately discounted allowing learning a policy for our goal-based problem. Thus, the values of this parameter were selected in order to ensure convergence with a given reward type. This parameter is a part of the MDP specification and should not be considered to be the parameter of the algorithm but rather a part of domain description (see how MDPs were defined in Section 2.1.3).

g from any starting state s_0 in a stochastic environment is considered. The trajectory to reach a goal state when following policy π can be $\xi_\pi = (s_0, s_1, ..., g)$. We point out here that in some domains, like for example navigation problems, for considerably many pairs of states (s_i, s_j) where $s_i \in \xi_\pi$ and $s_j \in \xi_\pi$ such that $i < j$, the path between these states has many opportunities to be suboptimal (see Figure 7.1). It means that the trajectory ξ_π may not be connecting states s_i and s_j via the shortest path. This raises the possibility of suboptimal solutions, e.g., in navigation problems. As an example of such tasks, a maze-based domain is investigated in our work. A specific example of a maze task in which this problem does not exist is the random walk domain (see Section 5.4.1 and Figure 5.1 for details) which is also used in the final part of this chapter.

A policy which represents the solution to the RL task can be obtained from different definitions of the reward. Specifically, the policy for the stochastic shortest path problem can lead to the same optimal behaviour when learning takes place either according to the discounted goal reward or penalties given for each step. Even though these types of rewards are very different, they determine the same optimal behaviour: navigating to the goal state under the constraints of transition probabilities of the underlying MDP. The particular feature of this task is that regardless of which reward is used, the reward sets only one objective, that is, the behaviour of reaching the goal state as fast as possible. Thus, the following categories of RL tasks can be distinguished:

- the reward determines one objective regardless of which type of reward is used (e.g., the goal-based or penalty-based reward in stochastic shortest path problems),

- the reward determines more than one objective; one reward can encourage the agent to reach a particular goal state with a positive reward (winning a game) subject to constraints imposed by another reward which penalises or rewards for some other behaviour (an objective of avoiding a particular situation, for example, losing a game).

In this chapter, a thorough analysis of an instance of the first category is carried out. Our attempt to classify RL domains in this way is to define the scope of our analysis which may help RL practitioners to determine which kind of domains our findings can be generalised directly to.

Experiments are performed on the domain which has been commonly used in the RL literature (Sutton 1990). This navigation maze task (S-maze) was originally shown in Figure 5.3 and described in Section 5.4.3. For the sake of the reader, it is repeated here in Figure 7.2.

7.3 Goal-based Rewards

In the first set of experiments, the goal-based reward, R_g, is evaluated. Firstly, different exploration strategies which result from a different initialisation of the Q-table are considered, and then additional experiments look into the influence of the learning rate, α, and the exploration rate, ϵ.

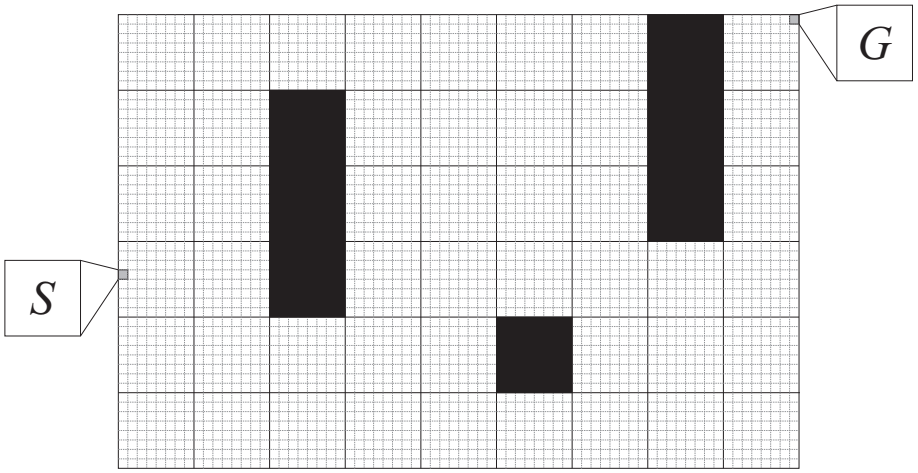


Figure 7.2: The stochastic navigation maze task - S-maze (Sutton & Barto 1998: Figure 9.5).

7.3.1 Pessimistic Exploration

Goal of Experiment: The analysis of λ under the goal-based reward function and pessimistic exploration.

Parameters: λ changes in the range $[0, 1]$, $\alpha = 0.1$, and $\epsilon = 0.3$.

Results: In this experiment, the goal-based reward is used with the default initialisation of the Q-table:

$$\forall s, a : Q(s, a) = 0. \tag{7.4}$$

This yields pessimistic exploration for this type of reward function. Results with a range of λ values are in Figure 7.3 (the top graph of this figure shows the first 10^3 episodes). It can be observed in Figure 7.3 that learning with $\lambda = 0$ has the largest delay in reaching the level of performance which is much earlier obtained with higher values of λ (therefore if the performance would be compared only at the very beginning of learning, the highest values of λ would be evaluated higher). However, an interesting insight into this learning process is shown by the asymptotic convergence when learning takes place up to 2.5×10^4 episodes. Higher values of λ, especially $\lambda > 0.5$, show worse asymptotic performance when compared with $\lambda = 0$. Even though the learning rate is relatively low ($\alpha = 0.1$), exploration which is based on the current content of the Q-table is significantly dominated by the suboptimal paths propagated by longer eligibility traces. This can be well observed in the trajectories for $\lambda = 0.9$ which are shown in Figure 7.4. A general shape of the trajectory which is followed after few dozens of episodes is reinforced and may stay up to the end of learning, yielding a final solution which is suboptimal.

The most destructive influence in this learning process was caused by $\lambda = 1.0$. To help explain this issue, Table 7.1 shows the length of the eligibility trace which corresponds to particular values of parameters. This table shows exact values of the trace length which rapidly grows

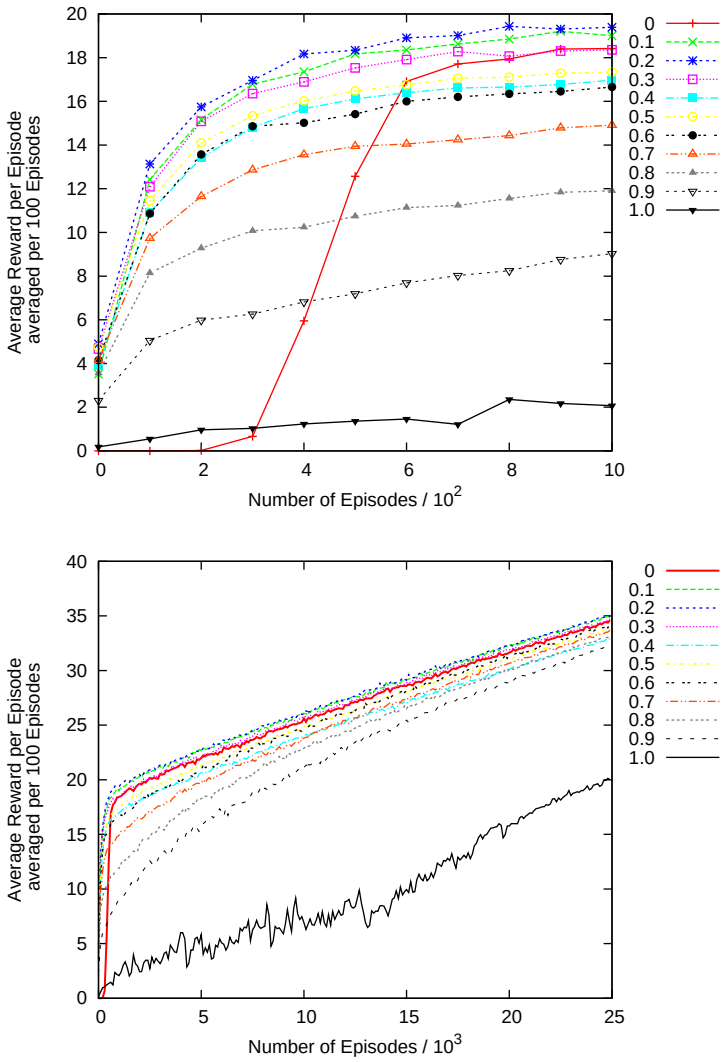


Figure 7.3: Results for the goal-based reward function with pessimistic initialisation. Each curve corresponds to a different value of λ. The top graph shows the first 10×10^2 episodes, and the bottom graph is for the whole period of learning, i.e. 25×10^3 episodes.

when λ approaches one. For lower values of λ, the length of the trace is of the same order of magnitude as the maximum path length in the domain (the maximum path length is around 100 in S-maze). For $\lambda > 0.9$ and 1.0 in particular, traces become significantly longer. Thus, the destructive influence of $\lambda = 1.0$ in our experiment can be attributed to the fact that the trace is in this case considerably longer than with other values of λ - especially 0.9. The comparison of values collected in Table 7.1 and our results in this experiment and experiments below suggest that the maximum path length in the domain should be taken into account when selecting the value of

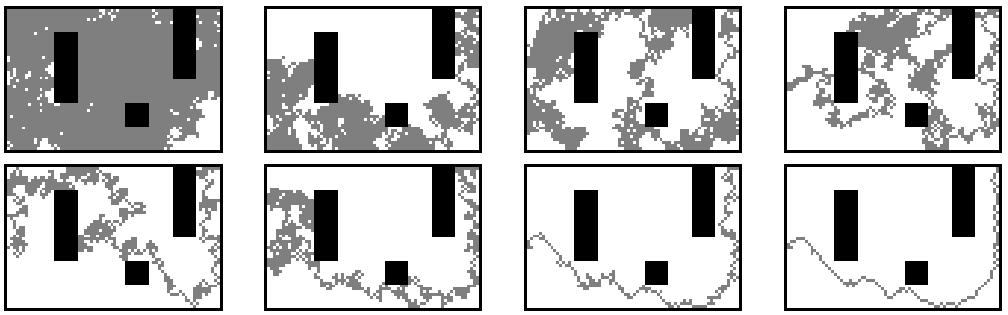


Figure 7.4: Trajectories during the learning process with the goal-based reward function, pessimistic initialisation and $\lambda = 0.9$. When counting figures from the left to the right, the following numbers of learning episodes correspond to these figures: 1, 3, 5, 7 in the first row and 13, 15, 1.2×10^3, 2.5×10^4 in the second row.

γ	λ	Trace Length	γ	λ	Trace Length
1.0	0.0	0	0.99	0.0	0
1.0	0.1	9	0.99	0.1	8
1.0	0.2	12	0.99	0.2	12
1.0	0.3	17	0.99	0.3	17
1.0	0.4	22	0.99	0.4	22
1.0	0.5	29	0.99	0.5	29
1.0	0.6	40	0.99	0.6	39
1.0	0.7	58	0.99	0.7	56
1.0	0.8	92	0.99	0.8	88
1.0	0.9	196	0.99	0.9	179
0.99	1.0	2061	0.99	1.0	2061

Table 7.1: The length of the eligibility trace which is computed according to the following function: $f(\gamma, \lambda) = \lfloor \log_{(\gamma\lambda)} 10^{-9} \rfloor$. Values shown in this table are domain independent since the length of the trace depends on γ and λ only.

the λ parameter. Because values of λ close to 1.0 yield significantly longer traces, a reasonable heuristic would be to select λ in such a way so that the length of the trace is comparable with the maximum path length in the domain (more discussion on this issue will be in the final part of this chapter).

7.3.2 Optimistic Exploration

Goal of Experiment: The analysis of λ under the goal-based reward function and optimistic exploration.

Parameters: λ changes in the range $[0, 1]$, $\alpha = 0.1$, and $\epsilon = 0.3$.

Results: Next, a different approach to initialisation was applied and here the aim was to check

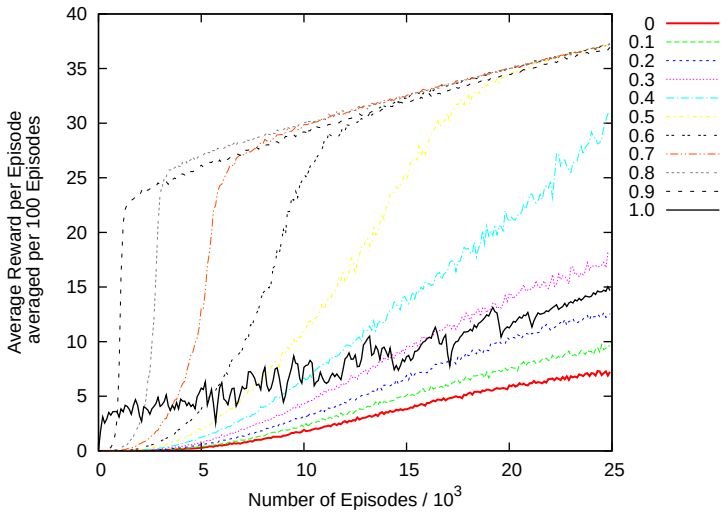


Figure 7.5: Results for the goal-based reward type with an optimistic initialisation. Each curve corresponds to a different value of λ.

the convergence of learning with a goal-based reward function and an optimistic initialisation. In this case, the Q-table was initialised in the following way:

$$\forall s, a : Q(s, a) = 100, \forall g \in \mathbb{G}, a : Q(g, a) = 0,$$

where $\mathbb{G}$ is the set of goal states. The highest possible value which is equal to $R_g = 100$ was assigned to all non-goal Q-values. The learning curve for this case is in Figure 7.5. Here, higher values of λ lead to a better long term convergence. The difference is observed for $\lambda = 0.9$ and 1.0. Even though the initial convergence is the most rapid, the asymptotic performance is decreased. In the case of $\lambda = 0.9$, it is only a slight decrease whereas $\lambda = 1.0$ fails more significantly and explanation from the previous subsection, which is based on Table 7.1 applies here as well. As shown in the next section in Figure 7.15, $\lambda = 0.9$ does not have problems with asymptotic convergence when the step reward function is used. We conjecture that this difference is caused by the different reward type. Even though the optimism under uncertainty is used in both cases, the different character of rewards (R_g or R_s) can influence the propagation of the value function differently. And based on these results, we can emphasise that *the type of reward used in the domain is an important factor in the analysis of eligibility traces*.

7.3.3 Resetting the Trace

Goal of Experiment: The goal of this experiment is to check whether clearing the trace after each episode will influence results obtained in two sections above, 7.3.1 and 7.3.2.

Parameters: All parameters are as in Sections 7.3.1 and 7.3.2. The only difference is that now

the eligibility trace is cleared before every new episode starts.

Results: There is one more issue about eligibility traces which has not been discussed much in the literature. Our evaluation in this work is based on the standard version of the SARSA(λ) algorithm which is shown in Algorithm 3. One of the properties of this algorithm is that the eligibility trace is initialised before learning takes place and eligibility is transferred between episodes during learning. The question which can be asked is whether it would be profitable to reset the trace - i.e. re-initialise the whole trace with values of 0 - at the beginning of each new episode. It would require inserting $e(s, a) = 0$ and executing it for all state-action pairs directly after Line 2 in Algorithm 3. This simple idea was used, for example, in (Leng et al. 2009; Stone & Sutton 2001; Stone et al. 2005). Again, in order to have better insight into the processes which are analysed in this chapter, this modification to the standard SARSA(λ) is also evaluated. Thus for experiments of this subsection, we modify the SARSA(λ) algorithm and re-initialise the eligibility trace with values of 0 for all state-action pairs. The modified algorithm was evaluated with pessimistic and optimistic exploration for the goal-based reward function. The result with the pessimistic initialisation does not change when comparing with the standard version which is in Figure 7.3. The only difference is that the performance with $\lambda = 1.0$ is slightly better in this case, but is still much lower than the performance of lower values of this parameter. The same applies to the case with optimistic exploration. The result is like in Figure 7.5 and $\lambda = 1.0$ is slightly better as well. The fact that $\lambda = 1.0$ behaves better here is not surprising because the trace is relatively longer with this value of λ than in other cases (see Table 7.1) and resetting the trace reduces its size to those states which can be visited within one single episode. Because these results mirror results with our basic standard configuration, we do not test this property in other configurations of parameters.

7.3.4 Changing Learning Rates

Results discussed until now were based on our baseline values of two RL parameters: the learning rate, $\alpha = 0.1$, and the exploration rate, $\epsilon = 0.3$. In this section, the impact of the learning rate is analysed in detail. Specifically, experiments reported in Sections 7.3.1 and 7.3.2 are repeated here for the following values of α: 0.01, 0.05, 0.1 and $0.3 - 0.9$.

7.3.4.1 Pessimistic Exploration

Goal of Experiment: The goal of this experiment is to test λ with a range of learning rates α under pessimistic exploration and the goal-based reward function.

Parameters: λ changes in the range $[0, 0.9]$, α is in the range $[0.01, 0.9]$, and $\epsilon = 0.3$.

Results: With $\alpha = 0.01$ and 0.05, the higher λ the worse the asymptotic performance is (see Figure 7.6). When α increases, the difference in the performance of different λ values becomes smaller, as shown in Figure 7.3 for $\alpha = 0.1$. When α is further increased, learning with all values of λ starts converging to the same performance, though with $\alpha = 0.3$, $\lambda = 0$ is still the best. With $\alpha = 0.5$, $\lambda = 0.9$ performs substantially worse and curves of all smaller values of λ are

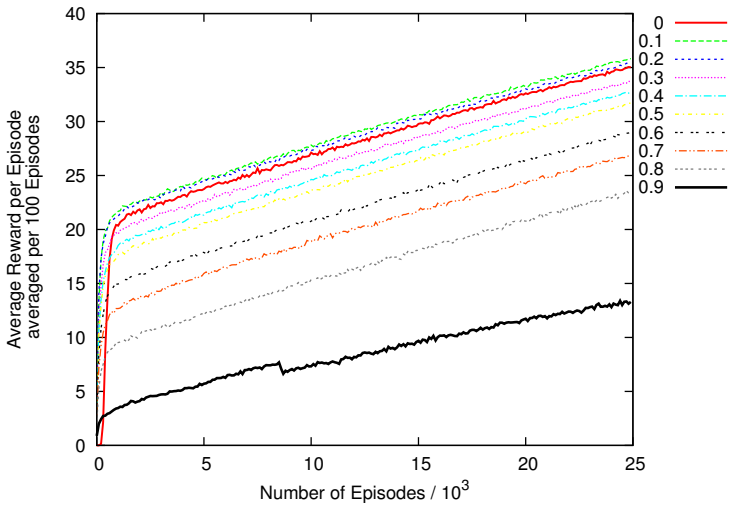


Figure 7.6: Results for the goal-based reward type with pessimistic initialisation and $\alpha = 0.01$. Each curve corresponds to a different value of λ.

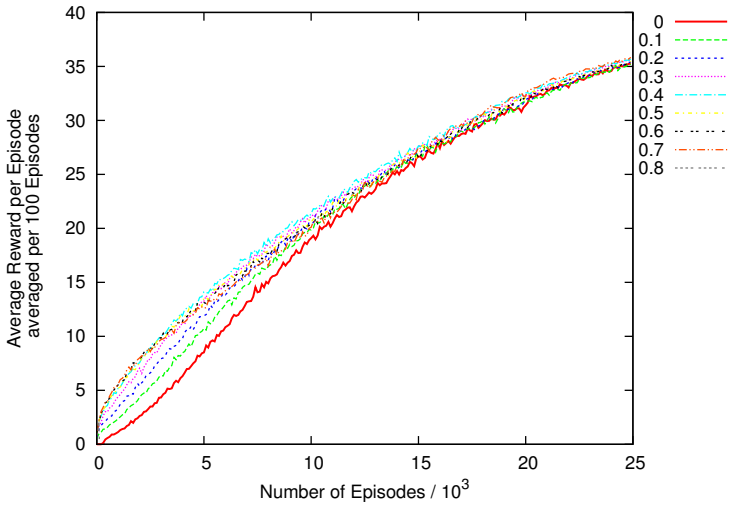


Figure 7.7: Results for the goal-based reward type with pessimistic initialisation and $\alpha = 0.9$. Each curve corresponds to a different value of λ.

very close. But, in this case, the best performance is due to $\lambda = 0.7$ and 0.8. When $\alpha = 0.7$ and 0.9, all curves are very close (see Figure 7.7) and $\lambda = 0.9$ does not converge in both cases (not presented in this graph), i.e. there were very long episodes and it was infeasible to wait until they finish. An important observation is that with higher α values, the best performance can be achieved with $\lambda > 0$.

The overall conclusion, which is important with regard to the direction of the analysis in this

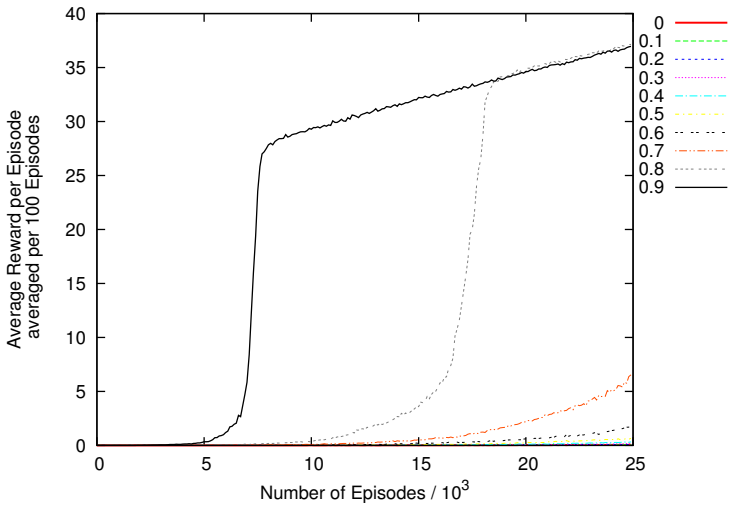


Figure 7.8: Results for the goal-based reward type with optimistic initialisation and $\alpha = 0.01$. Each curve corresponds to a different value of λ.

chapter, is that with the goal-based reward type, R_g, and pessimistic exploration high λ reduces the performance significantly with small α, and some values of λ (in particular 0.7 and 0.8) lead to a rather small improvement with high α.

7.3.4.2 Optimistic Exploration

Goal of Experiment: The goal of this experiment is to test λ with a range of learning rates α under optimistic exploration and the goal-based reward function.

Parameters: λ changes in the range $[0, 0.9]$, α is in the range $[0.01, 0.9]$, and $\epsilon = 0.3$.

Results: When learning with optimistic exploration and $\alpha = 0.01$ and 0.05, highest λ values lead to the most rapid improvement of the performance (see Figures 7.8 and 7.9). Other smaller values of $\lambda \leq 0.7$ were considerably slower. They need more time to get rid of optimism using their shorter traces. When α grows, all curves generally get closer and again highest λ values ($\lambda = 0.8$ and 0.9) also lose at a certain point. The combination of the long trace and a high learning rate lead to radical changes in the Q-table in each update in this case. Specifically, in Figure 7.10 there is a result with $\alpha = 0.9$ in which $\lambda = 0.8$ did not succeed in reaching the goal state in several runs (the curve is the average of those runs which were successful), and $\lambda = 0.9$ was not successful at all. The overall trend which appears in this configuration is that longer traces are generally better with different values of α because they help to propagate lower Q-values and get rid of optimism in the Q-table. In particular, $\lambda = 0$ led to the lowest performance with all tested values of α. The higher λ the better the performance was, except for the highest values of 0.8 and 0.9 in combination with highest values of α. In such a case, these two factors lead to rapid changes of Q-values, and resulting changes have negative impact on exploration.

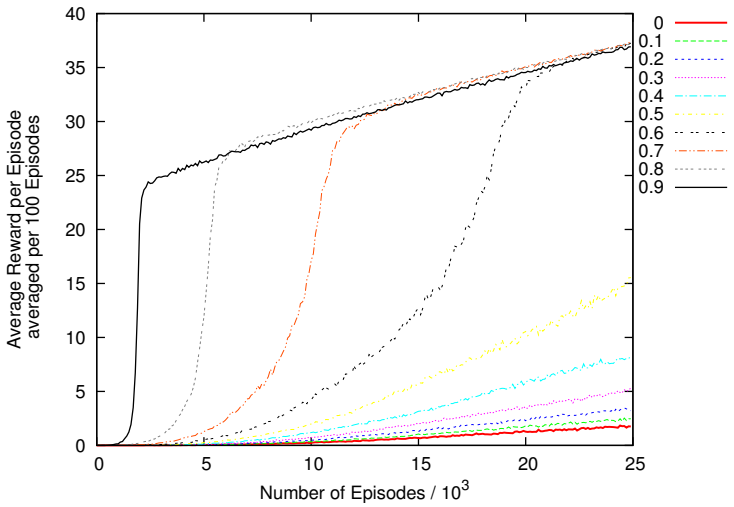


Figure 7.9: Results for the goal-based reward type with optimistic initialisation and $\alpha = 0.05$. Each curve corresponds to a different value of λ.

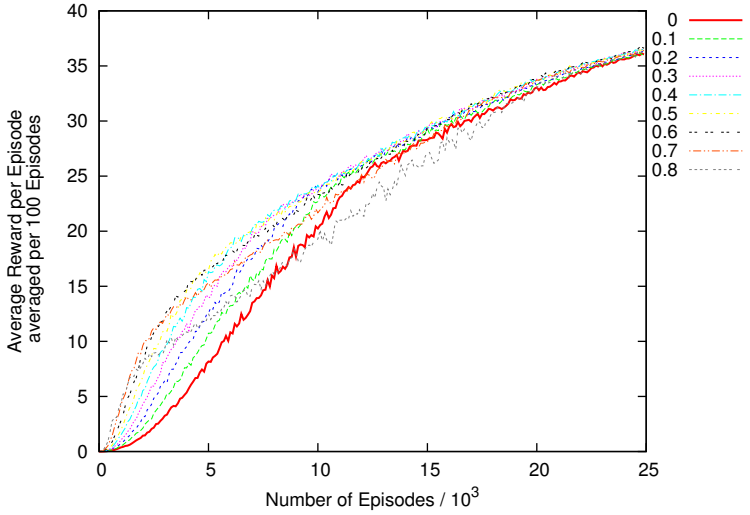


Figure 7.10: Results for the goal-based reward type with optimistic initialisation and $\alpha = 0.9$. Each curve corresponds to a different value of λ.

7.3.5 Changing Exploration Rates

In this section, another RL parameter is tested in order to gain better insight into the influence of eligibility traces on exploration. This time, the impact of the exploration rate is analysed in detail. Specifically, experiments reported in Sections 7.3.1 and 7.3.2 are repeated here for the following values of ϵ: 0.01, 0.05, 0.1 and $0.3 - 0.9$.

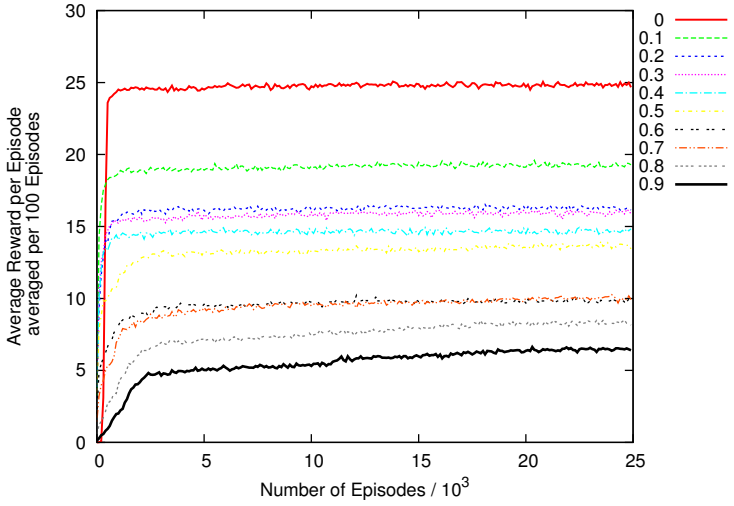


Figure 7.11: Results for the goal-based reward type with pessimistic initialisation and $\epsilon = 0.01$. Each curve corresponds to a different value of λ.

7.3.5.1 Pessimistic Exploration

Goal of Experiment: The goal of this experiment is to test λ with a range of exploration rates ϵ under pessimistic exploration and the goal-based reward function.

Parameters: λ changes in the range $[0, 0.9]$, ϵ is in the range $[0.01, 0.9]$, and $\alpha = 0.1$.

Results: The first graph for this case is in Figure 7.11. The first observation is that in all cases the algorithm rapidly reaches its fixed performance which cannot be improved during further learning. The quality of solutions becomes lower when λ is higher. In particular, the lowest result is with $\lambda = 0.9$. In this case, the exploration deviates only slightly from the existing policy. Since longer traces are quicker to obtain their initial policy, they perform less of the initial exploration and they stick to the initial solution. When λ is close to 0, it takes more time to obtain an initial policy, and this yields more random acting which allows exploring more and in this way improves the initial policy.

When experiments were carried out for higher values of ϵ, two issues were noticed. Firstly, curves for higher values of λ were becoming closer to those for low λ values, and additionally all curves were steeper. These two things are correlated. When ϵ is higher than there is more random acting at the beginning, thus performance is low (so curves are low initially and go higher when ϵ is decreased according to our linear scaling) and additionally this gives enough exploration for learning with all λ values. One particular experiment with $\epsilon = 0.7$ is in Figure 7.12. It shows, that in this case, the value of λ is not a matter of importance.

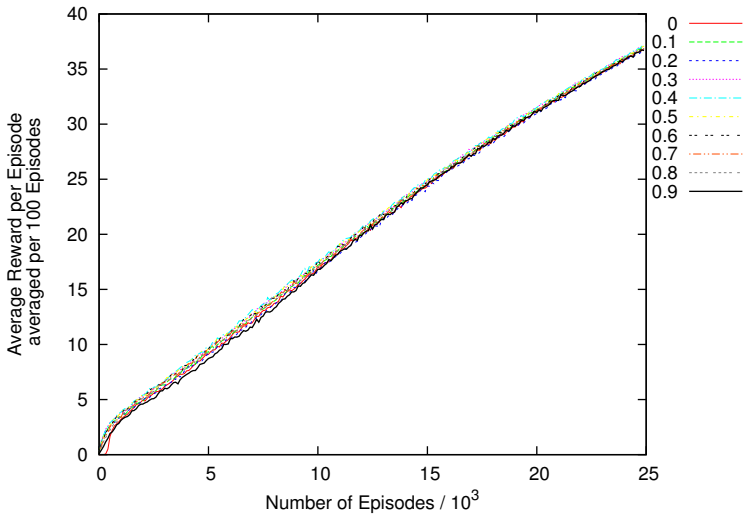


Figure 7.12: Results for the goal-based reward type with pessimistic initialisation and $\epsilon = 0.7$. Each curve corresponds to a different value of λ.

7.3.5.2 Optimistic Exploration

Goal of Experiment: The goal of this experiment is to test λ with a range of exploration rates ϵ under optimistic exploration and the goal-based reward function.

Parameters: λ changes in the range $[0, 0.9]$, ϵ is in the range $[0.01, 0.9]$, and $\alpha = 0.1$.

Results: In this experiment, λ is of much greater importance than in the previous evaluation. Especially high values of this parameter perform well with both small and high ϵ. The result for $\epsilon = 0.01$ is in Figure 7.13. Here, higher values of λ are generally better. Though, $\lambda = 0.8$ and 0.9 is weaker asymptotically then $0.5 - 0.7$ which are the best.

When ϵ grows, $\lambda = 0.8$ and 0.9 get closer asymptotically to those cases which performed the best in previous runs with smaller ϵ. With $\epsilon = 0.5$, $\lambda = 0.8$ and 0.9 catch up with previously better values. Figure 7.14 shows the case when $\epsilon = 0.9$. It shows clearly that results improve monotonically with higher λ values. This case is also different to pessimistic exploration investigated in Section 7.3.5.1 with regard to the information which comes from optimistic exploration. In this case, optimism in itself encourages the algorithm to execute those actions which seem to be promising. Because most actions are not that good in reality, the algorithm has to learn quickly where it is mistaken and longer eligibility traces help with this. This is generally shown in results in this section. When learning with pessimistic exploration (in Section 7.3.5.1), the algorithm has to either explore a lot using external guidance (random in this case when ϵ is high) or apply very short eligibility traces for cases when it does not deviate much from the current policy (low ϵ).

When ϵ grows, all curves become also steeper in the same way as in the previous experiment

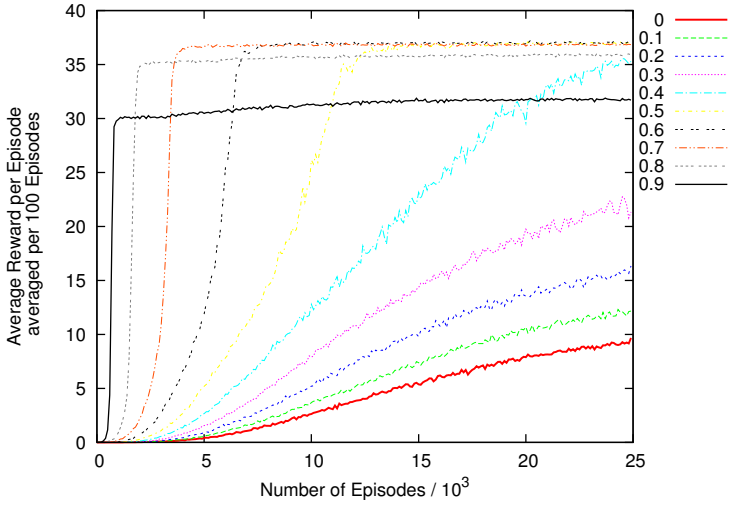


Figure 7.13: Results for the goal-based reward type with optimistic initialisation and $\epsilon = 0.01$. Each curve corresponds to a different value of λ.

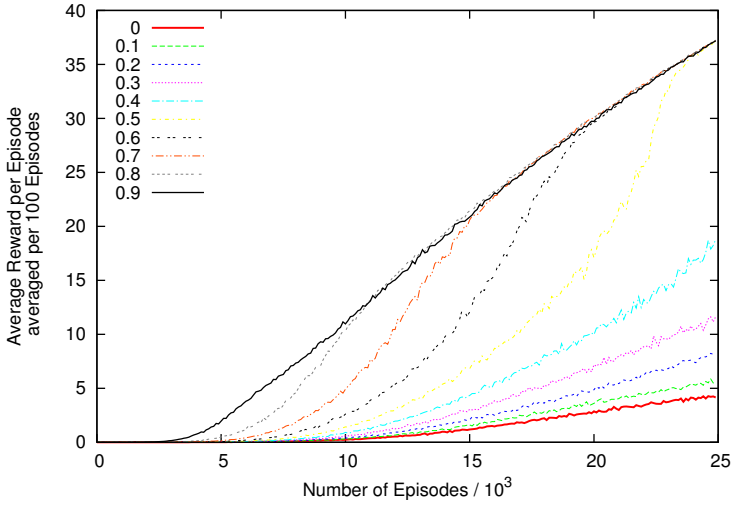


Figure 7.14: Results for the goal-based reward type with optimistic initialisation and $\epsilon = 0.9$. Each curve corresponds to a different value of λ.

were the reason for this was explained (see Section 7.3.5.1).

7.4 Step Rewards

This section repeats the analysis of the previous section with a different reward type. This time, our analysis of exploration with eligibility traces is for the case when there is a negative reward, R_s, for each action in the environment.

7.4.1 Optimistic Exploration

Goal of Experiment: The analysis of λ under the step reward function and optimistic exploration.

Parameters: λ changes in the range $[0, 0.99]$, $\alpha = 0.1$, and $\epsilon = 0.3$.

Results: Learning with the step reward function and zero-initialised Q-table (according to Equation 7.4) shown in Figure 7.15 is accelerated during the entire period of learning when $\lambda > 0$ and the performance improves monotonically when λ increases (except the highest value of $\lambda = 1.0$). The best result is for $\lambda = 0.9$. This fact can be justified as follows. The Q-table is initialised optimistically here and these optimistic values correspond to a trivial heuristic, although this heuristic is admissible (Russell & Norvig 2002). The agent is forced to do exhaustive exploration (this situation is similar to the case when best-first search is executed with a trivial heuristic $h(s) = 0$ for each state s). The propagation of temporal differences when $\lambda > 0$ allows for a faster 'propagation of pessimism' (the negative reward) whereas enough optimism is still preserved to have broad exploration. Because there is enough optimism, long eligibility traces do not accelerate the convergence to the suboptimal solution. Rather, they accelerate the propagation of the negative reward which needs to be propagated anyway due to an optimistic and trivial, from heuristics point of view (Russell & Norvig 2002), initialisation. This reasoning can be further extended to the case when $\lambda = 1.0$ where the performance is decreased. In the very initial phase of learning with this value of λ, the algorithm has the best improvement because it is dealing most rapidly with optimism. However, the very long traces which correspond to this value of λ (shown in Table 7.1) have a negative impact after this short initial period and the performance is spoiled in a similar way as in the experiments of the previous section.

Consistent results with the step-based reward function can be identified in (Framling 2007). This paper does not analyse eligibility traces in the way we do it here - specifically only one type of the reward is considered. However different values of λ are considered in this paper and what can be found in the results is that with the step-based reward the highest values of λ perform the best. Experiments in (Framling 2007) are on a different domain and this fact indicates that our findings can be generalised to other domains. Also in (Leng et al. 2009), high λ is good with the step-penalty reward function.

7.4.2 Semi-Optimistic Exploration

Goal of Experiment: The analysis of λ under the step reward function and semi-optimistic exploration.

Parameters: λ changes in the range $[0, 0.99]$, $\alpha = 0.1$, and $\epsilon = 0.3$.

Results: In order to widen the scope of the evaluation, the step reward function with a pessimistic initialisation was also intend to be checked. But, this case is trickier than it might look at first glance. If the Q-table is simply initialised to a negative value (e.g., -200) for all state-action pairs, then under the condition $\gamma = 1$ this initialisation does not change anything when compared to

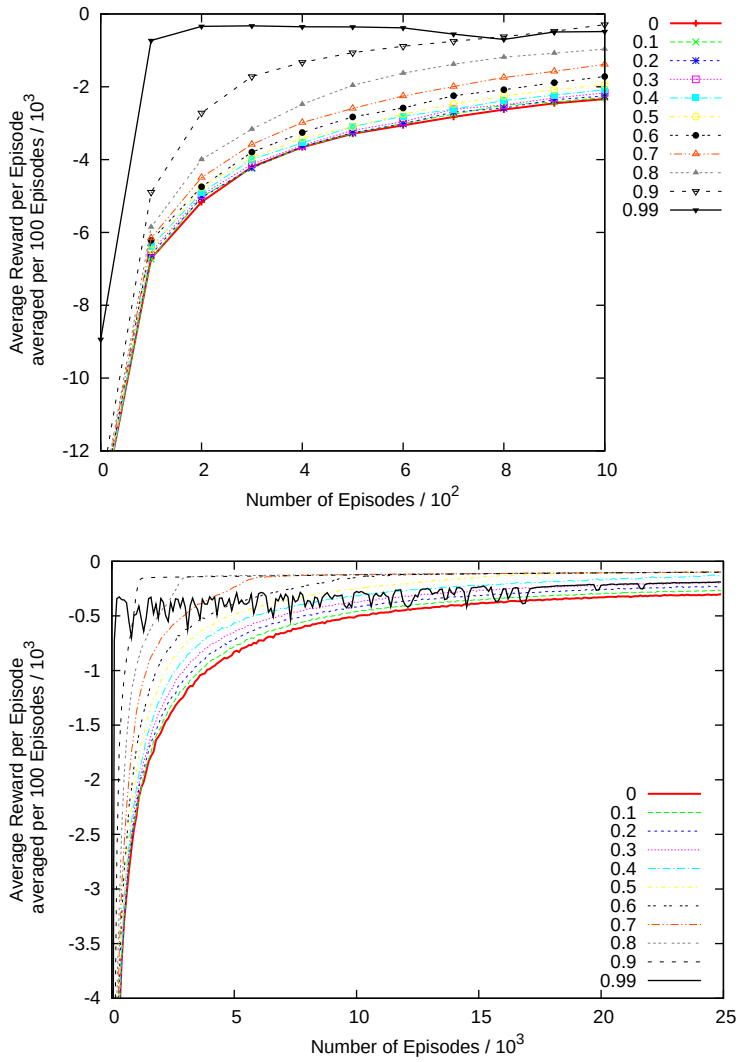


Figure 7.15: Results for the step reward function, R_s, with an optimistic initialisation. Each curve corresponds to a different value of λ.

initialisation with 0 (all Q-values are simply being decreased not from 0 but from another value, e.g., -200). The first thing which can be done in order to change optimistic initialisation is a different initialisation of goal and non-goal states:

$$\forall s \notin \mathbb{G}, a : Q(s,a) = -200, \forall g \in \mathbb{G}, a : Q(g,a) = 0. \tag{7.5}$$

This yields a configuration which we call a semi-optimistic initialisation, because it leads to unchanged learning when the value function from only non-goal states is being propagated. The

propagation of the value function from the goal state gives properties of the pessimistic initialisation because values back-propagated from the goal state are higher than those which come from the initialisation. We explained this issue in detail in our separate work in (Grześ & Kudenko 2009). Results with this initialisation for our experimental configuration of this chapter are shown in Figure 7.16. Learning with a step reward function, R_s, is in this case accelerated by increasing values of λ during the entire period of learning. Only, similarly to the case with optimistic initialisation in the previous subsection, $\lambda = 1.0$ led to a reduced performance. Even though there was an element of the pessimism in this experiment, it was not spoiling the asymptotic performance and λ has the same influence on learning as with optimistic exploration and this type of reward function.

There is, however, one more factor which can make learning with the step reward function less optimistic than the semi-optimistic approach introduced in the previous paragraph. A short analysis of how the temporal difference is computed in the previous experiment shows that if instead of $\gamma = 1$ one uses $\gamma < 1$ this can ensure pessimism even before the propagation of the value function from the goal state starts. To explain this situation, we use the following lemma:

Lemma 3. *The temporal difference, δ, in the first SARSA update is positive when the Q-values - i.e. values of state-action pairs - are initialised to negative values $Q_{init} < 0$, $Q(s,a) = Q_{init} < 0$, $\gamma < 1$ and the step reward satisfies: $r_s > Q_{init}(1-\gamma)$.*

Proof. The temporal difference of the first SARSA update can be written as follows:

$$\begin{aligned} \delta &= r_s + \gamma Q(s', a') - Q(s,a) \\ &= r_s + \gamma Q_{init} - Q_{init} \\ &= r_s + Q_{init}(\gamma - 1) \end{aligned} \tag{7.6}$$

The first equality in this equation is the definition of the temporal difference, the second comes from initialisation $Q(s,a) = Q_{init}$ for all state action pairs, and the last from a direct transformation. From the last line in this equation, we obtain that $\delta > 0$ when $Q_{init} < 0$, $\gamma < 1$ and:

$$r_s > Q_{init}(1-\gamma). \tag{7.7}$$

□

In our domain in which $r_s = -1$, the use of $\gamma = 0.99$ and $Q_{init} = -200$ satisfies Equation 7.7 and specifically $\delta = -1 + 0.99(-200) - (-200) = 1$. This means that such a configuration of parameters leads to a positive temporal difference. Now, we explain why positive temporal difference is so destructive for the exploration based on the Q-table. This is because the positive temporal difference will lead to $Q(s,a) > Q_{init}$ where (s,a) is the updated state-action pair. In this way, an updated entry $Q(s,a)$ for this state action pair gains the highest value

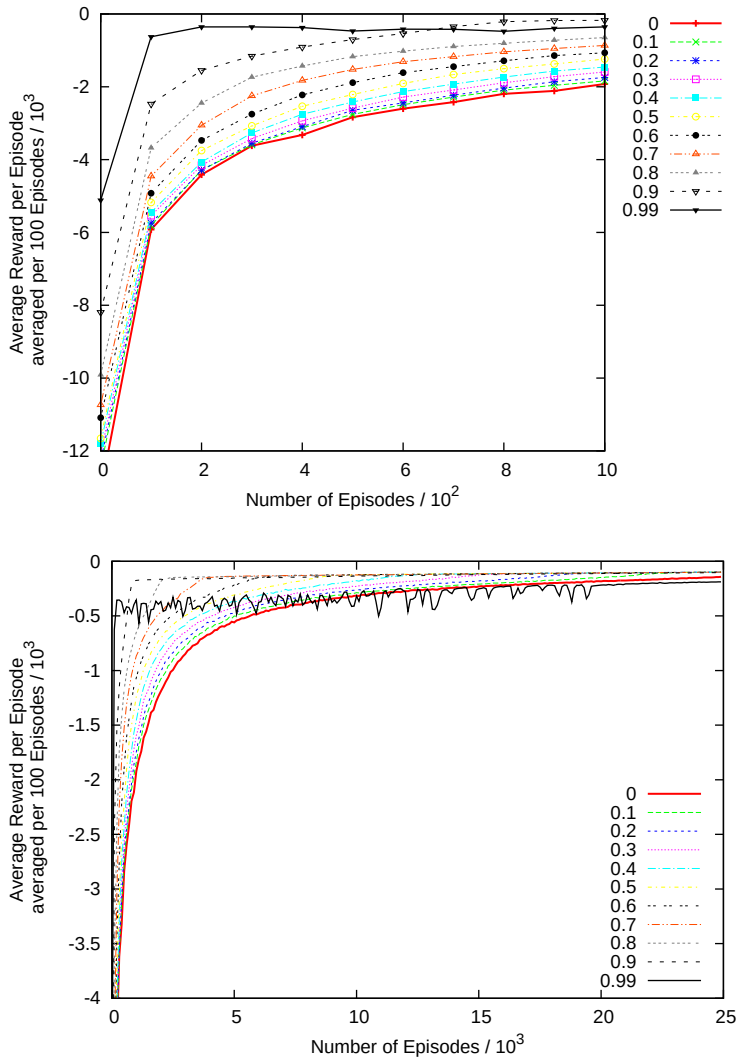


Figure 7.16: Results for the step reward function with a semi-optimistic initialisation. Each curve corresponds to a different value of λ.

when compared to remaining actions of the same state. When the exploration is based on the current content of the Q-table, the agent will be encouraged to execute this action again. Thus, if any loop appears in the current trajectory the agent will stay following this loop with only small divergence (divergence guided by the value of ϵ in the ϵ-greedy exploration). In order to give empirical evidence to support this theoretical discussion, we present empirical results of this configuration. In the left-hand side graph of Figure 7.17, the area of the state space in which the agent was at least once during the first 10^4 steps of the first episode is shown. The right-hand side graph shows the first 10^6 steps. It is worth noting that in both cases it was the first episode

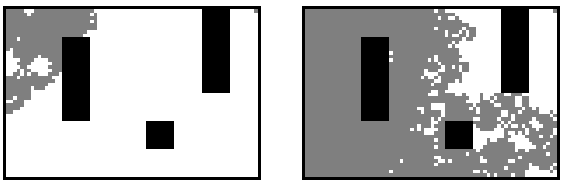


Figure 7.17: An explored (visited) area of the state space when the step reward function is used with the semi-optimistic initialisation of the Q-table and $\gamma = 0.99$: after 10^4 iterations in the left part and after 10^6 iterations in the right part of the figure.

of learning and during given numbers of steps the agent was not able to reach the goal even once. This experiment is of minor significance for this work but makes our analysis of the pessimistic initialisation with the penalty-based reward more comprehensive.

7.4.3 Resetting the Trace

Goal of Experiment: The goal of this experiment is to check whether clearing the trace after each episode will influence results obtained in Section 7.4.1 where λ was analysed under the step reward function and optimistic exploration.

Parameters: All parameters are as in Section 7.4.1. The only difference is that now the eligibility trace is cleared before every new episode starts.

Results: In Section 7.3.3, the goal-based reward function was evaluated with resetting the eligibility trace before each new episode. The same property is analysed in this subsection for the step reward function and optimistic initialisation. The result and ordering of algorithms according to the performance was exactly as in the standard algorithm (see Figure 7.15). The only difference was that, in this case, learning with $\lambda = 1.0$ was more stable and led to higher final result, though it was still worse asymptotically than learning with $\lambda = 0.8$ or 0.9.

7.4.4 Changing Learning Rates

Goal of Experiment: The goal of this experiment is to test λ with a range of learning rates α under optimistic exploration and the step-based reward function.

Parameters: λ changes in the range $[0, 0.9]$, α is in the range $[0.01, 0.9]$, and $\epsilon = 0.3$.

Results: The influence of the learning rate, α, was also analysed with the step reward type and default initialisation of the Q-table with values of 0, which lead to the optimistic exploration in our case (similar tests with the goal-based reward function are in Section 7.3.4).

The overall trend in experiments with different α values is generally the same as in the main result in the Figure 7.15. When α is very small, then highest λ values are considerably better (see Figure 7.18). Here also, the problem is to explore via the reduction of optimism and high λ speeds this up. When α is increased then lower λ values start improving and the difference between small (close to 0) and high (close to 1.0) values of λ becomes smaller. With $\alpha = 0.9$ the difference is very small, but sill higher values of lambda are better, except those highest (like

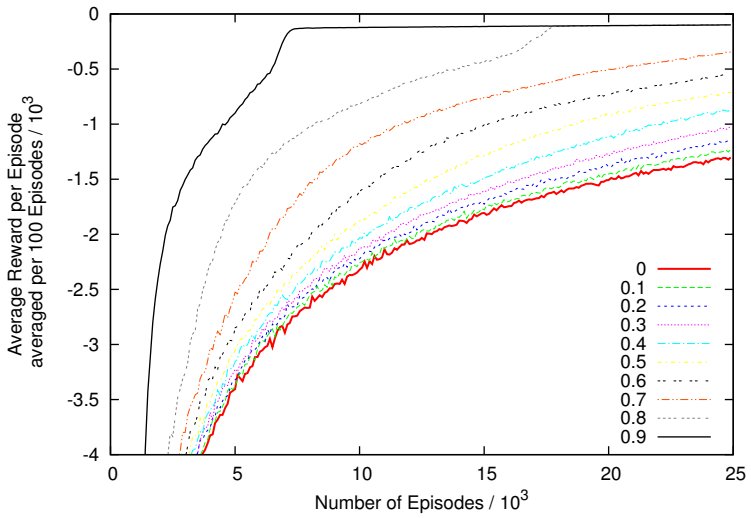


Figure 7.18: Results for the step reward with optimistic initialisation and $\alpha = 0.01$. Each curve corresponds to a different value of λ.

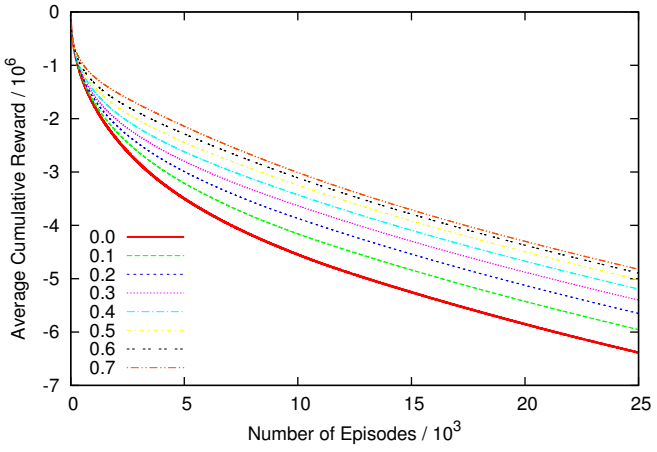


Figure 7.19: Results for the step reward with optimistic initialisation and $\alpha = 0.9$. Each curve corresponds to a different value of λ.

$\lambda = 0.8$ and 0.9) where $\lambda = 0.9$ starts losing asymptotically since $\alpha = 0.3$. The result with $\alpha = 0.9$ is in Figure 7.19. Because, the difference in performance between tested algorithms was not very well observable in the graph which presents average episode reward, this time the overall cumulative reward is shown. The goal of presenting this graph was to show that also in this case, the algorithm's performance improves when λ is higher. For the same reasons as mentioned in Section 7.3.4.2 and explained in detail in Section 7.6, highest values of $\lambda = 0.8$ and 0.9 were not successful in reaching the goal state in a reasonable time.

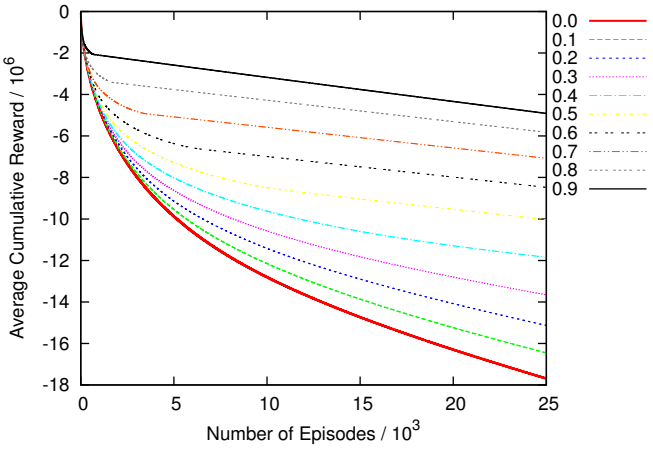


Figure 7.20: Results for the step reward with optimistic initialisation and $\epsilon = 0.01$. Each curve corresponds to a different value of λ.

7.4.5 Changing Exploration Rates

Goal of Experiment: The goal of this experiment is to test λ with a range of exploration rates ϵ under optimistic exploration and the step-based reward function.

Parameters: λ changes in the range $[0, 0.9]$, ϵ is in the range $[0.01, 0.9]$, and $\alpha = 0.1$.

Results: In this experiment, the influence of the exploration rate, ϵ, was also analysed with the step reward type and default initialisation of the Q-table with values of 0, which lead to the optimistic exploration in our case (similar tests with the goal-based reward function are in Section 7.3.5).

Throughout all the runs of this investigation, i.e. with all values of ϵ, the same trend of the influence of λ was preserved. Specifically, higher values of λ monotonically improved the performance (both the initial convergence and the final asymptotic quality). The only problem which was observed was for the case when $\lambda = 0.9$ and ϵ adopts its smallest values (see Figure 7.20). This configuration displays slightly worse asymptotic result even though it has the best initial speed-up. The cumulative reward is used in Figure 7.20, because it was the most legible presentation which shows the discussed problem. This problem disappears with $\epsilon = 0.3$ where $\lambda = 0.9$ becomes the best (see Figure 7.15). So here more exploration, which comes from higher ϵ, allows learning with higher λ to reach the same asymptotic performance as with lower λ values. Even though optimism is used, $\lambda = 0.9$ which yields the most rapid propagation may converge to a suboptimal result and additional randomness in exploration due to higher ϵ allows the algorithm to execute more explorative actions which improves the policy.

7.5 Further Analysis on Random Walk

Goal of Experiment: The goal of this experiment is to test λ with the goal-based reward function and pessimistic initialisation on a domain which has different properties than the S-maze task which was used in the main experiment in Section 7.3.1 where this configuration of reward and initialisation was particularly problematic. This experiment is based on the random walk task.

Parameters: λ changes in the range $[0, 1]$, $\epsilon = 0.3$, and $\alpha = 0.1$.

Results: In this experiment, we use the random walk task (see Section 5.4.1 and Figure 5.1 for details). This domain does not have problems which are depicted in Figure 7.1. The importance of the choice of this task is also the fact that it has been used in the literature in many empirical evaluations of eligibility traces (Andrecut & Ali 2004; Singh & Sutton 1996; Sutton 1988; Sutton & Barto 1998). Particularly we are interested in the performance of the goal-based reward with a pessimistic Q-table in this task because this configuration caused problems in our tests on S-maze in Section 7.3.1. There are 128 states in this experiment. The agent starts in the most left position and has to reach the most right position where reward 100 is given upon entering the final state. All other rewards are equal to 0. Other details are according to the description of this domain in Section 5.4.1. All learning parameters are the same as in our main configuration (see Section 7.2.1). Results are in Figure 7.21. The convergence in the initial phase of learning (the top graph of Figure 7.21) shows that $\lambda > 0$, in contrast to $\lambda = 0$, accelerates learning at the beginning. The expected observation is that in the latter period of learning (up to 10^4 learning episodes in the top graph of Figure 7.21) no suboptimal convergence is observed for all values of λ. This is an expected observation because this domain does not have the property shown in Figure 7.1. Only for $\lambda = 0.9$, initially lower performance can be observed. This domain does not yield suboptimal solutions which exist in more general navigation problems (Figure 7.1) and learning even with high values of λ can be successful in a given configuration of the reward function and initialisation (this was not the case in the same experiment on S-maze as shown in Figure 7.3). This task is substantially more specific and this fact should be taken into account when designing empirical evaluations of RL algorithms. It is worth emphasising that this domain has been often used in analyses of RL with eligibility traces (Andrecut & Ali 2004; Singh & Sutton 1996; Sutton 1988; Sutton & Barto 1998). The highest value of λ which is reported in this experiment is 0.9. The value of 1 was also evaluated but in this domain learning with $\lambda = 1$ led to considerably long traces and was not successful, that is, the agent ran into very long episodes and it was infeasible to wait until it reached the goal state which would end the episode. This specific issue is addressed in detail in Section 7.6.

7.6 Non-converging Settings

During experimentation of this chapter, in several cases learning was not successful in a sense that the agent was not able to reach the goal state in a feasible time and such configurations

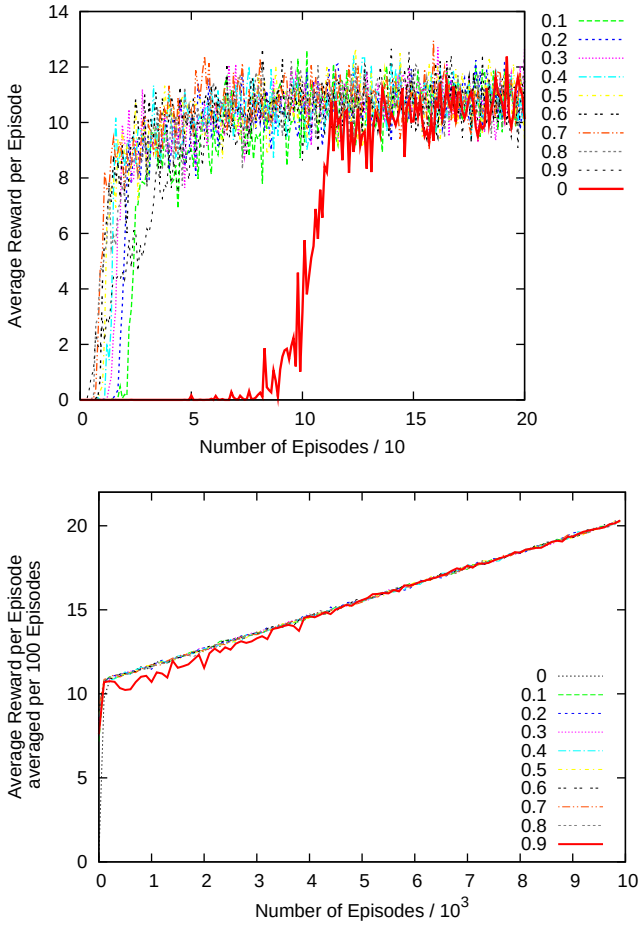


Figure 7.21: Results for the goal-based reward with pessimistic initialisation on RW. Each curve corresponds to a different value of λ.

were treated as not converging. This happened for example in Section 7.5 with $\lambda = 1.0$, in Section 7.3.4.1 $\lambda = 0.9$ also has problems with high α, or in Section 7.3.4.2 this is also the case with $\lambda = 0.8$ and 0.9. Some explanation for such behaviour has been already presented in Section 7.3.1 where it was supported by the comparison of the length of traces according to relevant parameters, which is presented in Table 7.1. The length of eligibility traces grows exponentially with λ and becomes significantly longer when λ approaches 1. Thus, in such cases traces are relatively much longer and this leads to more radical changes in the Q-table after each SARSA update. This increased number of updates turned out not to be problematic when learning with high ϵ because of the contribution of random exploration which allowed overcoming loopy paths encoded in the Q-table. In the case of different values of the learning rate α, λ appears to catalyse the impact of the learning rate. In particular, when both of these parameters are high,

their contribution is very high which leads to non-systematic updates in the Q-table and loopy paths arise which restrict the algorithm from reaching the goal state when ϵ is sufficiently small. A certain amount of additional debugging was performed to check the content of the Q-table in such situations (e.g., with $\lambda = 1.0$ in RW), and it confirmed that Q-values become non-monotonic with regions of low and high values which easily trap the agent in loopy paths.

The overall analysis of the results obtained in this chapter indicates that a reasonable heuristic to obtain the maximum value of the λ parameter could be the one which yields the eligibility trace of the length comparable (i.e. of the same order of magnitude) with the maximal path length in the domain. This is however only our guess which, although supported by the results of this chapter, requires more experimentation on wider range of domains to test its validity in a more general context.

7.7 Summary and Discussion

In this chapter, the influence of eligibility traces on exploration in the SARSA(λ) algorithm was analysed under different types of reward function, initialisation of the Q-table, and exploration strategies. The analysis was focused on both the initial learning rate and the asymptotic convergence which usually is omitted in empirical evaluations when eligibility traces are used. The quality of the policy is compared instead of the error of the value function estimation because good acting regardless on how well the value function is approximated is most often desired. Most of the empirical analyses of eligibility traces existing in the literature have focused also mainly on one type of reward function at a time. In some cases, the step reward (R_s) was used (Cichosz 1995; Framling 2007; Leng et al. 2009), and in others the final goal reward (R_g) (Preux 2002; Wyatt et al. 1999; Cichosz 1996).

The experimental setup and findings reported in this chapter are novel as according to our best knowledge the problem has never been discussed and analysed in the literature from this perspective. *Results yield better and novel understanding of relationships between λ, the type of reward function, and exploration based on the value function.* Additional experiments conducted in this work, uncovered some further relevant relationships between λ, the learning rate, α, and the exploration rate, ϵ.

The overall findings of this chapter can be summarised as follows:

1. The impact of λ depends on whether exploration is pessimistic or optimistic. With pessimistic exploration, high λ easily leads to low asymptotic performance, even though it is good at the very beginning. With optimistic exploration, higher λ values in most cases lead to a better performance because they allow for faster propagation of lower (non-optimistic) values.

2. Even though higher λ values improve learning with optimistic initialisation, the type of reward function is also important. Specifically, the goal-reward function is more sensitive

to high values of λ than the step-based reward function.

3. It was shown that high λ does not cause problems for pessimistic initialisation with the goal-based reward function on another type of domain. In particular, learning with highest values of λ is not destructive in random walk, because there are fewer opportunities for the path to be suboptimal. The general 2D maze scenario is more problematic (see Figure 7.1).

4. The size of the eligibility trace should have at most the same order of magnitude as the maximum path length in the domain.

5. Clearing the eligibility trace before every new episode does not change results with most values of λ on both evaluated classes of reward function. The only difference was with the highest values of $\lambda = 1$ when clearing the traces was slightly improving the result.

6. When exploration stays sufficiently close to the current policy (i.e. the exploration rate ϵ is appropriately small), α and λ cannot take their maximum (or close to maximum) values because the resulting exploration may not be successful and the algorithm may not be able to reach the goal state and finish the episode. It was also observed and mentioned by Cichosz (1995) that high λ may require lower α.

7. When learning with the α value close to 1, high λs become more destructive when learning with the goal-reward and pessimistic initialisation. Optimistic initialisation with both types of reward function is more robust against variations in the value of α.

8. In the baseline experiment, higher λs were improving performance with optimistic initialisation. This improvement was even higher when the learning rate α was small.

9. In all experiments with high exploration rate ϵ, λ becomes less important. The explanation for this is that in this case exploration becomes independent from the current policy because actions are selected randomly regardless of the corresponding Q-values. This case is however not interesting from the point of view of practical applications because the agent acts mostly randomly.

10. When the current policy influences exploration more (i.e. when ϵ is small in our case), high λ values are destructive when learning with the goal-based reward function and pessimistic initialisation. With optimistic exploration, higher λ was generally better with different values of ϵ.

Results of similar experiments with three versions of Q(λ) (i.e. Watkins', Peng's and naive Peng & Williams 1996; Sutton & Barto 1998) may be different particularly because of the off-policy character of Q-learning and its implications on eligibility traces. It is worth noting here that SARSA and Q-learning become equivalent when used with greedy exploration based on the

current content of the Q-table, e.g., ϵ-greedy with $\epsilon = 0$. This means that the difference between Q-learning and SARSA becomes smaller in this case when ϵ is getting closer to 0.

It was highlighted in Section 7.2.2 that the empirical evaluation in this chapter was carried out on the instance of the broad class of RL problems in which regardless of its type (i.e. goal-based reward or step penalty) the reward function determines one objective. In this case it is to navigate as fast as possible to the goal state subject to constraints imposed by transition probabilities of the underlying MDP. We conjecture that our findings in this chapter can be generalised to at least similar problems. The evaluations with rewards which provide more than one objective would be enlightening in the context of our investigation and could be seen as an interesting future work.

One more important factor which should be stressed here is that our results are based on the tabular representation of the state space with a distinct value for each state-action pair. We conjecture that function approximation would mitigate the convergence to a suboptimal solution with the goal-based reward and a pessimistic initialisation. In function approximation, the value function is shared between at least neighbouring states and such a knowledge transfer may help in avoiding convergence to suboptimal solutions. But, we leave the analysis of eligibility traces with function approximation for future work, particulary as the convergence mechanisms of learning with function approximation have not been well understood in the field (Stone et al. 2005) and function approximation in itself can pose problems (Boyan & Moore 1995; Gordon 1995).

This research was entirely based on replacing eligibility traces when the eligibility of the current state is always set to 1 instead of being increased by 1 as it is the case in accumulating eligibility traces (Singh & Sutton 1996). Replacing traces are more common in the literature (Kuhlmann & Stone 2004; Stone et al. 2005) and in existing empirical comparisons were performing better than accumulating traces (Reynolds & Wiering 2002; Singh & Sutton 1996; Sutton & Barto 1998). This is the reason for using replacing traces in this work. However, after conducting research work which is presented in this chapter, we have some intuition of when accumulating traces may work better and when not. We conjecture that accumulating traces could be generally better (with a given value of λ) in situations when high λ is better - e.g., with the step-based reward and default optimistic exploration. On the other hand, we expect accumulating traces to work worse in situations when high λ values are decreasing performance. This issue could be seen as a future work on understanding and explaining the role of eligibility traces on exploration in reinforcement learning.

In this chapter, our analysis was focused on the robustness of eligibility traces when different types of the reward function and initialisation/exploration are used. The insight into the problem which we gained from this work shows that interesting conclusions can be drawn about the benefits and disadvantages of different reward configurations and initialisations of the Q-table. Such an investigation was presented in (Grześ & Kudenko 2009) where the issue of a different initialisation of goal and non-goal states was exploited and investigated on how it can improve exploration.

CHAPTER 8

Conclusion and Future Work

This chapter summarises the thesis. It starts with a brief overview of the problem which is addressed in this research. After that, the hypothesis is revisited, and the proceeding section summarises all main research contributions of this thesis. Then, thesis limitations are discussed, and they are followed by future work. The document ends with final remarks.

8.1 Overview

Solving reinforcement learning problems is a hard task. It is generally a common problem in science that theoretical models become intractable when applied to real environments of a considerable and realistic size. The same situation exists also in reinforcement learning (RL). A substantial part of RL research has focused on algorithms which require the so called Markov property to be satisfied. It allows for neat and convenient mathematical modelling, and also easiness of algorithm design and analysis, but also yields one of the most serious problems: *the exponential state space explosion*. This problem arises from the fact that each state feature added to the encoding of the state space yields an exponential increase in the number of states. When the state space grows in this way, the inherent problems of reinforcement learning such as the *exploration-exploitation* and the *temporal credit assignment problems* become more significant. This naturally leads to the need for heuristics and approximate solutions.

The characteristic feature of RL is that the agent usually has significant influence on the training data which is available (e.g., in the form of samples) because the agent itself decides which actions are executed during learning and thus how it moves in the environment. This creates two distinct issues in the design of RL algorithms: (1) how to represent and approximate

an optimal policy from existing data, and (2) how to select actions during learning so that the number of suboptimal actions which are executed during the training process is minimised. The second issue is known as the exploration-exploitation problem.

In the simplest case, the RL agent uses an arbitrary exploration strategy to select actions to obtain more information from the environment (i.e. samples) and improve its estimation of the policy. For practical reasons, when facing huge state spaces, practitioners are forced to use exploration which is based on the current policy with a certain degree of randomness which deviates from such a policy.

An important issue in RL is that the way the policy is approximated and updated influences the actual exploration which comes from the above mentioned exploration which is based on the current policy. Another important factor which can positively influence exploration of the RL algorithm is domain knowledge. Domain knowledge can be used in different ways. This thesis pays significant attention to reward shaping. Reward shaping applies heuristic knowledge in order to give additional (artificial) reward to the agent in order to improve its learning. This external reward is in addition to the original reward which comes from the environment. Such a reward when based on good heuristics may, already in the early stages of learning, increase the value of the best actions and those actions will be chosen more often. This fact emphasises the indirect influence of the shaping reward on exploration.

In this thesis, we were looking for both novel and better ways of improving exploration, and for better understanding of the interactions between the processes of (1) estimating the policy and (2) deciding on exploration. In particular, the focus was on the use of existing domain knowledge via reward shaping or similar ways, obtaining knowledge when it is not available, and restructuring it through reasoning and learning when it is not easily applicable in RL. In addition to the explicit use of knowledge to improve exploration in RL, this thesis analysed also two specific RL features (reward shaping and eligibility traces) which govern the way the algorithm approximates the policy and which as a result influence actual exploration.

8.2 Hypothesis

The hypothesis of this thesis is:

> *A detailed analysis of knowledge used in reinforcement learning as well as an analysis of domains' and algorithms' properties will create new ways in which domain knowledge is incorporated, help to improve exploration in reinforcement learning and determine when and how to deploy reinforcement learning algorithms.*

8.2.1 Brief Summary of What Was Achieved

A number of steps were undertaken to show evidence supporting this hypothesis. The use of knowledge requires well established techniques for incorporating knowledge into RL algorithms. Reward shaping was used as such a framework for incorporating procedural knowledge into

model-free RL. Two new ways of obtaining heuristics for potential-based shaping were introduced and evaluated. The first one (Chapter 3) uses high level symbolic knowledge, and the second one (Chapter 4) applies a different hypothesis space to learn the heuristic. The empirical evaluation provided evidence that our approaches are successful. *These techniques open the way to improve RL via reward shaping in situations when there is no information about the potential function.*

Our encounter with reward shaping led to another major contribution which showed specific properties of reward shaping. In particular, it was formally specified what the actual shaping reward is under different parameters, and empirical evidence was also shown which confirms the theoretical findings. *These are novel findings which allow for easy identification (through derived equations) of potential problems when using reward shaping.*

In the context of model-based RL, a novel technique to incorporate knowledge into the initial MDP-models was proposed, evaluated, and proven to meet properties of PAC-MDP learning. It was also shown that the empirical performance of this 'greedy' optimistic exploration was improved with our method. *The contribution of this research is the fact that it shows how to use specific domain knowledge in PAC-MDP algorithms in a theoretically correct way.*

One of the important factors which influence exploration in RL is the concept of eligibility traces. The last contribution shows a detailed analysis of how it influences exploration under a multitude of conditions. *This is a contribution in the light of existing limited analyses, which focus mainly on the prediction problem.*

8.3 Summary of Main Contributions and Findings

In this section, the main contributions and findings of this thesis are summarised.

8.3.1 Plan-based Reward Shaping

A new method to define the potential function for potential-based reward shaping using abstract plan knowledge represented in the form of STRIPS operators was shown. We empirically compared the performance of our approach with the performance of RL without any reward shaping, RL with a manually shaped reward, as well as a related automatic shaping approach based on abstract MDPs (Marthi 2007). The results of the experiments demonstrate that STRIPS-based reward shaping improves in certain situations both the quality of the learned policy and the speed of convergence over the alternative techniques. The evaluation was performed with two types of exploration, which yields also a deeper insight into the effect of reward shaping on model-free learning and this constitutes an additional contribution of this work. Overall, this contribution can be summarised as follows:

1. Pessimistic exploration in conjunction with model-free RL represents a good approach for solving large RL problems because of the way it deals with huge state spaces (i.e. the pessimistic exploration guarantees that unnecessary state action pairs can be neglected and

model-free RL does not have to learn the model). But, such an approach may result in extremely poor results when the domain is difficult in terms of exploration, e.g., when there are many goals with different rewards. The first part of the experimental section of this chapter showed that this kind of RL can be successfully implemented even in domains with difficult exploration when used with plan-based reward shaping.

2. The strong point of the plan-based reward shaping is that it avoids exploring and storing many unnecessary states in the Q-table. This was confirmed in our experiments and this property makes it particularly suitable also with more sophisticated function approximation methods which can profit from informative avoidance of unnecessary states, e.g. the Kanerva coding implementation in (Wu & Meleis 2008, 2009a,b).

3. Tests with optimistic exploration showed that already in our artificial test domain this kind of exploration when used without reward shaping requires a considerable number of episodes to converge (this is a standard disadvantage of this type of exploration), though it can find the optimal solution with regard to the number of collected flags. Any type of reward shaping is extremely helpful in cutting the search space of optimistic exploration, however the quality of the reward shaping plays an important role when asymptotic convergence is considered. The plan-based reward shaping was also very competitive under this criterion.

4. STRIPS-based shaping showed in several cases better results than the MDP-based approach, because the agent was strongly influenced by the plan that guides it towards a good policy. Thus, this observation suggests one potential improvement to MDP-based reward shaping. Instead of using the value function of the entire state space as the potential function, the best path which corresponds to the STRIPS plan can be extracted and used with our algorithm to define the potential function.

5. Evaluation with wrong knowledge showed that specific reward shaping methods are more or less resistant to certain knowledge inaccuracies, and general patterns were identified. For example, the MDP-based reward shaping is not resistant to situations when planning knowledge does not include all goal predicates. Plan-based reward shaping is the most error prone in situations when inaccurate knowledge assumed high level transitions which do not exist in the actual environment. These observations should be taken into account when deploying reward shaping and when certain predictions can be made on what kind of things may be wrong in the abstract model of the domain.

The advantages of STRIPS based reward shaping were additionally discussed at the end of the closing section of Chapter 3. Here, we emphasise the overall significance of this contribution. *This research opens the way to improve RL via reward shaping in situations when there is no information about the potential function, but when symbolic knowledge about STRIPS actions*

can be identified and restructured to the form of the potential function. A rigorous comparison against alternative approaches and under two different exploration strategies showed what kind of improvement is expected in each of them. The results yield additional insight into the effect of reward shaping under optimistic and pessimistic exploration in model-free RL. *In the case of pessimistic exploration, the reward shaping is encouraging the policy which agrees with shaping knowledge whereas the agent does not tend to diverge from this policy. When optimistic exploration is considered, reward shaping tends to prune irrelevant states because the agent explores broadly due to optimism.*

8.3.2 Reward Shaping and Mixed Resolution Function Approximation

This contribution deals with the same problem as the work discussed in the previous section. This time another approach is proposed which is applicable in situations when there is no STRIPS-like knowledge. Here, we propose using two hypotheses spaces, that is, function approximation with different levels of expressiveness in RL. Two approaches to obtain learning with such a mixed resolution are introduced and empirically evaluated when applied to tile coding. *The results show that simultaneous learning at two levels and learning with mixed resolution FA can converge to a stable solution and improve overall learning performance.* We conjecture that the success of this approach is due to the fact that our implementation is based on the SARSA algorithm (on-policy temporal difference learning) which has been shown in the literature (Stone et al. 2005) to work better with function approximation than Q-learning (which is off-policy).

Results on tasks selected according to different properties show that the application of these extensions to RL are especially beneficial when:

1. there are many actions in each state,
2. a high resolution of the policy is required (due to details in the environment) with a wide range of values of state variables, i.e. on the larger instance of the domain,
3. a high level guidance can be extracted from a subset of state variables,
4. the reward is given only upon reaching goal states.

Reward shaping with mixed FA was the best in all runs on large instances. Actually only on the car parking task with the original size and $\lambda = 0$ the standard algorithm was the best, though without statistical difference. Learning with only mixed FA was the second-best on two domains but reward shaping without mixed resolution was better on one domain, that is, when the path to the goal led via states with very constrained values of state variables (entering the parking space in the car parking task). Overall, the results show that reward shaping with mixed resolution FA at the ground level was the most successful.

The strong point of the algorithm is the improved convergence rate, especially on domains with properties outlined in the list above.

The comparison between learning with $\lambda > 0$ and $\lambda = 0$ showed that our algorithms generally lead to better absolute improvement when $\lambda = 0$, but good asymptotic properties were preserved in both cases in most experiments. Additionally, our algorithms without eligibility traces gained a similar performance faster than SARSA(λ) with eligibility traces. Eligibility traces, even when using a more efficient version (truncating is used in our experiments) yield certain computational overhead. With $\lambda = 0$, only one backup is performed after each step and with $\lambda = 0.7$ (and other relevant parameters according to our experimental design) the number of backups is $N = 56$. The computational complexity is significant and was empirically observed during evaluation. *This observation indicates that with our methods applied without eligibility traces, a comparable convergence can be achieved at lower cost, because there is at most one backup of the V-function for each SARSA backup. Eligibility traces require significantly more updates.* In contrast to eligibility traces, separate and external representation of knowledge is obtained in our method with reward shaping.

It is important to emphasise again that ideas proposed in this work do not require any 'unusual' domain knowledge. In its basic form, abstract learning can be defined using the same knowledge which is used to design tile coding at the ground level. The most straightforward approach is the use of wider intervals of high level tiles, and this would allow applying reward shaping derived from the high level value function. Experiments in this research were based on standard RL domains. It would be easy however to construct scenarios where different state features are responsible for different behaviour and learning an individual value function for each behaviour would yield additional improvement to the main learning process when such value functions would be used for reward shaping. For brevity, we did not consider such tailored domains in this work, and our aim was to show improvement on standard domains.

8.3.3 Analysis of Reward Shaping

Our experimentation with reward shaping led us to several novel findings about the actual shaping reward which is given to the learning agent. The contribution of this work can be summarised as follows:

1. When $\gamma = 1$, the potential function can be both positive and negative and in both cases the performance is exactly the same.

2. Even when $\Phi(s) = V(s)$ (which is a desirable potential function), the learning algorithm still needs to learn effects of actions and for this reason scaling the shaping reward up ($\tau > 1$) improves the learning rate, because the exploration is improved. When $\tau > 1$, the theoretical requirements of potential-based reward shaping according to Ng et al. (1999); Wiewiora (2003) are preserved.

3. In domains with faulty heuristics, one cannot scale the potential function up too much because the agent may be heavily penalised for diverging from the shaping reward and this

may result in failures in reaching the goal state. However the scaling factor, τ, with the value of 2 yielded best results on all tested domains with different quality of the heuristic function and this value can be considered in practical applications.

4. The analysis of the actual shaping reward in domains with $\gamma < 1$ was conducted (summary in Tables 5.1 and 5.2) and results allow explaining the outcomes of the empirical analysis.

5. When $\gamma < 1$ and learning with the step reward, R_s, the positive potential function performs worse than the negative one and the scaling factor $\tau > 1$ improves learning with the positive potential function. The negative potential function is better in the initial configuration but breaks when conditions defined in equations which were derived (Equations 5.11-5.13) are significantly violated. This can happen even with a very accurate heuristic function. The re-initialisation of the Q-table to higher values allows avoiding very long, loopy episodes which prevent the algorithm from reaching the goal state.

6. The goal reward, R_g, seems to be more challenging to reward shaping. Generally, both types of potential function lead to a considerable improvement only at the very beginning of learning, when the non-shaping agent performs random exploration. For higher lengths of the random walk domain (e.g., length of 128) or generally situations when conditions in derived equations (Equations 5.2-5.4) are violated to a higher extent, the positive potential function, even though good initially, is significantly worse than no shaping. The negative potential function leads in these cases to loopy episodes, and a different initialisation of the Q-table is required to allow the agent to reach the goal state in a reasonable time.

7. Additional analysis of the previous case revealed that transitions $s \rightarrow s$ should be rewarded according to the standard evaluation of the shaping reward, $F(s, s)$, because these transitions should be constantly penalised (see Table 5.1).

8. A new method to evaluate the shaping reward from the potential function was proposed. It applies a different discount factor for computing the shaping reward (see Equation 5.17). It was proven that with such a reward shaping the optimal policy of the shaped MDP may be different from the original one which does not use shaping. However, in large domains, where approximate solutions are satisfactory it may still be more profitable to learn such a shaped policy faster than the original one more slowly, despite possibly not reaching the same performance on the main MDP.

9. Experimental evaluation of this approach, Equation 5.17, showed that it solves the problem of different learning with the positive and negative potential functions on tasks where the heuristic function which is used as the potential function is relatively accurate. The problems were still encountered on the S-maze task where the heuristic function is the faultiest.

10. The final contribution of this chapter showed that even the standard potential-based reward shaping approach may lead to the policy which is not equivalent to the policy of the original MDP on domains with many goal states. The potential function of goal states requires special treatment in such cases (see Section 5.8).

Our findings do not violate the relevant theory on potential-based reward shaping (Ng et al. 1999; Wiewiora 2003). When Equation 5.1 is used, the optimal policy of the shaped MDP is the same as of the original non-shaped MDP. The problem on which our research focuses is how successful exploration is under different reward shaping approaches, algorithm properties and domain properties. Since this kind of reward shaping has the equivalent initialisation, the same problems can be encountered with the corresponding initialisation of the value function. The proposed shaping with Equation 5.17 has good potential of applicability in large domains where the optimal solution cannot be found in a reasonable time (or even very long time), but the policy of the shaped MDP can be learned faster and represent a more accurate solution to the target MDP.

8.3.4 PAC-MDP Learning with Knowledge-based Admissible Models

PAC-MDP algorithms are particularly interesting from an analytical point of view because they approach the exploration-exploitation problem in a way which guarantees that with high probability, the algorithm performs near optimally for all but a polynomial number of steps. It was shown in this work, how the performance of these algorithms can be further improved by incorporating domain knowledge to guide the learning process. The lack of such methods was shown to be a weak point of PAC-MDP algorithms and alternative non-PAC-MDP methods were proposed and shown to be competitive (Kolter & Ng 2009). In this work, we propose a framework to use partial knowledge about effects of actions in a theoretically well-founded way. This contribution can be summarised as follows:

1. With the use of a symbolic specification of MDP actions in the PPDDL formulation, potentially available domain knowledge was distinguished and it was shown how to use this knowledge with PAC-MDP algorithms in a way which preserves theoretical properties of these algorithms.

2. The empirical evaluation shows that our proposed method is more efficient than reward shaping which represents an alternative approach to incorporate background knowledge. Reward shaping requires an admissible heuristic which has to be designed manually, but in domains represented symbolically via PPDDL it is difficult to design such admissible heuristics. Our solution uses only local action knowledge and can be applied when such heuristics cannot be designed or when those designed are not accurate. Our results show that even if such heuristics exist, our approach can be more efficient. Informally, it can be argued that our approach will always be better than reward shaping, because in our

case domain knowledge is used when state action pairs are still unknown (i.e. unknown in the Rmax sense). Knowledge injected via reward shaping is used only with known state-action pairs. This comparison between reward shaping and methods which we proposed applies only to model-based RL. It does not apply to model-free RL where reward shaping is a powerful method for incorporating domain knowledge. This work gives also a good insight into the significance of different kinds of knowledge on the learning performance of various PAC-MDP algorithms.

3. Our solution is also very competitive when compared with the Bayesian Exploration Bonus (BEB) algorithm. BEB is not PAC-MDP, however it can exploit domain knowledge via informative priors. We show how to use the same kind of knowledge in the PAC-MDP framework in a way which preserves all theoretical guarantees of PAC-MDP learning.

4. The presented results indicate also that FSA knowledge leads to more informative admissible models and should be preferred to AO knowledge when applied to PAC-MDP algorithms such as Rmax.

8.3.5 Analysis of Exploration with Eligibility Traces

In this contribution, the influence of eligibility traces on exploration in the SARSA(λ) algorithm was analysed under different types of reward function, initialisation of the Q-table, and exploration strategies. The analysis was focused on both the initial learning rate and the asymptotic convergence which usually is omitted in empirical evaluations when eligibility traces are used. The quality of the policy is compared instead of the error of the value function estimation because good acting (regardless how well the value function is approximated) is most often desired. Most of the empirical analyses of eligibility traces existing in the literature have focused also mainly on one type of the reward function at a time. In some cases, the step reward (R_s) was used (Cichosz 1995; Framling 2007; Leng et al. 2009), and in others the final goal reward (R_g) (Preux 2002; Wyatt et al. 1999; Cichosz 1996).

The experimental setup and findings reported in this research are novel as according to our best knowledge the problem has never been discussed and analysed in the literature from this perspective. *Results yield better and novel understanding of relationships between λ, the type of the reward function, and exploration based on the value function.* Additional experiments conducted in this work, uncovered some further relevant relationships between λ, the learning rate, α, and the exploration rate, ϵ. The overall findings can be summarised as follows:

1. The impact of λ depends on whether exploration is pessimistic or optimistic. With pessimistic exploration, high λ easily leads to low asymptotic performance, even though it is good at the very beginning. With optimistic exploration, higher λ values in most cases lead to a better performance because they allow the algorithm for faster propagation of lower (non-optimistic) values.

2. Even though higher λ values improve learning with optimistic initialisation, the type of reward function is also important. Specifically, the goal-reward function is more sensitive to high values of λ than the step-based reward function.

3. It was shown that high λ does not cause problems for pessimistic initialisation with the goal-based reward function on another type of domain. In particular, learning with highest values of λ is not destructive in random walk, because there are fewer opportunities for the path to be suboptimal. The general 2D maze scenario is more problematic (see Figure 7.1).

4. The size of the eligibility trace should have at most the same order of magnitude as the maximum path length in the domain.

5. Clearing the eligibility trace before every new episode does not change results with most values of λ on both evaluated classes of reward function. The only difference was with the highest values of $\lambda = 1$ when clearing the traces was slightly improving the result.

6. When exploration stays sufficiently close to the current policy (i.e. the exploration rate ϵ is appropriately small), α and λ cannot take their maximum (or close to maximum) values because the resulting exploration may not be successful and the algorithm may not be able to reach the goal state and finish the episode. It was also observed and mentioned by Cichosz (1995) that high λ may require lower α.

7. When learning with the α value close to 1, high λs become more destructive when learning with the goal-reward and pessimistic initialisation. Optimistic initialisation with both types of reward function is more robust against variations in the value of α.

8. In the baseline experiment, higher λs were improving performance with optimistic initialisation. This improvement was even higher when the learning rate α was small.

9. In all experiments with high exploration rate ϵ, λ becomes less important. The explanation for this is that in this case exploration becomes independent from the current policy because actions are selected randomly regardless of the corresponding Q-values. This case is however not interesting from the point of view of practical applications because the agent acts mostly randomly.

10. When the current policy influences exploration more (i.e. when ϵ is small in our case), high λ values are destructive when learning with the goal-based reward function and pessimistic initialisation. With optimistic exploration, higher λ was generally better with different values of ϵ.

Results showed that the performance of SARSA(λ) depends on the reward type and is additionally correlated with the initialisation of the Q-table when exploration is based on the current policy. The influence of the learning rate, α, and the exploration rate, ϵ, was also investigated in

detail. Overall, high λ reinforces the impact of α. When at the same time, both α and λ were close to their maximum values, learning did not converge, that is, the goal state was not reached in a feasible time. High exploration rate, ϵ, on the other hand, yields random exploration which does not depend on the content policy encoded in the Q-table and λ becomes less important.

8.4 Limitations

In this section the most significant limitations of presented work are discussed.

Goals in Symbolic Planning The inherent property of existing symbolic planning techniques is that they require a well specified goal formula (e.g., a first order expression which is satisfiable by one or more states of the environment). The application of our plan-based reward shaping technique is thus limited to domains with a well defined abstract goal state, and on a domain with this property the algorithm was evaluated. Goal-based problems are common in RL and more generally in AI, and our approach still has a wide application potential despite this limitation.

Reward Type Our approach to learning shaping rewards from multi-resolution function approximation turned out to be successful in domains with a goal-based reward type which represents an important generic type of reward in RL (Randløv 2001; Xu & Xie 2005; Torrey et al. 2008; Wingate & Seppi 2005; Epshteyn & DeJong 2006). Our preliminary experiments with the step-based reward type showed that the existing approach was not sufficiently successful. It would be interesting to see in the future work, whether this deficiency can be eliminated and how our algorithm can be generalised to a broader range of RL scenarios.

Use of Domain Knowledge It would be rather easy to find RL researchers who claim that the idea and main advantage of RL is to learn '*tabula rasa*', that is, without any background knowledge. In this thesis, we are interested however in knowledge-based improvements to RL. Knowledge proved to be important in related AI areas like informed heuristic search or domain configurable symbolic planners, and in this thesis it was shown that the use of knowledge allowed creating novel and better RL implementations. There exists work of other researchers who aim at designing algorithms which can be successful with minimal input from human expertise (see, for example, the thesis of Whiteson 2007). In our opinion, our knowledge-based approach to RL is in line with such work, because *algorithms which 'try' to learn with minimal influence of humans should be exploited in order to study which kind of representations and algorithms are suitable for specific types of domains*. With this in mind, the work of Whiteson (2007) could be seen as one the important tools to study RL algorithms in order to generalise how to match specific algorithms with domains at hand. For example, a systematic application of neural evolution could be used to learn neural networks (particularly the structure of networks) for representing a policy or a value function in a range of domains, and the analysis of obtained networks would be useful to make generalisations (scientific modelling of RL algorithms and problems) to other domains at least within the same class. With such generalisations/knowledge, the human designer of new

RL implementations would be able to make apt decisions with regard to the choice of algorithms and their parameters.

8.5 Future Work

The knowledge-based approach of this thesis can be extended to a wider perspective. A bigger picture of the use of knowledge in RL should consider, for example, techniques for knowledge revision and improvement, especially with regard to symbolic knowledge. In our work in Chapter 3, knowledge deficiencies were evaluated, but an interesting direction for further research would be to design techniques which could revise and correct abstract knowledge.

In this thesis, we were looking for correlations between domains' and algorithms' parameters, and their performance. Our guess at the end of this research is that it would be valuable to develop either qualitative or quantitative measures which would express the difficulty of the domain with regard to exploration (in particular difficulties other than the size of the state space). The rigorous experimental methodology which was applied throughout this thesis led to some promising indications which we would like to pursue in the future. In particular, we would suggest looking at global properties of the domain by treating it as a particular entity instead of using local measures which were suggested by Ratitch & Precup (2002).

As pointed out in the empirical part of Chapter 6, Rmax, MBIE and BEB algorithms have not been systematically analysed in the existing literature (even in their basic form without the use of special domain knowledge). Our preliminary results indicate that the type of the reward function will uncover interesting relationships in behaviour of these algorithms. This could further lead to the theoretical advancement in the understanding of these algorithms and their relationships and properties, especially in the context of what was said in the previous paragraph.

Applicability of our approach from Chapter 6 is relatively straightforward in domains with a PPDDL description. This representation is behind the theoretical explanation of our solution and exists in many practical RL/planning domains (Ghallab et al. 2004; Russell & Norvig 2002). PPDDL search spaces are massively broader than those in mazes, and we expect more significant improvements on these kinds of problems in future work. This line of research may lead to powerful RL solutions in the context of symbolic PPDDL representations.

In Chapter 6, AO and FSA knowledge displayed encouraging speed-up of PAC-MDP learning. This kind of knowledge is directly based on the PPDDL representation, therefore it is easy to acquire and understand. It would be interesting to see if in future work it could be shown that different (either more general or more specific) types of domain knowledge would meet requirements of PAC-MDP learning. This could be, e.g., 'feature-based heuristics' which indicate that certain actions are more promising than other actions.

This thesis is focused primarily on the case when a single agent is learning individually and the existence of possible other agents is omitted. We believe that ideas explored in this thesis could be successfully applied in the multi-agent learning scenario as well. In particular, reward

shaping could be seen as a promising research direction. It can, for example, naturally encourage cooperation or heterogenous behaviour (Tan 1993).

8.6 Final Remarks

Exploration is an immensely complicated process in RL and is influenced by numerous factors. This thesis presented a new range of methods for dealing more efficiently with the exploration-exploitation dilemma which is a crucial issue of applying reinforcement learning in practice. Novel knowledge-based methods were developed, and empirically or theoretically justified relationships between algorithms' properties and their parameters against the performance on a given family of domains were identified.

References

Altman, E. (1995). *Constrained Markov Decision Processes*. Chapman and Hall.

Anderson, C. & Crawford-Hines, S. (1994). Multigrid Q-learning. Technical Report CS-94-121, Colorado State University.

Andrecut, M. & Ali, M. K. (2004). Reinforcement learning with goal-directed eligibility traces. *International Journal of Modern Physics C*, *15*(9), 1235–1247.

Asmuth, J., Li, L., Littman, M. L., Nouri, A., & Wingate, D. (2009). A Bayesian sampling approach to exploration in reinforcement learning. In *Proceedings of the 25th Conference on Uncertainty in Artificial Intelligence*.

Asmuth, J., Littman, M. L., & Zinkov, R. (2008). Potential-based shaping in model-based reinforcement learning. In *Proceedings of AAAI Conference on Artificial Intelligence*.

Bacchus, F. & Kabanza, F. (2000). Using temporal logics to express search control knowledge for planning. *Artificial Intelligence*, *116*(1-2), 123–191.

Baird, L. (1998). Gradient descent for general reinforcement learning. In *Advances in Neural Information Processing Systems, 11*.

Barto, A. G., Bradtke, S. J., & Singh, S. P. (1995). Learning to act using real-time dynamic programming. *Artificial Intelligence*, *72*(1-2), 81–138.

Bertsekas, D. P. (2007). *Dynamic Programming and Optimal Control (2 Vol Set)*. Athena Scientific, 3rd edition.

Bertsekas, D. P. & Tsitsiklis, J. N. (1996). *Neuro-Dynamic Programming*. Athena Scientific.

Bishop, C. M. (1996). *Neural Networks for Pattern Recognition*. Oxford University Press.

Blum, A. L. & Furst, M. L. (1997). Fast planning through planning graph analysis. *Artificial Intelligence*, *90*, 281–300.

Blum, A. L. & Langford, J. C. (1998). Probabilistic planning in the graphplan framework. In *Proceedings of the Fifth European Conference on Planning*, (pp. 8–12).

Boger, J., Poupart, P., Hoey, J., Boutilier, C., Fernie, G., & Mihailidis, A. (2005). A decision-theoretic approach to task assistance for persons with dementia. In *Proceedings of International Joint Conference on Artificial Intelligence*, (pp. 1293–1299).

Böhm, N., Kókai, G., & Mandl, S. (2005). An evolutionary approach to Tetris. In *Proceedings of The Sixth Metaheuristics International Conference (MIC2005)*.

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

Boutilier, C., Dean, T., & Hanks, S. (1999). Decision-theoretic planning: Structural assumptions and computational leverage. *Journal of Artificial Intelligence Research*, *11*, 1–94.

Boyan, J. & Moore, A. (1995). Generalization in reinforcement learning: Safely approximating the value function. In *Proceedings of Neural Information Processing Systems*, (pp. 369–376).

Brafman, R. I. & Tennenholtz, M. (2002). R-max - a general polynomial time algorithm for near-optimal reinforcement learning. *Journal of Machine Learning Research*, *3*, 213–231.

Bruske, J., Ahrns, I., & Sommer, G. (1996). Practicing Q-learning. In *Proceedings of the 4th European Symposium on Artificial Neural Networks* (pp. 25–30).

Chow, C. S. & Tsitsiklis, J. N. (1991). An optimal one-way multigrid algorithm for discrete-time stochastic control. *IEEE Transactions on Automatic Control*, *36*(8), 898–914.

Cichosz, P. (1995). Truncating temporal differences: On the efficient implementation of TD(λ) for reinforcement learning. *Journal of Artificial Intelligence Research*, *2*, 287–318.

Cichosz, P. (1996). Truncated temporal differences with function approximation: Successful examples using CMAC. In *Proceedings of the 13th European Symposium on Cybernetics and Systems Research*.

Cimatti, A., Giunchiglia, F., Giunchiglia, E., & Traverso, P. (1997). Planning via model checking: A decision procedure for AR. In *Proceedings of the European Conference on Planning*, (pp. 130–142).

Cohen, P. R. (1995). *Empirical methods for artificial intelligence*. MIT Press.

Crites, R. H. & Barto, A. G. (1996). Improving elevator performance using reinforcement learning. In *Advances in Neural Information Processing Systems*, volume 8, (pp. 1017–1023).

Croonenborghs, T., Ramon, J., & Bruynooghe, M. (2004). Towards informed reinforcement learning. In *Proceedings of the ICML'04 Workshop on Relational Reinforcement Learning*.

Currie, K. & Tate, A. (1991). O-plan: the open planning architecture. *Artificial Intelligence*, *52*(1), 49–86.

Dayan, P. & Abbott, L. F. (2001). *Theoretical Neuroscience: Computational and Mathematical Modeling of Neural Systems*. The MIT Press.

Dayan, P. & Hinton, G. E. (1993). Feudal reinforcement learning. In *Proceedings of Advances in Neural Information Processing Systems*.

Dayan, P. & Niv, Y. (2008). Reinforcement learning: The good, the bad and the ugly. *Current Opinion in Neurobiology*, *18*, 185–196.

Dearden, R., Friedman, N., & Russell, S. J. (1998). Bayesian Q-learning. In *Proceedings of the Fifteenth National Conference on Artificial Intelligence*, (pp. 761–768). AAAI.

Dietterich, T. G. (2000). Hierarchical reinforcement learning with the MAXQ value function decomposition. *Journal of Artificial Intelligence Research*, *13*, 227–303.

Doherty, P., Gustafsson, J., & Karlsson, L. (1998). Tal: Temporal action logics - language specification and tutorial. *Linköping Electronic Articles in Computer and Information Science*, *3*(15).

Doherty, P. & Kvarnström, J. (2001). Talplanner: A temporal logic based planner. *AI Magazine*, *3*.

Dolgov, D. & Durfee, E. (2004). Optimal resource allocation and policy formulation in loosely-coupled markov decision processes. In *Proceedings of the 14th International Conference on Automated Planning and Scheduling*, (pp. 315–324).

Doucet, A., de Freitas, N., & Gordon, N. (Eds.). (2001). *Sequential Monte Carlo Methods in Practice*. Springer.

Duff, M. O. (2002). *Optimal learning: computational procedures for Bayes-adaptive Markov decision processes*. PhD thesis, University of Massachusetts Amherst.

Džeroski, S., Raedt, L. D., & Driessens, K. (2001). Relational reinforcement learning. *Machine Learning*, *43*, 7–52.

Epshteyn, A. & DeJong, G. (2006). Qualitative reinforcement learning. In *Proceedings of the 23rd International Conference on Machine Learning*, (pp. 305–312).

Fikes, R. & Nilsson, N. (1971). Strips: a new approach to the application of theorem proving to problem solving. *Artificial Intelligence*, *2*, 189–208.

Framling, K. (2007). Replacing eligibility trace for action-value learning with function approximation. In *Proceedings of the European Symposium on Artificial Neural Networks*.

Ghallab, M., Nau, D., & Traverso, P. (2004). *Automated Planning, Theory and Practice*. Elsevier, Morgan Kaufmann Publishers.

Gordon, G. (1995). Stable function approximation in dynamic programming. In *Proceedings of International Conference on Machine Learning*.

Grounds, M. & Kudenko, D. (2005). Combining reinforcement learning with symbolic planning. In *Fifth European Workshop on Adaptive Agents and Multi-Agent Systems*.

Grześ, M. & Kudenko, D. (2008a). Plan-based reward shaping for reinforcement learning. In *Proceedings of the 4th IEEE International Conference on Intelligent Systems (IS'08)*, (pp. 22–29). IEEE.

Grześ, M. & Kudenko, D. (2008b). Robustness analysis of SARSA(λ): Different models of reward and initialisation. In *Proceedings of the 13th International Conference on Artificial Intelligence: Methodology, Systems, Applications*, volume 5253 of *LNAI*.

Grześ, M. & Kudenko, D. (2009). Improving optimistic exploration in model-free reinforcement learning. In *Proceedings of the International Conference on Adaptive and Natural Computing*

Algorithms (ICANNGA'09), volume 5495 of *LNCS*. Springer.

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

Gullapalli, V. & Barto, A. G. (1992). Shaping as a method for accelerating reinforcement learning. In *Proceedings of the 1992 IEEE International Symposium on Intelligent Control*, (pp. 554–559).

Hoffmann, J. & Nebel, B. (2001). The ff planning system: Fast plan generation through heuristic search. *Journal of Artificial Intelligence Research*, *14*, 253–302.

Hu, Q. & Yue, W. (2007). *Markov Decision Processes with Their Applications*. Advances in Mechanics and Mathematics. Springer.

Jong, N. K. & Stone, P. (2007). Model-based exploration in continuous state spaces. In *The Seventh Symposium on Abstraction, Reformulation, and Approximation*.

Jouffe, L. (1998). Fuzzy inference system learning by reinforcement methods. *IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews*, *28*(3), 338–355.

Kaelbling, L. P. (1993). Hierarchical learning in stochastic domains: Preliminary results. In *Proceedings of International Conference on Machine Learning*, (pp. 167–173).

Kaelbling, L. P., Littman, M. L., & Moore, A. P. (1996). Reinforcement learning: A survey. *Journal of Artificial Intelligence Research*, *4*, 237–285.

Kakade, S. M. (2003). *On the Sample Complexity of Reinforcement Learning*. PhD thesis, Gatsby Computational Neuroscience Unit, University College, London.

Kearns, M. & Singh, S. (2002). Near-optimal reinforcement learning in polynomial time. *Machine Learning*, *49*, 209–232.

Kearns, M. J. & Vazirani, U. V. (1994). *An Introduction to Computational Learning Theory*. The MIT Press.

Kohl, N. & Stone, P. (2004). Policy gradient reinforcement learning for fast quadrupedal locomotion. In *Proceedings of the IEEE International Conference on Robotics and Automation*.

Kolter, J. Z. & Ng, A. (2009). Near-Bayesian exploration in polynomial time. In *Proceedings of ICML*.

Kuhlmann, G. & Stone, P. (2004). Progress in learning 3 vs. 2 keepaway. In Polani, D., Browning, B., Bonarini, A., & Yoshida, K. (Eds.), *RoboCup-2003: Robot Soccer World Cup VII*, Berlin. Springer Verlag.

Kuter, U. & Nau, D. (2005). Using domain-configurable search control for probabilistic planning. In *Proceedings of the National Conference on Artificial Intelligence (AAAI)*.

Lazaric, A., Restelli, M., & Bonarini, A. (2007). Reinforcement learning in continuous action spaces through sequential Monte Carlo methods. In *Proceeding of Neural Information Processing Systems*.

Leng, J., Fyfe, C., & Jain, L. C. (2009). Experimental analysis on sarsa(λ) and q(λ) with different eligibility traces strategies. *Journal of Intelligent and Fuzzy Systems*, *20*(1-2), 73–82.

Lin, C.-S. & Kim, H. (1991). CMAC-based adaptive critic self-learning control. *IEEE Transactions on Neural Networks*, *2*, 530–533.

Lin, L.-J. (1992). Self-improving reactive agents based on reinforcement learning, planning and teaching. *Machine Learning*, *8*, 293–321.

Littman, M. L. (1994). Markov games as a framework for multi-agent reinforcement learning. In *Proceedings of the 11th International Conference on Machine Learning*, (pp. 157–163). Morgan Kaufmann.

Loch, J. & Singh, S. (1998). Using eligibility traces to find the best memoryless policy in partially observable Markov Decision Processes. In *Proceedings of the 15th International Conference on Machine Learning*, (pp. 323–331).

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

Mataric, M. J. (1994). Reward functions for accelerated learning. In *In Proceedings of the 11th International Conference on Machine Learning*, (pp. 181–189).

Melo, F. S., Meyn, S. P., & Ribeiro, M. I. (2008). An analysis of reinforcement learning with function approximation. In *Proceedings of International Conference on Machine Learning*, (pp. 664–671).

Meuleau, N., Peshkin, L., eung Kim, K., & Kaelbling, L. P. (1999). Learning finite-state controllers for partially observable environments. In *Proceedings of the fifteenth conference on uncertainty in artificial intelligence*, (pp. 427–436).

Mitchell, T. M. (1997). *Machine Learning*. McGraw-Hill.

Moore, A., Baird, L., & Kaelbling, L. P. (1999). Multi-value-functions: Efficient automatic action hierarchies for multiple goal MDPs. In *Proceedings of the International Joint Conference on Artificial Intelligence*, (pp. 1316–1323).

Moriarty, D. E., Schultz, A. C., & Grefenstette, J. J. (1999). Evolutionary algorithms for reinforcement learning. *Journal of Artificial Intelligence Research*, *11*, 199–229.

Munos, R. & Moore, A. (2002). Variable resolution discretization in optimal control. *Machine Learning*, *49*(2-3), 291–323.

Nau, D. (2003). Shop2: An htn planning system. *Journal of Artificial Intelligence Research*, *20*, 379–404.

Nau, D. S. (2007). Current trends in automated planning. *AI Magazine*, *28*(4), 43.

Ng, A. Y., Harada, D., & Russell, S. J. (1999). Policy invariance under reward transformations: Theory and application to reward shaping. In *Proceedings of the 16th International Conference on Machine Learning*, (pp. 278–287).

Ng, A. Y. & Jordan, M. (2000). PEGASUS: A policy search method for large MDPs and POMDPs. In *In Proceedings of Uncertainty in Artificial Intelligence*, (pp. 406–415).

Parr, R. & Russell, S. (1997). Reinforcement learning with hierarchies of machines. In *Proceedings of Advances in Neural Information Processing Systems*, volume 10.

Pasula, H. M., Zettlemoyer, L. S., & Kaelbling, L. P. (2007). Learning symbolic models of stochastic domains. *Journal of Artificial Intelligence Research*, *29*, 309–352.

Peng, J. & Williams, R. J. (1996). Incremental multi-step Q-learning. *Machine Learning*, *22*, 283–290.

Pohl, I. (1970). Heuristic search viewed as path finding in a graph. *Artificial Intelligence*, *1*, 193–204. (Weighted A* was introduced in this paper for the first time).

Poupart, P., Vlassis, N., Hoey, J., & Regan, K. (2006). An analytic solution to discrete bayesian reinforcement learning. In *Proceedings of International Conference on Machine Learning*, (pp. 697–704).

Preux, P. (2002). Propagation of q-values in tabular td(lambda). In *Proc of ECML*.

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

Pérez-Uribe, A. & Sanchez, E. (1999). A comparison of reinforcement learning with eligibility traces and integrated learning, planning and reacting. In M. Mohammadian (Ed.), *Computational Intelligence for Modelling, Control and Automation* (pp. 154–159). IOS Press.

Randløv, J. (2001). *Solving Complex Problems with Reinforcement Learning*. PhD thesis, University of Copenhagen.

Randløv, J. & Alstrom, P. (1998). Learning to drive a bicycle using reinforcement learning and shaping. In *Proceedings of the 15th International Conference on Machine Learning*, (pp. 463–471).

Ratitch, B. & Precup, D. (2002). Characterizing Markov decision processes. In *Proceedings of the European Conference on Machine Learning*.

Rayner, D. C., Davison, K., Bulitko, V., Anderson, K., & Lu, J. (2007). Real-time heuristic search with a priority queue. In *Proceedings of the 2007 International Joint Conference on Artificial Intelligence*, (pp. 2372–2377).

Reynolds, S. I. & Wiering, M. A. (2002). Fast q(lambda) revisited. Technical Report CSRP-02-02, School of Computer Science, The University of Birmingham, Birmingham, UK.

Riedmiller, M. (2005). Neural fitted Q iteration - first experiences with a data efficient neural reinforcement learning method. In *Proceedings of the European Conference on Machine Learning*, (pp. 317–328).

Rintanen, J. (2000). Incorporation of temporal logic control into plan operators. In *Proceedings of The European Conference on Artificial Intelligence*, (pp. 526–530).

Russell, S. J. & Norvig, P. (2002). *Artificial Intelligence: A Modern Approach (2nd Edition)*. Prentice Hall.

Ryan, M. R. K. (2002). Using abstract models of behaviours to automatically generate reinforcement learning hierarchies. In *Proceedings of the 19th International Conference on Machine Learning*, (pp. 522–529).

Samuel, A. L. (1959). Some studies in machine learning using the game of checkers. *IGM*

Journal on Research and Development, *3*, 210–229.

Schmidhuber, J. (2001). Sequential decision making based on direct search. In R. Sun & C. L. Giles (Eds.), *Sequence Learning: Paradigms, Algorithms, and Applications*, volume 1828 of *LNAI*. Springer.

Sherstov, A. A. & Stone, P. (2005). Function approximation via tile coding: Automating parameter choice. In *Symposium on Abstraction, Reformulation, and Approximation*, (pp. 194–205).

Singh, S. P. & Sutton, R. S. (1996). Reinforcement learning with replacing eligibility traces. *Machine Learning*, *22*(1–3), 123–158.

Stanley, K. O. (2004). *Efficient Evolution of Neural Networks Through Complexification*. PhD thesis, Department of Computer Sciences, The University of Texas at Austin, Austin, TX.

Stone, P. & Sutton, R. S. (2001). Scaling reinforcement learning toward robocup soccer. In *The 18th International Conference on Machine Learning*, (pp. 537–544). Morgan Kaufmann, San Francisco, CA.

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

Stone, P. & Veloso, M. (2000). Layered learning. In *Proceedings of the 11th European Conference on Machine Learning*.

Strehl, A. L., Li, L., & Littman, M. L. (2006). Pac reinforcement learning bounds for rtdp and rand-rtdp. In *Proceedings of AAAI Workshop on Learning for Search*.

Strehl, A. L., Li, L., & Littman, M. L. (2009). Reinforcement learning in finite MDPs: PAC analysis. *Journal of Machine Learning Research*, *10*, 2413–2444.

Strehl, A. L., Li, L., Wiewiora, E., Langford, J., & Littman, M. L. (2006). Pac model-free reinforcement learning. In *Proceedings of the 23rd International Conference on Machine Learning*, (pp. 881–888)., New York, NY, USA. ACM.

Strehl, A. L. & Littman, M. L. (2008). An analysis of model-based interval estimation for Markov decision processes. *Journal of Computer and System Sciences*, *74*, 1309–1331.

Strens, M. J. A. (2000). A Bayesian framework for reinforcement learning. In *Proceedings of the 17th International Conference on Machine Learning*, (pp. 943–950).

Sutton, R. (1988). Learning to predict by the methods of temporal differences. *Machine Learning*, *3*, 9–44.

Sutton, R. & Singh, S. P. (1994). On step-size and bias in temporal-difference learning. In *Center for Systems Science, Yale University*, (pp. 91–96).

Sutton, R. S. (1984). *Temporal credit assignment in reinforcement learning*. PhD thesis, Department of Computer Science, University of Massachusetts, Amherst.

Sutton, R. S. (1990). Integrated architectures for learning, planning, and reacting based on approximating dynamic programming. In *Proceedings of the 7th International Conference on Machine Learning*, (pp. 216–224).

Sutton, R. S. (1996). Generalization in reinforcement learning: Successful examples using sparse

coarse coding. In *Advances in Neural Information Processing Systems*, volume 8, (pp. 1038–1044).

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

Sutton, R. S., Mcallester, D., Singh, S., & Mansour, Y. (1999). Policy gradient methods for reinforcement learning with function approximation. In *In Advances in Neural Information Processing Systems 12*, (pp. 1057–1063). MIT Press.

Sutton, R. S., Precup, D., & Singh, S. P. (1999). Between MDPs and Semi-MDPs: A framework for temporal abstraction in reinforcement learning. *Artificial Intelligence*, *112*(1-2), 181–211.

Szepesvari, C. (2009). Reinforcement learning algorithms for MDPs. Technical Report TR09-13, Department of Computing Science, University of Alberta.

Szita, I. & Lörincz, A. (2006). Learning Tetris using the noisy cross-entropy method. *Neural Computation*, *18*(12), 2936–2941.

Tan, M. (1993). Multi-agent reinforcement learning: Independent versus cooperative agents. In *Proceedings of ICML*, (pp. 330–337).

Taylor, M. E., Whiteson, S., & Stone, P. (2006). Comparing evolutionary and temporal difference methods in a reinforcement learning domain. In *GECCO 2006: Proceedings of the Genetic and Evolutionary Computation Conference*, (pp. 1321–1328).

Tesauro, G. (1992). Practical issues in temporal difference learning. *Machine Learning*, *8*, 257–277.

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

Thayer, J. T. & Ruml, W. (2008). Faster than weighted A*: An optimistic approach to bounded suboptimal search. In *Proceedings of the Eighteenth International Conference on Automated Planning and Scheduling*.

Thrun, S. (1992). Efficient exploration in reinforcement learning. Technical Report CMU-CS-92-102, Carnegie Mellon University, Computer Science Department.

Torrey, L., Shavlik, J., Natarajan, S., Kuppili, P., & Walker, T. (2008). Transfer in reinforcement learning via Markov logic networks. In *Proceedings of the AAAI'08 Workshop on Transfer Learning for Complex Tasks*.

van Eck, N. J. & van Wezel, M. (2008). Application of reinforcement learning to the game of Othello. *Computers & Operations Research*, *35*, 1999–2017.

Wallace, N. (2004). Hierarchical planning in dynamic worlds. In S. Rabi (Ed.), *AI Game Programming Wisdom 2*. Charles River Media.

Whiteson, S. (2007). *Adaptive Representations for Reinforcement Learning*. PhD thesis, Department of Computer Sciences, University of Texas at Austin.

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

Wiewiora, E. W., Cottrell, G., & Elkan, C. (2003). Principled methods for advising reinforcement

learning agents. In *Proceedings of the 20th International Conference on Machine Learning*.

Wilkins, D. & desJardins, M. (2001). A call for knowledge-based planning. *AI Magazine*, *22*, 99–115.

Williams, R. J. (1992). Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning*, *8*, 229–256.

Wingate, D. & Seppi, K. D. (2005). Prioritization methods for accelerating MDP solvers. *Journal of Machine Learning Research*, *6*, 851–881.

Wu, C. & Meleis, W. (2008). Adaptive Kanerva-based function approximation for multi-agent systems (short paper). In *Proc. of 7th Int. Conf. on Autonomous Agents and Multiagent Systems (AAMAS 2008)*, (pp. 1357–1360).

Wu, C. & Meleis, W. (2009a). Adaptive fuzzy function approximation for multi-agent reinforcement learning. In *Proceedings of IEEE/WIC/ACM International Conference on Intelligent Agent Technology (IAT)*.

Wu, C. & Meleis, W. (2009b). Fuzzy Kanerva-based function approximation for reinforcement learning (short paper). In *Proc. of 8th Int. Conf. on Autonomous Agents and Multiagent Systems (AAMAS 2009)*.

Wyatt, J., Hayes, G., & Hallam, J. (1999). Investigating the behaviour of Q(λ). In *Proceedings of the Colloquium on Self-Learning Robots*.

Wyatt, J. L. (2001). Exploration control in reinforcement learning using optimistic model selection. In *Proceedings of the 18th International Conference on Machine Learning*, (pp. 593–600).

Xu, X. & Xie, T. (2005). A reinforcement learning approach for host-based intrusion detection using sequences of system calls. In *Proceedings of International Conference on Intelligent Computing*, (pp. 995–1003).

Yoon, S. W., Fern, A., & Givan, R. (2007). FF-replan: A baseline for probabilistic planning. In *Proceedings of the International Conference on Automated Planning and Scheduling (ICAPS)*, (pp. 352–359).

Younes, H. L. S. & Littman, M. L. (2004). PPDDL1.0: An extension to PPDDL for expressing planning domains with probabilistic effects. Technical Report CMU-CS-04-162, Carnegie Mellon University.

Zheng, Y., Luo, S., & Lv, Z. (2006). Control double inverted pendulum by reinforcement learning with double CMAC network. In *The 18th International Conference on Pattern Recognition*, (pp. 639–642). IEEE Computer Society.

Zhu, W. & Levinson, S. (2002). PQ-learning: an efficient robot learning method for intelligent behavior acquisition. In *Proceedings of the 7th International Conference on Intelligent Autonomous Systems*.

Index

ϵ-greedy, 32

agent, 25

backtracking, 100
basis functions, 37
belief state, 31
Boltzman, 32

conceptual knowledge, 39

discount factor, 98, 113, 146

eligibility traces, **34**, 40, 142, 143, 172
    accumulating, 143
    replacing, 143
experience replay, 37
exploration
    optimistic, 33, 144
    pessimistic, 33, 144
exploration based on the current policy, 33
exploration-exploitation, 19, 171

function approximation, 36
    Kanerva coding, 66
    radial basis functions, 66
    tile coding, 66, 73

heuristic
    admissible heuristic, 46
Hoeffding bound, 124

Kanerva coding, 69, 174
knowledge about the state representation, 38

Markov property, 19, **26**, 33, 37, 142
Markov Reward Processes, 141
MDP, **28**

PAC-MDP, 32
policy, 18, 25, 26
POMDP, 31
potential function, 44, 113, 119
PPDDL, 126, 128, 129, 139, 178
procedural knowledge, 39

Q-learning, 30

reinforcement learning, 26
reinforcement learning algorithms, 26
reinforcement learning problem, 26
reward, 98
    reward shaping, 19, 21, **34**, 40, 42, 97, 113, 172

SARSA(λ), 34, 143
state space explosion, 19, 171

STRIPS, 44
symbolic planning, 37

temporal credit assignment problem, 19, 33,
142, 171

Citation Index

Altman (1995), 96
Anderson & Crawford-Hines (1994), 75
Andrecut & Ali (2004), 166
Asmuth et al. (2008), 21, 36, 67, 97, 99, 120, 129, 132
Asmuth et al. (2009), 31, 32
Böhm et al. (2005), 28
Bacchus & Kabanza (2000), 20
Baird (1998), 28
Barto et al. (1995), 133
Bertsekas & Tsitsiklis (1996), 28, 30
Bertsekas (2007), 29, 30
Bishop (1996), 35–38, 66, 73
Blum & Furst (1997), 38, 44
Blum & Langford (1998), 38, 126
Boger et al. (2005), 124, 126, 128
Boutilier et al. (1999), 20, 43, 44, 69
Boutilier (1999), 96
Boyan & Moore (1995), 170
Brafman & Tennenholtz (2002), 30–32, 36, 67, 68, 123, 124, 126, 130
Bruske et al. (1996), 143
Chow & Tsitsiklis (1991), 75
Cichosz (1995), 81–83, 142, 145, 168, 169, 179, 180
Cichosz (1996), 82, 83, 142, 168, 179
Cimatti et al. (1997), 20
Cohen (1995), 49, 133
Crites & Barto (1996), 32
Croonenborghs et al. (2004), 97
Currie & Tate (1991), 20
Džeroski et al. (2001), 97
Dayan & Abbott (2001), 38
Dayan & Hinton (1993), 77
Dayan & Niv (2008), 38
Dearden et al. (1998), 47
Dietterich (2000), 38, 77, 96
Doherty & Kvarnström (2001), 20
Doherty et al. (1998), 21
Dolgov & Durfee (2004), 96
Doucet et al. (2001), 31
Duff (2002), 31
Epshteyn & DeJong (2006), 181
Fikes & Nilsson (1971), 43
Framling (2007), 142, 159, 168, 179
Ghallab et al. (2004), 20, 37, 58, 140, 182
Gordon (1995), 37, 170
Grounds & Kudenko (2005), 43
Grześ & Kudenko (2008a), 67, 97
Grześ & Kudenko (2008b), 142

Grześ & Kudenko (2009), 120, 161, 170
Grześ & Kudenko (2010), 23
Gullapalli & Barto (1992), 97
Hoffmann & Nebel (2001), 38
Hu & Yue (2007), 78
Jong & Stone (2007), 146
Jouffe (1998), 84
Kaelbling et al. (1996), 25
Kaelbling (1993), 77
Kakade (2003), 67, 125
Kearns & Singh (2002), 123, 124
Kearns & Vazirani (1994), 124
Kohl & Stone (2004), 28
Kolter & Ng (2009), 31, 124–126, 132, 133, 136, 139, 178
Kuhlmann & Stone (2004), 170
Kuter & Nau (2005), 20
Lazaric et al. (2007), 84, 93
Leng et al. (2009), 142, 152, 159, 168, 179
Lin & Kim (1991), 38, 72
Lin (1992), 37
Littman (1994), 96
Loch & Singh (1998), 33, 142
Marthi (2007), 43, 46, 68, 75, 76, 173
Mataric (1994), 98
Melo et al. (2008), 37
Meuleau et al. (1999), 44, 64, 67
Mitchell (1997), 20, 39, 72, 96
Moore et al. (1999), 77
Moriarty et al. (1999), 28
Munos & Moore (2002), 76, 77, 82
Nau (2003), 20
Nau (2007), 20, 38
Ng & Jordan (2000), 28
Ng et al. (1999), 21, 35, 40, 98, 103, 114–116, 119, 120, 122, 129, 176, 178
Pérez-Uribe & Sanchez (1999), 33, 142
Parr & Russell (1997), 38, 77
Pasula et al. (2007), 47
Peng & Williams (1996), 143, 169
Pohl (1970), 40
Poupart et al. (2006), 31, 124, 126
Preux (2002), 142, 168, 179
Puterman (1994), 28, 96, 97, 142
Randløv & Alstrom (1998), 21, 97, 116, 129
Randløv (2001), 35, 181
Ratitch & Precup (2002), 182
Rayner et al. (2007), 128
Reynolds & Wiering (2002), 170
Riedmiller (2005), 37
Rintanen (2000), 20
Russell & Norvig (2002), 20, 28, 36, 37, 46, 59, 97, 98, 140, 159, 182
Ryan (2002), 43, 47
Samuel (1959), 30
Schmidhuber (2001), 28
Sherstov & Stone (2005), 77
Singh & Sutton (1996), 81, 141, 143, 144, 166, 170
Stanley (2004), 28
Stone & Sutton (2001), 44, 152
Stone & Veloso (2000), 77
Stone et al. (2005), 44, 69, 76, 94, 152, 170, 175
Strehl & Littman (2008), 67, 125–127, 130, 132
Strehl et al. (2006), 33, 36
Strehl et al. (2009), 32, 41
Strens (2000), 32, 47
Sutton & Barto (1998), 18, 22, 25, 27, 29, 30, 34, 35, 48, 66, 72, 77, 80–82, 100, 101, 141, 143–146, 148, 166, 169, 170
Sutton & Singh (1994), 141
Sutton et al. (1999), 28, 44, 77
Sutton (1984), 33, 71, 142

Sutton (1988), 33, 141, 142, 166
Sutton (1990), 30, 31, 67, 68, 147
Sutton (1996), 36, 72, 73
Szepesvari (2009), 37, 141
Szita & Lörincz (2006), 28
Tan (1993), 183
Taylor et al. (2006), 28
Tesauro (1992), 80, 145
Tesauro (1994), 30, 32, 66, 98
Thayer & Ruml (2008), 40
Thrun (1992), 32
Torrey et al. (2008), 181
Wallace (2004), 20
Whiteson (2007), 26, 28, 181
Wiewiora et al. (2003), 97
Wiewiora (2003), 35, 98, 103, 122, 176, 178
Wilkins & desJardins (2001), 20, 39
Williams (1992), 28
Wingate & Seppi (2005), 181
Wu & Meleis (2008), 66, 69, 174
Wu & Meleis (2009a), 66, 69, 174
Wu & Meleis (2009b), 66, 69, 174
Wyatt et al. (1999), 142, 168, 179
Wyatt (2001), 32
Xu & Xie (2005), 181
Yoon et al. (2007), 126, 127
Younes & Littman (2004), 126
Zheng et al. (2006), 76
Zhu & Levinson (2002), 142
van Eck & van Wezel (2008), 28

List of Symbols

α	learning rate, see equation (2.5), page 30
δ	temporal difference, see equation (2.7), page 34
ϵ	probability of executing a non-greedy action in the ϵ-greedy exploration strategy, page 32
γ	MDP discount factor, page 28
$\hat{Q}$	approximate state-action value function, see equation (6.1), page 125
$\hat{V}$	approximate state value function, see equation (6.1), page 125
λ	scaling factor for decreasing eligibility $e(s, a)$, page 34
$\mathbb{A}$	set of actions, page 28
$\mathbb{S}$	set of MDP states, page 28
Φ	potential function, see equation (2.11), page 35
π	policy, page 25
π^*	optimal policy, page 25
τ	scaling factor to scale the shaping reward, page 101
θ	parameter vector for a policy, π, or for the value function, V, page 28
B	exploration bonus, see equation (6.1), page 125
$e(s, a)$	eligibility of the state action pair (s, a), see equation (2.9), page 34

F shaping reward, see equation (2.10), page 34

M MDP model, page 116

M' augmented MDP model, e.g. with reward shaping, page 98

Q state-action value function, page 29

Q^* optimal state-action value function, page 29

R reward function, page 28

T state transition function, page 28

V state value function, page 29

V^* optimal state value function, page 29