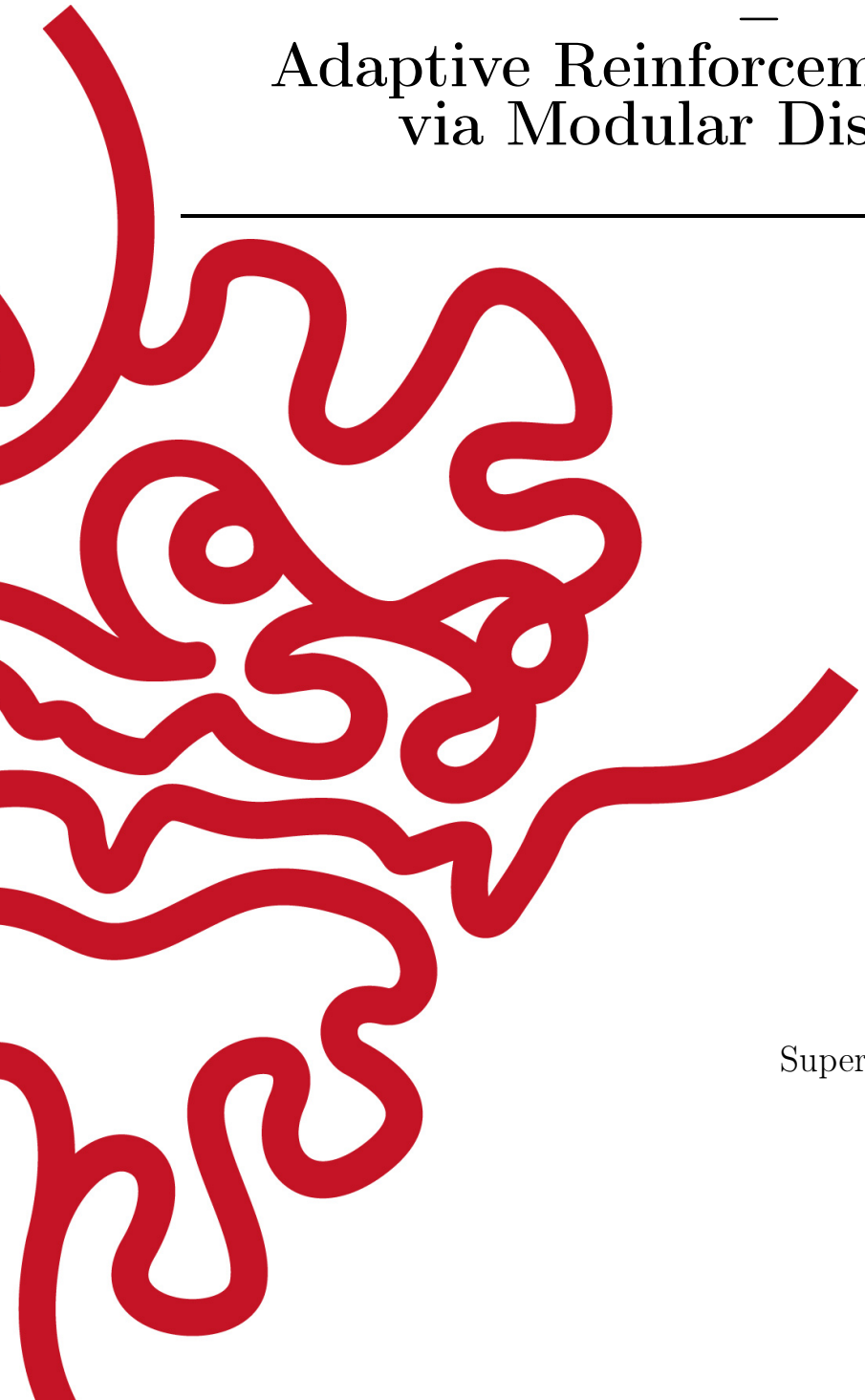
Okinawa Institute of Science and Technology
Graduate University

Thesis submitted for the degree

Doctor of Philosophy

The Gamma-Ensemble
—
Adaptive Reinforcement Learning via Modular Discounting



by

Chris Reinke

Supervisor: **Prof. Dr. Kenji Doya**

June, 2018

Declaration of Original and Sole Authorship

I, Chris Reinke, declare that this thesis entitled *The Gamma-Ensemble - Adaptive Reinforcement Learning via Modular Discounting* and the data presented in it are original and my own work.

I confirm that:

- No part of this work has previously been submitted for a degree at this or any other university.
- References to the work of others have been clearly acknowledged. Quotations from the work of others have been clearly indicated, and attributed to them.
- In cases where others have contributed to part of this work, such contribution has been clearly acknowledged and distinguished from my own work.
- None of this work has been previously published elsewhere, with the exception of the following: Reinke, Uchibe, & Doya 2015; 2017

Date: June, 2018
Signature:

Abstract

The Gamma-Ensemble
Adaptive Reinforcement Learning via Modular Discounting

Reinforcement learning allows artificial agents to learn complex tasks, such as playing Go on an expert level. Still, unlike humans, artificial agents lack the ability to adapt learned behavior to task changes, or to new objectives, such as to capture as many opponent pieces within a given number of moves, instead of simply winning. The Independent Gamma-Ensemble (IGE), a new brain-inspired framework, allows such adaptations. It is composed of several Q-learning modules, each with a different discount factor. The off-policy nature of Q-learning allows modules to learn several policies in parallel, each representing a different solution for the payoff between a high reward sum and the time to gain it. The IGE adapts to new task conditions by switching between its policies (transfer learning). It can also decode the expected reward sum and the required time for each policy, allowing it to immediately select the most appropriate policy for a new task objective (zero-shot learning). Additionally, this allows to optimize the average reward in discrete MDPs where non-zero reward is only given in goal states. The convergence to the optimal policy can be proven for such MDPs. The modular structure behind the IGE can be combined with many reinforcement learning algorithms and applied to various tasks, allowing to improve the adaptive abilities of artificial agents in general.

Acknowledgment

The support of various people made it possible for me to undertake my Ph.D. Without their help, I would not be where I am today. First and foremost, I want to thank my parents and my family for their love and encouragement throughout my long academic journey.

Ich danke meinen Eltern und meiner Familie und für ihre innige und dauerhafte Unterstützung. Ich konnte während meines gesamten akademischen Weges auf ihre Hilfe und Anerkennung bauen. Trotzalledem, dass Okinawa eine grosse Distanz zwischen uns herstellte haben sie mich unterstützt und bekräftigt diese Herausforderung anzunehmen und zu meistern.

I am extremely grateful to my Ph.D. supervisor, Kenji Doya. He gave me the freedom to follow my own ideas and the academic support to succeed. His supervision and scientific input was always very helpful and inspiring. I am also grateful for the mentoring of Eiji Uchibe, who worked as a Group Leader in the Neural Computations Unit. His rich knowledge of reinforcement learning and machine learning was extremely helpful in our many discussions. I thank Kenji and Eiji not only for their supervision of this thesis, but also for their guidance in writing other publications (Reinke, Uchibe, & Doya 2015; 2017). I also want to acknowledge my fellow students in the Neural Computations Unit for our discussions and their patience in enduring my often lengthy progress reports at lab meetings. I especially thank Tadashi Kozuno, who helped me with earlier proofs regarding the optimality of my framework for average-reward optimization, which were used in one of my publications (Reinke, Uchibe, & Doya 2017). I also want to mention Hiroaki Hamada, who is a good friend and helped me with many daily issues, making my life in Japan easier. Not to forget are the research administrators of the Neural Computation Unit, Emiko Asato, Kikuko Matsuo, and Misuzu Saito, who also helped me a lot to handle my life in Japan. I enjoyed my stay in the Neural Computation Unit and I collected many fond memories thanks to my wonderful colleagues.

Many thanks also to my mentor, Gail Tripp. She had always an open ear for my problems and supported me during some difficult times. I am also very thankful for her proofreading of the thesis. I want to also thank Nathaniel Daw and Nic Shannon, the examiners at my Ph.D. proposal defense. Nathaniel made the long trip from the USA to Okinawa and gave me valuable comments in regard to my project. Furthermore, I want to thank my examiners of the Ph.D. thesis, Sridhar Mahadevan and Mathew Taylor. They also took long trips from the USA and Canada to come for my defense and provided insightful comments helping to improve the thesis. Many thanks also to Bernd Kuhn, who was a member of my Ph.D. thesis committee and who supported me throughout my Ph.D. I would also like to thank Saori Tanaka for her valuable discussions about her

research that gave me inspiration for the γ-Ensemble.

I also want to thank the staff of the Graduate School, the former Dean, Jeff Wickens, who was the chair in my Ph.D. proposal examination, and the current Dean Ulf Skoglund, who was the chair in my final Ph.D. examination. The Graduate School not only supported me financially, but also assisted me in other aspects of my daily life in Japan, allowing me to focus on academics. Their efforts made my stay in Okinawa very comfortable, for which I am very grateful. And last, but not least, I want to mention Steven Aird, who is the technical editor at OIST. He, together with my mentor Gail Tripp, spend a lot of time de-Germanizing my grammar in this thesis. Steve also edited my other publications (Reinke, Uchibe, & Doya 2015; 2017).

Abbreviations

AR-IGE	Average Reward Independent γ-Ensemble
CA-IGE	Context Adaptive Independent γ-Ensemble
CMDP	Context Markov Decision Process
CP-IGE	Constant Punishment Independent γ-Ensemble
CP-MDP	Constant Punishment Markov Decision Process
DOA-IGE	Decoding Objective Adaptive Independent γ-Ensemble
DGE	Dependent γ-Ensemble
IGE	Independent γ-Ensemble
IR-IGE	Interruption Risk Independent γ-Ensemble
IR-MDP	Interruption Risk Markov Decision Process
MDP	Markov Decision Process
MOMDP	Multi-Objective Markov Decision Process
MORL	Multi-Objective Reinforcement Learning
OA-IGE	Objective Adaptive Independent γ-Ensemble
OA-MDP	Objective Adaptive Markov Decision Process
POA-IGE	Prediction Objective Adaptive Independent γ-Ensemble
POMDP	Partially Observable Markov Decision Process
RR-IGE	Reward Risk Independent γ-Ensemble
RR-MDP	Reward Risk Markov Decision Process
SMDP	Semi Markov Decision Process
TD	Temporal Difference

Notation

General Math Notations

$\lfloor x \rfloor$	Floor of x, i.e. round down to nearest integer
$\lceil x \rceil$	Ceiling of x, i.e. round up to nearest integer
$\mathbb{I}(x)$	Indicator function, i.e. $\mathbb{I}(x) = 1$ if x is true, otherwise $\mathbb{I}(x) = 0$
$\lvert x \rvert$	Number of elements in vector x
$x^\intercal$	Transpose of vector x
∞	Infinity
$\leftarrow$	Setting of a variable, e.g. $x \leftarrow 1$
$\rightarrow$	Tends towards, e.g. $x \rightarrow \infty$
$\forall x : y$	"For all" x holds y
$\exists x : y$	It "Exists" at least one x such that y holds
∇	Vector of first derivatives
$\mathbb{N}$	Set of natural numbers
$\mathbb{R}$	Set of real numbers
$\mathcal{L}(\theta; D)$	Likelihood function
$\Pr(x)$	Probability function
$\Pr(x \lvert y)$	Conditional probability function
$\mathrm{E}[x]$	Expectation over x
$\mathrm{E}_y[x]$	Expectation over x with respect to y
$\mathcal{U}(x, y)$	Uniform distribution between x and y

Reinforcement Learning Notations

α	Learning rate
γ	Discount factor
Γ	Set of discount factors
β	Exploration factor of the softmax action selection
ϵ	Exploration rate of the ϵ-Greedy action selection
A	Finite set of actions
S	Finite set of states
S^G	Set of goal/terminal states
$\pi(s)$, $\pi(a; s)$	Policy
$Q(s, a)$	Q-function
$Q_\gamma(s, a)$	Q-Gamma function

$R(s,a), R(s,a,s')$	Reward functions
$T(s,a,s')$	Transition probability function
$V(s)$	Value function
$V_\gamma(s)$	Value-Gamma function

Contents

Declaration of Original and Sole Authorship iii

Abstract v

Acknowledgment vii

Abbreviations ix

Notation xi

Contents xiii

List of Algorithms xvii

List of Figures xix

List of Tables xxiii

Introduction 1

1 Fundamentals and Related Work 5

1.1 Reinforcement Learning 5

1.1.1 Markov Decision Processes 6

1.1.2 Overview over Reinforcement Learning Algorithms 9

1.1.3 Model-Free, Value-Based Algorithms 10

1.2 Transfer Learning 15

1.2.1 Transfer of Policies 15

1.2.2 Transfer of Skills and Options 16

1.3 The Horde Architecture 20

1.4 Inspiration from Neuroscience 22

1.4.1 Model-Free Reinforcement Learning in the Brain 23

1.4.2 Modular Discounting Structure in the Brain 24

1.4.3 The Dependent γ-Ensemble 24

1.5 Conclusion 27

2 The Independent γ-Ensemble **29**
2.1 The IGE Framework . . . 29
2.2 Learning a Set of Policies . . . 30
2.2.1 Type of Learned Reward Trajectories . . . 32
2.2.2 Type of Learned Policies in General MDPs . . . 34
2.2.3 Type of Learned Policies in MDPs with several Goal States . . . 35
2.3 Decoding Information about Policies . . . 42
2.3.1 General MDPs . . . 42
2.3.2 Deterministic, Goal-Only-Reward MDPs . . . 44
2.4 Optimal Distribution of γ-Modules . . . 45
2.5 Effect of the Discount Factor on the Exploration . . . 50
2.6 Conclusion . . . 51

3 Adaptation to Contextually Changing Tasks **53**
3.1 Context Adaptive Markov Decision Processes . . . 54
3.2 The Context Adaptive Independent γ-Ensemble . . . 56
3.2.1 The CA-IGE Framework . . . 57
3.2.2 Application Scenarios and Optimality . . . 58
3.3 Derivation of γ-Mappings . . . 62
3.3.1 Interruption Risk Tasks . . . 62
3.3.2 Reward Risk Tasks . . . 68
3.3.3 Constant Punishment Tasks . . . 75
3.4 Learning of γ-Mappings . . . 80
3.4.1 Algorithms to Learn γ-Mappings . . . 81
3.4.2 Experimental Design . . . 87
3.4.3 Experimental Results . . . 92
3.4.4 Discussion . . . 100
3.5 Conclusion . . . 102

4 Adaptation to Changing Objectives **105**
4.1 Objective Adaptive Markov Decision Processes . . . 106
4.2 The Objective Adaptive Independent γ-Ensemble . . . 107
4.3 Decoding of Trajectories . . . 111
4.3.1 The Decoding OA-IGE . . . 111
4.3.2 Deterministic, Goal-Only-Reward Tasks . . . 113
4.3.3 Conclusion . . . 119
4.4 Prediction of Trajectory Statistics . . . 119
4.4.1 The Trajectory Prediction OA-IGE . . . 119
4.4.2 Stochastic, Goal-Only-Reward Tasks . . . 121
4.4.3 General Tasks . . . 124
4.4.4 Conclusion . . . 128
4.5 Discussion . . . 128
4.6 Conclusion . . . 134

5 Average Reward Optimization **135**
5.1 Existing Average-Adjusted Approaches . . . 136
5.1.1 Average-Reward Optimization in Episodic MDPs . . . 138
5.1.2 Episodic Average-Adjusted Algorithms . . . 140
5.1.3 Conclusion . . . 140
5.2 The Average-Reward Independent γ-Ensemble . . . 141
5.2.1 Existence of the Optimal Policy . . . 143
5.2.2 Alternative Geometric Proof of Optimality . . . 145
5.3 The Decoding Average-Reward IGE . . . 146
5.3.1 The DAR-IGE Algorithm . . . 147
5.3.2 Proof of Optimality . . . 148
5.4 The Value-Quotient Average-Reward IGE . . . 149
5.4.1 The VQAR-IGE Algorithm . . . 149
5.4.2 Proof of Optimality . . . 151
5.5 Experimental Evaluation . . . 154
5.5.1 Task Design . . . 154
5.5.2 Learning Parameters . . . 156
5.5.3 Experimental Results . . . 158
5.5.4 Conclusion . . . 163
5.6 Discussion . . . 163
5.6.1 Average Reward per Episode vs Over All Steps . . . 163
5.6.2 Relationship between Number of γ-Modules and Performance . . . 164
5.7 Conclusion . . . 171

Conclusion **173**

A Supplementary Data for the Context Adaptive Energy Foraging Task **181**

B Supplementary Data for the Average Reward Experiments **189**

C Analytical Solution to Theorem 21 **193**

Bibliography **195**

List of Algorithms

1.1 Q-learning for discrete, episodic MDPs 14

2.1 Independent γ-Ensemble 31

3.1 Interruption Risk IGE 64
3.2 Reward Risk IGE 70
3.3 Constant Punishment IGE 77
3.4 Context Adaptive Q-IGE 82
3.5 PGPE-IGE and EPHE-IGE 85
3.6 PGPE-IGE - update of the policy distribution 86
3.7 EPHE-IGE - update of the policy distribution 86

4.1 Decoding Objective Adaptive IGE 112
4.2 Time-dependent Q-learning for goal-only-reward OA-MDPs 116
4.3 Prediction Objective Adaptive IGE 120
4.4 Time-dependent Q-learning for general OA-MDPs 126

5.1 Average-adjusted algorithms for continuous average-reward MDPs 138
5.2 Average-adjusted algorithms for episodic average-reward MDPs 141
5.3 Decoding Average-Reward IGE 147
5.4 Value-Quotient Average-Reward IGE 150
5.5 Identification of the reduced set of γ-regions 168

List of Figures

1.1 Examples of a decision-tree and grid-world MDP 8
1.2 Framework of TD Agents 12
1.3 Example of transfer learning in path-finding tasks 17
1.4 The Horde architecture 21
1.5 Framework of the Dependent γ-Ensemble 25

2.1 Framework of the Independent γ-Ensemble 30
2.2 Weighting of future reward dependent on the discount factor 33
2.3 Examples of MDPs leading to different learnable policies by the IGE . . . 34
2.4 Example of a grid-world MDP for which the IGE learns different policies . 36
2.5 Example of Q-functions learned by the IGE in a grid-world MDP 37
2.6 Example for reward boundaries of the IGE for several choices 40
2.7 Results for the decoding of reward trajectories by an inverse γ-matrix . . . 43
2.8 Results for the decoding of choice properties by γ-module pairs 45
2.9 Invariances of the mapping between γ-modules and choices 46
2.10 Boundaries between γ-regions of choices for different choice sets 50
2.11 Effect of discount factors on exploration 51

3.1 Example of an energy foraging task 54
3.2 Differences in CMDP tasks between contexts 55
3.3 Framework of the Context Adaptive IGE 57
3.4 Structure of Interruption Risk MDPs 62
3.5 Interruption Risk MDP used for experiments 65
3.6 Results for Interruption Risk MDPs 66
3.7 Structure of Reward Risk MDPs 68
3.8 Reward Risk MDPs used for experiments 71
3.9 Results for Reward Risk MDPs 72
3.10 Structure of Constant Punishment MDPs 75
3.11 Constant Punishment MDP used for experiments 78
3.12 Results for Constant Punishment MDPs 79
3.13 Energy foraging tasks used to evaluate CMDP algorithms 89
3.14 Learning parameters for the evaluation of CMDP algorithms 91
3.15 Results of the reward per episode for CMDP algorithms for task settings 1 and 2 94
3.16 Results of the reward per episode for CMDP algorithms for task settings 3, 4, and 5 95

3.17 Results of the learning rate for CMDP algorithms under sub-contexts for task settings 1 and 2 . . . 96
3.18 Results of the cumulative reward for CMDP algorithms under sub-contexts for task settings 1 and 2 . . . 97
3.19 Unused observations of CMDP algorithms learning under sub-contexts for task settings 1 and 2 . . . 99

4.1 Framework of the Objective Adaptive IGE . . . 108
4.2 OA-MDP for evaluating objective-adaptive algorithms in deterministic and stochastic, goal-only-reward tasks . . . 114
4.3 Learning parameters for objective-adaptive algorithms in deterministic, goal-only-reward tasks . . . 117
4.4 Results for objective-adaptive algorithms in deterministic, goal-only-reward tasks . . . 118
4.5 Learning parameters for objective-adaptive algorithms in stochastic, goal-only-reward tasks and general tasks . . . 122
4.6 Results for objective-adaptive algorithms in stochastic, goal-only-reward tasks . . . 123
4.7 OA-MDP for evaluating objective-adaptive algorithms in general tasks . . . 125
4.8 Results for objective-adaptive algorithms in general tasks . . . 127

5.1 Reformulation of episodic MDPs as continuous MDPs . . . 139
5.2 Framework of the Average-Reward IGE . . . 142
5.3 Geometric interpretation of the optimality proof for the AR-IGE . . . 146
5.4 Examples of MDPs to evaluate average-reward algorithms . . . 155
5.5 Learning parameters for average-reward algorithms in decision-tree tasks . . . 156
5.6 Learning parameters for average-reward algorithms in grid-world tasks . . . 157
5.7 Results of the total average reward for average-reward algorithms . . . 159
5.8 Results of differences between tasks for average-reward algorithms . . . 161
5.9 Results of the average reward per episode for average-reward algorithms . . . 162
5.10 Example of an MDP with a difference between average-reward per episode and over episodes . . . 164
5.11 γ-regions guaranteed to learn the optimal average-reward choice for different performance bounds . . . 167
5.12 Number of γ-modules needed to guarantee different average-reward bounds 167
5.13 Results of the optimal performance of different performance bounds for the DAR-IGE . . . 169
5.14 Results of the optimal performance of different performance bounds for the VQAR-IGE . . . 170

A.1 Results of the learning rate for CMDP algorithms learning under sub-contexts for task settings 3, 4, and 5 . . . 182
A.2 Results of the learning rate for CMDP algorithms learning under all contexts for task settings 1 and 2 . . . 183
A.3 Results of the learning rate for CMDP algorithms learning under all contexts for task settings 3, 4, and 5 . . . 184

A.4 Results of the cumulative reward for CMDP algorithms learning under sub-contexts for task settings 3, 4, and 5 185
A.5 Results of the cumulative reward for CMDP algorithms learning under all contexts for task settings 1 and 2 186
A.6 Results of the cumulative reward for CMDP algorithms learning under all contexts for task settings 3, 4, and 5 187
A.7 Unused observations of CMDP algorithms learning under sub-contexts for task settings 3, 4, and 5 188

B.1 Learning parameters for episodic average-reward algorithms in decision-tree tasks 189
B.2 Results for episodic average-adjusted algorithms in decision-tree tasks . . . 190
B.3 Learning parameters for continuous average-reward algorithms in grid-world tasks 191
B.4 Results for continuous average-adjusted algorithms in decision-tree tasks . 192

List of Tables

1.1 Classification of reinforcement learning methods 9

3.1 Task settings for the evaluation of CMDP algorithms 90

5.1 Results of the total average reward for average-reward algorithms 158

Introduction

The development of artificial agents that adapt autonomously to new tasks and situations is a major goal in the field of artificial intelligence. Machine learning approaches strive to fulfill this need. They learn the solution to new tasks from observations made on-task, allowing them to interact with a task and to collect data needed to learn a solution. Reinforcement learning is one of the sub-disciplines in machine learning. It studies methods for learning optimal behaviors for problems with multiple decision steps and with possibly delayed outcomes of the behavior, i.e. how the agent performed.

Reinforcement learning tasks consist of several states in which the agent has to decide between several alternative actions. Each action leads the agent to the next state. For example, in a computer game an agent has to find the exit from a maze. The agent's position in the maze is its state, and the directions it can walk are its actions. Depending on the action it chooses, its position will change. For each transition, the agent might receive a reward. Its aim is to maximize the rewards it obtains. Nonetheless, rewards are often delayed, meaning that the current action influences which future states can be visited and which rewards can be gained in the future. The aim of the agent in the maze is to find the exit. Therefore, a positive reward is given to it if it reaches the exit. For all other state transitions, a small negative is given. In such a task, the shortest path to the goal is maximizing the reward sum the agent can gain.

The agent learns such tasks by interacting with the environment. It explores different behaviors and observes their resulting reward. This allows the agent to learn the optimal behavior, which results in the maximum reward for a task. In the maze, the agent has to explore different paths to find the one that leads to the exit where it finally receives a positive reward.

Once the optimal behavior is learned, the agent can apply it again and again to escape the maze. However, our world is ever changing, including the tasks and the objectives that an agent has to fulfill. In the maze game, there might be coins that the agent can collect to get points. The agent gains 1 point for each collected coin, and 100 points for reaching the exit. The task could change so that the agent must not only reach the exit, but must also collect the maximum number of points. Another objective might be that the agent should collect the maximum number of points within a given time limit, or to collect at least a specified number of points before it can exit the maze. Classical reinforcement learning approaches adapt to such changes by relearning the task according to the new objectives. Relearning is often time intensive because the agent has to explore new behaviors.

However, in many situations an agent should quickly adapt to a new task situation. For example, a self-driving car that learned to drive on the right side in Germany should

adapt to driving on the left side if used in England. An autonomous mining robot on a different planet that learned the optimal path between its base station and the mining site should cope with a change in terrain due to a landslide. A financial broker algorithm should deal within seconds or better yet, milliseconds, to changes in the financial world. Situations in which autonomous agents have to quickly adapt are plentiful.

Classical reinforcement learning methods have problems with such quick adaptations. In order to adapt, they have to relearn the entire task, which is very time-consuming. In contrast, humans and animals can perform such adaptations swiftly and seemingly effortlessly. The human brain is evolved to cope with an ever-changing world. Not only can we learn complex tasks, but we also have mechanisms to adapt our behavior on the fly to new situations. Artificial intelligence strives to achieve the same adaptive abilities. This thesis attempts to take a step toward this goal. It presents and analyzes a new framework, the Independent γ-Ensemble, as an approach to improve the adaptive abilities of reinforcement learning approaches. The inspiration for the framework comes from the system it attempts to imitate, the human brain.

The Independent γ-Ensemble

The Independent γ-Ensemble (IGE) is an ensemble of modules, each representing an independent reinforcement learning agent. The modules aim to maximize the expected reward sum in a task. The difference between modules is their discounting of expected future rewards. Future rewards are discounted by weighting them in the reward sum with respect to the time delay before they can be acquired. Some modules have weak discounting. Rewards are weighted similarly, regardless of whether they are acquired sooner or later. Such modules learn long-term strategies that maximize the overall reward sum even if it takes a long time to reach a large reward sum. Other modules have stronger discounting. Proximal rewards are weighted more strongly than distant rewards. Such modules learn behaviors that optimize the short-term reward sum. Their behaviors usually result in quickly obtaining rewards without long delays. In the maze example, modules that optimize long-term rewards learn to collect many coins before locating the exit, whereas, modules that optimize short-term rewards prefer to reach the exit quickly. Finding the exit yields a high reward of 100 points, which these modules prefer to acquire rapidly. For them, the 100 points are strongly discounted if they must be acquired too far in the future. The inspiration for the IGE came from research about human-decision making. Tanaka, Doya, et al. (2004) and Tanaka, Schweighofer, et al. (2007) identified areas in the human brain that seemingly learn behaviors based on discounted future rewards in which each area has a different discounting strength.

As a result of having modules that discount differently, the IGE learns a repertoire of behaviors that can be used to adapt to different situations. The learned behaviors represent different strategies for the trade-off between the amount of reward and the time to acquire it. In the maze game, if a lot of points should be collected, a module that optimizes long-term reward is used, whereas, if the agent should find its way to the exit rapidly, it uses a module that optimizes short-term rewards.

A further property of the IGE is its ability to decode information about the behaviors it has learned. The IGE can decode their expected rewards and the time that is needed to gain them. This is helpful in identifying the most appropriate policy for a given situation.

Classical reinforcement learning approaches do not have this ability. Decoding of the information about its learned behaviors allows the IGE not only to adapt to different situations, but also to optimize the average reward an agent can acquire per time step. This objective is important for tasks that are repeated, such as, for example, if the agent plays the maze game in a loop. In this case, the goal would not be to get the most reward in one game, but to get the maximum reward averaged over the time steps. Classical reinforcement learning methods exist to optimize this objective, but the IGE provides an interesting new way to solve such problems.

Overview of the Thesis

The thesis has the following structure. Chapter 1 introduces the background to the IGE. It gives a short overview of reinforcement learning and introduces Q-learning, which is used as the algorithm to implement the modules of the IGE. Moreover, related methods from the field of transfer learning are reviewed. The aim of transfer learning is to improve the learning performance in new tasks by using knowledge gained from previously solved tasks. This chapter also introduces the Horde architecture that has a similar structure to the IGE, but its objective is to make predictions rather than to learn adaptive behavior. The chapter concludes with a discussion of the connection between reinforcement learning and learning processes in the brain, and with an introduction to the research that inspired the development of the IGE.

The IGE is described in detail in Chapter 2. The chapter analyses properties of the IGE that enable it to learn a set of behaviors and to decode information about its learned policies. It also provides insight into which modules, i.e. which discounting strategies, the agent should use to learn beneficial behaviors.

The following three chapters introduce different application areas of the IGE. Chapter 3 introduces the application of the IGE to tasks that change, depending on some context. For example, for a maze game the context could be a countdown timer. If the timer reaches zero, the agent gets a negative reward or the game ends. Depending on the context, the agent should select the most appropriate behavior that was learned by the ensemble. This chapter introduces some tasks in which such a mapping can be analytically derived. For other tasks it explores approaches to learn the mapping.

Chapter 4 explores the ability of the IGE to adapt to changes in the objectives that an agent has to fulfill in a given environment. In such cases, objectives are formulated in terms of the rewards that the IGE should acquire and the time it needs to achieve them. Because the IGE can decode this information for learned behaviors or learn a prediction about this information, it can immediately select the most appropriate behavior for a given objective.

The final chapter Chapter 5 analyzes the ability of the IGE to optimize average reward and compares it to existing approaches. This chapter proves that the IGE is guaranteed to learn the optimal behavior to optimize average reward in a subclass of tasks: deterministic, goal-only-reward MDPs. The proof allows also to define which modules the IGE needs to guarantee optimality. Furthermore, it shows a way to reduce the number of needed modules, given a lower bound on the final performance.

Chapter 1

Fundamentals and Related Work

This chapter introduces the fundamentals of the IGE, related work, and the inspiration for the framework. First, reinforcement learning will be introduced with a focus on model-free, value-based methods, which comprise the basis for the IGE. Afterward, related approaches from the field of transfer learning are reviewed. These approaches are designed to adapt agents to new tasks similar to the IGE. This is followed by an introduction to the Horde architecture, that uses the same basic framework as the IGE, but for the purpose of prediction instead of adaptation. The final section introduces the original inspiration for the IGE from the field of inter-temporal decision-making in humans.

1.1 Reinforcement Learning

Reinforcement learning is a subdiscipline of machine learning. It studies how solutions to multi-step decision problems can be solved. The aim is to maximize rewards that depend on decisions made by the agent.

Tasks with multiple decision steps are an abundant class of problems that appear in all areas of life, business, engineering, or science. For example, cooking is a multi-step decision problem. If we want to make a pasta sauce, we start with an empty pot. We have to decide which ingredients to add and how to prepare them, such as adding chopped, fried onions. After the onions are added, we have to decide upon the next ingredient and its preparation. The goal of this multi-step process is to maximize the tastiness of the final pasta. Every action that we take during the process has an influence on this outcome. Reinforcement learning provides tools to learn which decisions should be made to solve such tasks.

This chapter introduces fundamental aspects of reinforcement learning that are essential in order to understand the principles behind the IGE. Please refer to the book by Sutton & Barto (1998) for a comprehensive overview and introduction to reinforcement learning. The next section introduces the concept of Markov decision processes (MDPs), that are used to formalize multi-step decision problems. It is followed by an overview over different approaches in reinforcement learning. The final section concentrates on Q-learning, a model-free, value-based approach, that forms the basis of the IGE.

1.1.1 Markov Decision Processes

Reinforcement learning uses the framework of MDP's to formalize decision problems and tasks (Bellman 1957; Puterman 1994).

Definition 1. A Markov Decision Process (MDP) is a tupel:

$$\text{MDP} = (S, A, T, R) \ ,$$

where S is a finite number of states, and A a finite number of actions. T is a transition probability function with $T : S \times A \times S \mapsto [0, 1]$, and R a reward function with $R : S \times A \times S \mapsto \mathbb{R}$.

The states $S = \{s^1, \ldots, s^N\}$ are the possible situations in which an agent has to make a decision. In the cooking example, the state describes the current content of the pot. In the simplest form the states are discrete, i.e. they are represented by scalar natural numbers $S \in \mathbb{N}$. But they can also be continuous ($S \in \mathbb{R}$) or multi-dimensional ($S \in \mathbb{R}^M$). In a state, the agent has to choose between a set of actions $A = \{a^1, \ldots, a^K\}$. In the simplest form, the actions are also described by a set of natural numbers $A \in \mathbb{N}$ that are the same for each state. Nonetheless, the actions can be also continuous ($A \in \mathbb{R}$), or may depend on the state ($f_A : S \mapsto A$).

Depending on the agent's current state s_t at time point t and its action a_t, it transitions to the next state s_{t+1}. The transition function T defines these transitions using a probability distribution over the successor states, given the current state and action: $T(s_t, a_t, s_{t+1}) = \Pr(s_{t+1}|s_t, a_t)$. This allows MDPs to be stochastic, such that actions can lead with different probabilities to any of several successor states. In the cooking example, frying the onions should result in fried onions, but there is the probability of burning the onions so that they cannot be used. In the case of a deterministic MDP, an action has only one possible successor state. In this case, the probability for this transition is 1 and for all other transitions 0. The time of a transition is measured in discrete steps with $t = (0, 1, \ldots, T)$.

For each transition, the agent gains a reward according to the reward function R. The reward is a real-valued scalar and the general goal of the agent is to maximize the reward sum it can achieve in an MDP. The reward function can be defined over different elements of a transition at time point t. The reward function might depend only on the state and action at t: $r_t = R(s_t, a_t)$, or only on the state it transitions into: $r_t = R(s_{t+1})$, but it can also depend on all three components: $r_t = R(s_t, a_t, s_{t+1})$. Rewards may also be probabilistic. In this case, the reward function is the probability distribution over the rewards for a transition: $R(s_t, a_t, s_{t+1}) = \Pr(r_t|s_t, a_t, s_{t+1})$. In the cooking example, a reward is given to the agent according to the taste of the final pasta sauce. Nonetheless, intermediate rewards could be given based on testing of the sauce during its preparation.

The agent starts to interact with an MDP by initialising in a start state s_0 at time point $t = 0$. From there, the agent decides upon an action a_0, transitions to the next state s_1, and receives a reward r_0. The elements of this transition are called the observation of time step t: (s_t, a_t, r_t, s_{t+1}). The observations are the data points forming the means by which the agent learns to maximize the reward. The sequence of observations from a certain time point t until a certain time point T is the trajectory of the agent:

$$\tau_{t:T} = \big(s_t, a_t, r_t, s_{t+1}, a_{t+1}, r_{t+1}, \ldots, s_{T-1}, a_{T-1}, r_{T-1}, s_T\big) \ .$$

One important aspect of MDPs is the Markov property, which means that the probability to enter future states depends only on the present state and not on states visited previously:

$$\Pr(s_{t+1}|s_t, a_t, s_{t-1}, a_{t-1}, s_{t-2}, a_{t-2}, \ldots) = \Pr(s_{t+1}|s_t, a_t)$$

This allows to solve MDPs efficiently, because the optimal decision at the current time point depends only on information regarding the current state. In the cooking example, the state is the ingredients that are in the pot and their condition. We only need to know this much to determine the next step. The condition of the pasta at previous time points and which actions were performed to arrive at the current state are not important to finish the cooking.

Two major types of tasks are distinguished: episodic and continuous MDPs. Episodic MDPs consist of several episodes or runs. Each episode starts in an initial state and ends if a terminal condition is fulfilled. In many cases, a set $S^G \subset S$ of goals or terminal states is defined. An episode ends if the agent enters one of them. It is also possible to define a time limit at which an episode ends. The agent can either begin a new episode in a fixed state that is the same for each episode or according to a probability distribution over states. Continuous MDPs have no terminal condition. In that case, the number of steps goes to infinity. Control problems are often formulated as continuous tasks, for example, in the case of a pendulum swing-up task where an inverted pendulum should the balanced at its top position. Episodic tasks are often used for decision problems, for example, to find the optimal path to a goal state.

Decision-Tree and Grid-World MDPs

Fig. 1.1 shows examples of two MDP types that are used to evaluate the IGE. Both are episodic and have discrete state and action spaces. The first comprises decision-tree MDPs that are represented by directed rooted trees. Vertices are states that are identified by an index: $S = \{s^1, s^2, \ldots\}$. Directed edges represent the actions an agent can choose in a given state and the next state after a transition. The agent starts each episode in the root state s^1. The leaves of the tree are goal states in which the episode ends: $S^G = \{g_1, g_2, g_3\}$. The reward function is either defined over the state-action pairs as shown in Fig. 1.1 (a) or by the state the agent transitions into. Transitions in decision tree MDPs can be deterministic or stochastic. In the deterministic case, the agent transitions to the state as shown by the directed edge in the graph. For stochastic transitions, the probability of each possible successor state is given. The goal of the agent is to make decisions that lead to a high reward sum.

The second MDP type are grid worlds. Their states are arranged in a two-dimensional array. Each patch in Fig. 1.1 (b) represents a state. States are identified by an index with $S = \{s^1, \ldots, s^M\}$. Actions are the four possible movement directions of the agent ($A = \{\text{north}, \text{east}, \text{south}, \text{west}\}$). Depending on the chosen direction, the agent can transition between neighboring states. Black squares represent walls that cannot be entered. If the agent tries to move into a wall its state does not change ($s_t = s_{t+1}$).

An episode starts in a start state and ends if one of several possible goal states are reached. Different reward schemes are possible. The most common is to give positive rewards for reaching a goal state. Other transitions either give no reward or a small negative reward. The goal of the agent is to find the shortest path to a goal state. Grid

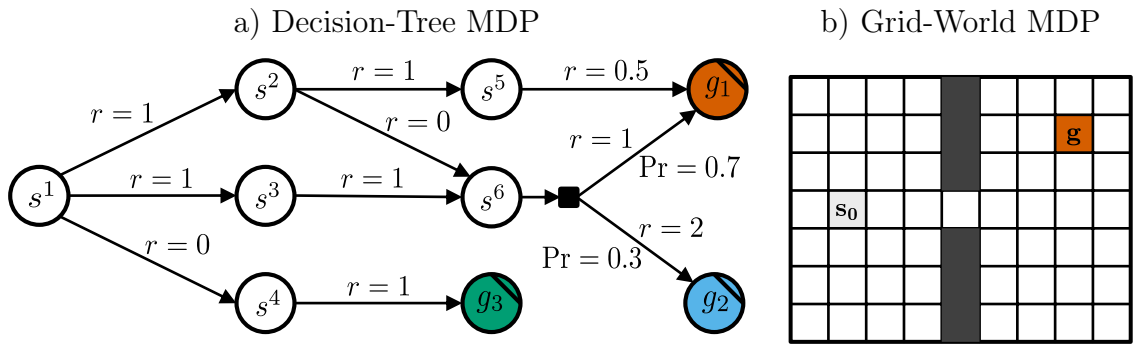


Figure 1.1: Examples for a decision-tree MDP and a grid-world MDP. In decision-tree MDPs, states are represented as vertices and possible actions and transitions as directed edges. The reward for a transition is shown above the edge. The black square for the action in state s^6 represents a stochastic transition that ends with $\Pr = 0.7$ in g_1 and with $\Pr = 0.3$ in g_2. The start state is s^1. Goal states are colored and have a black line in their upper right corners. In grid-world MDPs, states are represented as white patches. The action can reach neighboring patches with four actions ($A = \{\text{north}, \text{east}, \text{south}, \text{west}\}$). The start state is indicated with grey, while goal states are colored. Black squares represent walls that the agent cannot access.

worlds can be deterministic or stochastic. In deterministic cases, actions lead to a state in the desired direction. In the stochastic case, a probability η is given with which the agent moves in a random direction instead.

Policies and Objectives

The behavior of an agent in an MDP is defined by its policy π. A policy can be either a deterministic mapping from a state to an action: $a = \pi(s)$, or a probability distribution over the actions per state: $\pi(a; s) = \Pr(a|s)$. Starting in state s_0, the agent chooses an action according to its policy $a_0 \sim \pi(a; s_0)$, receives a reward r_0, and transitions to the next state s_1. The same procedure is repeated in the next state resulting in a trajectory τ.

The goal or objective of an agent is to find a policy that maximizes the rewards that the agent can gather during its trajectory. Two major types of objectives exist. The first is the maximization of the expected discounted future reward sum:

$$\mathrm{E}\left[\sum_{t=0}^{\infty} \gamma^t r_t\right] = \mathrm{E}[r_0 + \gamma r_1 + \gamma^2 r_2 + \gamma^3 r_3 + \cdots] \,, \tag{1.1}$$

where $\gamma \in [0, 1)$ is a discount factor. The objective is formulated as an expectation because trajectories are stochastic. Trajectories are a random variable because the transition or the reward function can be probabilistic, or the policy itself may be probabilistic. The sum of the expected discounted future reward is mainly used as objective for episodic tasks. The effect of the discount factor γ is to reduce the weight of temporally distant rewards in the sum. The chapter about the IGE discusses this point in more detail (Section 2.2). To use this objective in continuous MDPs the discount factor has to be $\gamma < 1$ so that the reward sum is bounded and does not diverge if t goes to infinity.

The second common objective is the maximization of the expected average reward:

$$\lim_{T\to\infty} \mathrm{E}\left[\frac{1}{T}\sum_{t=0}^{T-1} r_t\right] . \tag{1.2}$$

This objective is mainly used in the case of continuous tasks, because it has a stronger bound on the sum as t goes to infinity. Which objective is used depends on the type of task. Moreover, different algorithms exist for each objective. Although the IGE is primarily an algorithm to optimize the discounted objective, it can also be used to optimize the average reward under specific conditions, as discussed in Chapter 5.

1.1.2 Overview over Reinforcement Learning Algorithms

The following section gives a general overview over reinforcement learning algorithms and their classification. The goal of all algorithms is to learn the policy that maximizes the reward sum in an MDP, as defined by the objective. Algorithms can be categorized according to the information they use to optimize the policy and how they represent the policy (Table 1.1).

One major classification is the distinction between model-free and model-based approaches. Model-free approaches learn the policy based on data from observed trajectories of the agent interacting with the MDP. Model-based methods use a model of the MDP, where the model describes the transition probability and the reward function of the MDP. The model can be either given, or learned from observed trajectories.

Moreover, algorithms are differentiated into value-based and policy-based approaches. Value-based algorithms learn a value function to optimize the expected reward sum. The value for a state represents the expected future reward sum if the agent follows a certain policy starting from that state. This information is used to find the policy that maximizes the value. The earliest approaches of this class were Dynamic Programming algorithms, such as Policy and Value iteration (Bellman 1957). These are model-based approaches in which the model of the MDP is given.

Policy-based algorithms do not learn a value function to optimize the policy. Instead the policy is optimized directly. For this purpose, the policy $\pi(a; s, \theta)$ is usually parameterized by a vector of parameters $\theta \in \mathbb{R}^n$. The parameters determine the probabilities over

	value-based	**policy-based**
model free	TD-Algorithms such as Q-Learning or SARSA (Sutton & Barto 1998)	REINFORCE (Williams 1992) Natural Gradient (Kakade 2001)
model based	Dynamic Programming (Bellman 1957)	PILCO (Deisenroth & Rasmussen 2011)

Table 1.1: Classification of reinforcement learning methods with example algorithms.

the actions for each state. Many methods in this class use a gradient ascent approach by computing the gradient of the policy parameters with respect to the objective (Deisenroth, Neumann, & Peters 2011). Often, the average reward objective is used (Eq. 1.2). The gradient determines how the parameters should be changed to improve the objective. It can be computed for a point in the parameter space by sampling trajectories with policies based on parameters around it. Parameters are then adapted according to the gradient and the process is repeated. The process is described in more detail in Section 3.4.1 that uses such an approach.

The idea behind the IGE, to have modules that are independent reinforcement learners, each with its own discount factor, is very general. It is therefore compatible with any of the introduced classes of algorithms. For the purpose of this thesis, the IGE was implemented and analyzed in terms of a model-free, value-based approach. Model-free approaches can be applied to any problem, even if the model is not available. For a model-based a model has to be given or learned, which was avoided in order to analyze properties of the IGE without the additional complexity of model learning. The IGE uses a value-based instead of a policy-based approach, because the values can be used to decode more information about the learned policies (Section 2.3). This information proved helpful to adapt the IGE to different contexts and to optimize average reward. The basic concepts of model-free, value-based algorithms and Q-learning, which form the basis of the IGE, are introduced in the next section.

1.1.3 Model-Free, Value-Based Algorithms

The central element of value-based algorithms is the value function. It was first introduced to solve MDPs with the model-based dynamic programming approach (Bellman 1957). Nonetheless, it is also the central element of many model-free algorithms. In all these methods, values are associated with states and actions. Values describe predictions about the future outcome of being in these states or taking these actions in terms of the agent's objective. Based on this notion, an agent can decide to go to states and to take actions that maximize the objective. This section concentrates on the objective of maximizing the discounted future reward sum (Eq. 1.1). Value-based methods for the average-reward objective (Eq. 1.2) are discussed in Chapter 5. The next section introduces the concept of the value function and how it is connected to identifying the optimal policy for an MDP. Afterward, temporal difference (TD) algorithms and Q-learning that attempt to learn the value function are introduced.

Value Functions and Policies

The value function can be formulated as a state-value function $V(s)$ or an action-value function $Q(s, a)$, called a Q-function.

Definition 2. The value of a state is the expected discounted reward sum starting at state s and using policy π in successive states:

$$V^{\pi}(s_t) \;=\; E_{\pi}\left[\sum_{k=0}^{\infty} \gamma^k r_{t+k}\right] \;, \tag{1.3}$$

where $\gamma \in [0, 1)$ is the discount factor. The value of a state-action pair, called a Q-value, is the expected discounted reward sum starting from s using action a and following policy π:

$$Q^{\pi}(s_t, a_t) \;=\; E_{\pi}\left[\sum_{k=0}^{\infty} \gamma^k r_{t+k}\right] \;.$$

Bellman showed that the value function can be recursively defined in terms of the immediate reward and the value of the succeeding state:

$$\begin{aligned} V^{\pi}(s_t) &= \mathrm{E}_{\pi}\big[r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} + \dots\big] \\ &= \mathrm{E}_{\pi}\big[r_t + \gamma V^{\pi}(s_{t+1})\big] \\ &= \sum_{s_{t+1} \in S} T(s_t, a_t, s_{t+1}) \Bigg(\sum_{a_t \in A} \pi(a_t; s_t) R(s_t, a_t, s_{t+1}) + \gamma V(s_{t+1}) \Bigg) \;, \end{aligned}$$

where $R(s_t, a_t, s_{t+1})$ is the expected reward over the reward for one step. This so called Bellman Equation can be also formulated for the Q-function:

$$\begin{aligned} Q^{\pi}(s_t, a_t) &= \mathrm{E}_{\pi}\big[r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} + \dots\big] \\ &= \mathrm{E}_{\pi}\big[r_t + \gamma Q^{\pi}(s_{t+1}, \pi(s_{t+1}))\big] \\ &= \sum_{s_{t+1} \in S} T(s_t, a_t, s_{t+1}) \Bigg(R(s_t, a_t, s_{t+1}) + \gamma \sum_{a_{t+1} \in A} \pi(a_{t+1}; s_{t+1}) Q^{\pi}(s_{t+1}, a_{t+1}) \Bigg). \end{aligned}$$

Values depend on a policy and measure the expected discounted future reward that the policy produces. The goal is to find the optimal policy π^* that maximizes the value function for each state:

$$\forall s \in S, \pi : \; V^{\pi^*}(s) \geq V^{\pi}(s) \;.$$

Bellman showed that the optimal value can be reached, if for each state, the action that maximizes the immediate reward and the optimal discounted value of the successor state is chosen:

$$V^*(s_t) = \max_{a_t} \sum_{s_{t+1} \in S} T(s_t, a_t, s_{t+1}) \Big(R(s_t, a_t) + \gamma V^*(s_{t+1}) \Big) \;,$$

$$Q^*(s_t, a_t) = \sum_{s_{t+1} \in S} T(s_t, a_t, s_{t+1}) \Big(R(s_t, a_t, s_{t+1}) + \gamma \max_{a_{t+1}} Q^*(s_{t+1}, a_{t+1}) \Big) \;.$$

If the optimal state-value function is given, the optimal deterministic policy can be deduced by:

$$\pi^*(s_t) = \operatorname*{argmax}_{a_t} \sum_{s_{t+1} \in S} T(s_t, a_t, s_{t+1}) \Big(R(s_t, a_t, s_{t+1}) + \gamma V^*(s_{t+1}) \Big) \;.$$

The expression shows that the transition probability function T, and therefore a model is still necessary to identify the optimal policy. This is not necessary for the optimal Q-function where the optimal policy is given by:

$$\pi^*(s_t) = \operatorname*{argmax}_{a_t} Q^*(s_t, a_t) \;.$$

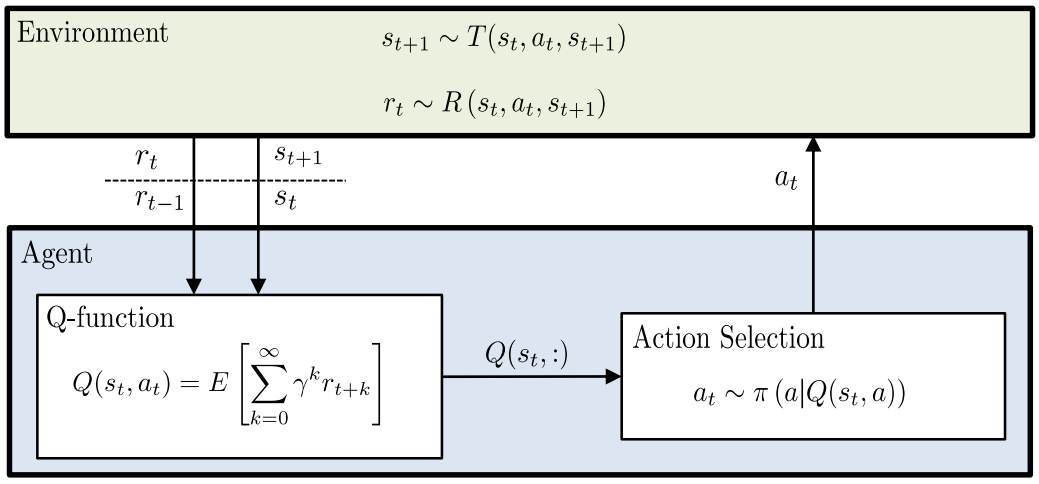


Figure 1.2: The general framework of TD agents. The agent is in state s_t given by the environment. Based on its Q-function is selects its action a_t. The environment informs the agent then about its next state s_{t+1} and what reward r_t it received for its action. This observation is used to update the agents Q-function.

As a consequence, model-free methods generally use Q-functions instead of state-value functions.

In summary, the value function represents the expected future discounted reward sum that an agent will receive if it follows a certain policy. Given the optimal value function the optimal policy for an MDP can be deduced. Value-based methods use this property by learning the optimal value-function to find the optimal policy.

Q-learning

Q-learning is a temporal difference (TD) approach that constitutes the algorithmic basis for the IGE framework (Fig. 1.2). The objective of TD methods (Sutton & Barto 1998) is to estimate the value function for a policy π. These methods make use of the recursive property of the value function, allowing them to perform online learning. In online learning, the prediction of the value-function can be updated after each transition of the agent. Q-learning aims to learn the value function for the optimal policy π^*. The next section introduces the general idea behind TD methods before Q-learning is introduced.

TD learning predicts the Q-function for an agent with policy π. It starts with an initial Q-function that can be arbitrarily initialized. The Q-function is updated after each transition of the agent from state s_t with action a_t to state s_{t+1}. $Q^\pi(s_t, a_t)$ is the agent's prediction of the sum of the expected reward $\mathrm{E}[r_t]$ of this step and the expected discounted sum over all future steps according to policy π: $\mathrm{E}\big[\gamma Q^\pi(s_{t+1}, a_{t+1})\big]$. After the transition the agent observed the reward r_t and the Q-value of the next state. Based on this observation the TD error δ_t is computed:

$$\delta_t = \underbrace{r_t + \gamma Q^\pi(s_{t+1}, a_{t+1})}_{\text{observation}} - \underbrace{Q^\pi(s_t, a_t)}_{\text{prediction}}, \tag{1.4}$$

which is the difference between the prediction and the observation. The next action a_{t+1}

follows the policy of the agent. The difference between the prediction and observation is employed to update the prediction using the Robbins-Monro approach for stochastic approximation (Robbins & Monro 1951):

$$Q^\pi(s_t, a_t) \leftarrow Q^\pi(s_t, a_t) + \alpha\delta_t \ ,$$

where the learning rate $\alpha \in [0, 1]$ defines how strong the update is. If α is not constant, but decays over time in an appropriate way, the algorithm will converge to the true prediction of Q^π.

Q-learning (Watkins 1989; Watkins & Dayan 1992) is based on the TD approach, but instead of learning values for a policy π, its purpose is to learn the values for the optimal policy π^*. It achieves this by updating the prediction after an observation with the Q-value for the action that maximizes the value over the next state:

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha \left(r_t + \gamma \max_{a_{t+1}} Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t) \right) \ . \tag{1.5}$$

Because of the max operator, Q-learning is an off-policy algorithm. It learns the value function of the optimal policy regardless of the policy that is used to produce the observations. The optimality of Q-learning, i.e. that it will converge to the optimal Q-function as the number of time steps goes to infinity, is guaranteed under some assumptions (Watkins & Dayan 1992; Tsitsiklis 1994). First, the environment has to be sufficiently explored, i.e. each state-action pair should be visited an infinite amount of time. Second, the learning rate α must decay to zero according to some conditions.

The Q-function is learned by interacting with the environment and by exploring different strategies, i.e. selecting different actions to explore their outcomes. Using a random policy would guarantee that the agent explores all state-action combinations, but this would take a long time because the exploration would be unfocused. For example, in the grid-world MDP of Fig. 1.1, the agent might once find a path to the goal state, but afterward it might explore again randomly, even though it would make more sense to follow this path and to explore different actions from it that might lead to a shorter path. Furthermore, the performance of the agent would be poor during the learning process with a random exploration. This is problematic in situations where the agent should not only learn, but already gather rewards during the learning.

As a result, the agent has to compensate between exploitation and exploration. Exploitation means that the agent uses the best policy it has learned until that point. This policy is defined by:

$$\pi(s) = \operatorname*{argmax}_a Q(s, a) \ ,$$

and is called the greedy policy. In contrast, during exploration the agent diverges from its greedy policy to try different actions. When to exploit and when to explore is handled by the action selection strategy. The action selection defines a probability function over actions depending upon their Q-values. One of the major approaches is the ϵ-greedy action selection. With probability $1 - \epsilon$, the agent selects the best action and with probability ϵ a random action. The policy is defined by:

$$\Pr(a|s) \ = \ \begin{cases} (1-\epsilon) + \frac{\epsilon}{|A|} & | \text{ if } a = \operatorname{argmax}_u Q(s,u) \\ \frac{\epsilon}{|A|} & | \text{ otherwise .} \end{cases}$$

The agent performs a random exploration with $\epsilon = 1$. With $\epsilon = 0$ the agent uses the greedy policy defined by the value function without any exploration.

Another approach that is often used is the softmax action selection. The softmax action selection prefers actions with higher Q-values:

$$\Pr(a|s) \;=\; \frac{\exp\big(\beta \cdot Q(s,a)\big)}{\sum_u \exp\big(\beta \cdot Q(s,u)\big)} \;,$$

where the strength of the preference can be controlled by the parameter $\beta \in \mathbb{R}^+$. The distribution is uniform for $\beta = 0$, resulting in random exploration. For $\beta \to \infty$, the policy becomes greedy.

Algorithm 1.1 shows the Q-learning algorithm for episodic MDPs with discrete states and actions. The Q-function for such MDPs can be represented by a table that has an entry that can be looked up for each state-action pair. If the state space is continuous, a function approximator such as a neural network would be needed to represent the Q-function. Because of the episodic structure of the MDP, it is also observed if a transition ends in a terminal ($isTerminal = true$). In this case, the final state has no successor. Therefore, the update of the last transition ignores the Q-value of the last state. Algorithms of all later chapters have the same structure, but the differentiation between terminal and non-terminal states is excluded so as to simplify the code of the algorithms.

In summary, Q-learning is a model-free, value-based method that is based on the TD approach. It learns the optimal Q-function, which estimates the expected future discounted reward sum for the optimal policy. Based on the Q-values, the agent can choose the action that maximizes the reward sum. This allows an agent to learn the optimal policy for multi-step decision problems.

Algorithm 1.1: Q-learning for discrete, episodic MDPs

Input:
Learning rate: $\alpha \in [0, 1]$
Discount factor: $\gamma \in [0, 1)$

initialize $Q(s, a)$ arbitrarily
repeat (for each episode)
  initialize state s
  repeat (for each step in episode)
    $a \leftarrow$ choose an action for s derived from $Q_{\tilde{\gamma}}(s, a)$ (e.g. ϵ-greedy)
    r, $s', isTerminal \leftarrow$ take action a, observe outcome
    if $isTerminal = false$ **then**
      $Q(s,a) \leftarrow Q(s,a) + \alpha\big(r + \gamma \max_{a'} Q(s',a') - Q(s,a)\big)$
    else
      $Q(s,a) \leftarrow Q(s,a) + \alpha\,(r - Q(s,a))$
    end
    $s \leftarrow s'$
  until s *is terminal-state*
until *termination*

1.2 Transfer Learning

In classical reinforcement learning, the agent learns the optimal policy to solve a specific MDP for a specific objective. The advantage of the IGE compared to classical methods is its adaptive capacity, in cases in which the MDP or the objective changes. It learns a set of behaviors for an environment that allows it to switch between them, if there are changes in the task. The field of transfer learning comprises similar approaches.

Transfer learning methods transfer knowledge from source tasks to target tasks (Taylor & Stone 2009; Lazaric 2012). The goal is to improve learning performance on the target task. This enhancement can improve the learning speed, resulting in a higher initial performance, or a higher final performance after learning. The type of transferred knowledge can be of different forms. For example, sampled data from source tasks, such as trajectories, can be used (Lazaric, Restelli, & Bonarini 2008). Another possibility is the representations used to describe a task or features thereof (Ferguson & Mahadevan 2006). It is also possible to transfer values, partial policies, or complete policies between tasks.

Two major settings for transfer methods can be distinguished. In the first setting, knowledge is transferred between tasks that have the same state and action space ($S \times A$). In the second setting, the source and target tasks have different state and action spaces. For the second setting, a mapping between the state and action spaces of the task is usually necessary and must be either given or learned. Moreover, for both settings the existence of one or multiple source tasks can be distinguished. In the case of several sources, not only the transfer itself has to be regarded, but also the choice of the sources from which knowledge should be transferred.

The IGE is part of the first setting, in which the source and target task share a common state and action space. Policies of the IGE can be continuously learned over several source tasks in a lifelong learning approach. The next section reviews approaches that also transfer knowledge between tasks with a common state and action space. The review concentrates on approaches that share policies or partial policies between tasks, which is similar to the strategy of the IGE.

1.2.1 Transfer of Policies

Several approaches exist that transfer full policies from source to target tasks. One of the simplest methods is to use the learned policy from a target task as an initial policy in the source task. However, one problem is that this simple method often fails if the reward or transition function between both tasks is not similar enough. A theoretical analysis of the bound on the performance of a transferred policy can be found in (Phillips 2006), which is based on metrics introduced by Ferns, Panangaden, & Precup (2004).

The approach by Bernstein (1999) constructs an average policy over all optimal policies of several source tasks. The agent can use this average policy similarly to an action by using it for a specified number of steps.

Other approaches build a policy library and use a specific policy from a source task, instead of an average over them. The library by Mehta et al. (2008) uses a subset of the optimal policies from source tasks. Given a new target task one of the policies is selected as the initial policy for the task. The tasks differ only in their reward functions, which

are based on feature vectors. Instead of a reward that is given per transition, a feature vector is given. This feature-vector function is the same for all tasks. Each task has a reward-weight vector holding a weight for each feature. The final reward is the sum over the weighted features. This approach is value-based and the value function learns the expected future reward feature vector for the optimal policy per task. Given a new task, the performance of policies in the library can be evaluated by weighting their value functions based on the task's reward weights. This allows to identify the policy with the best performance and to use it as the initial policy. After the policy is adapted to the target tasks, it is added to the library if its performance differs strongly enough from the other policies.

The approach by Fernández & Veloso (2006) and Fernández, García, & Veloso (2010) also builds a policy library from previously solved tasks. For each episode, one of the polices from the library is used to bias exploration of the agent in the new task, using its actions with some probability at each step. The policy that is used is stochastically selected at the beginning of each episode. The selection is based upon performance in the new task, computed by an off-policy approach. Also in this approach, the new policy from the new task is added to the policy library if its performance is sufficiently better than those of existing policies.

Mahmud et al. (2013) use a clustering approach to select source tasks from policies. Tasks are clustered according to how well their optimal policies perform in one another. Tasks for which optimal policies perform similarly in one another are clustered together. For each cluster, the policy that performed best is then used for the set of policies that guides the search in a new task.

The IGE learns a set of policies, as in most of the approaches that transfer complete policies. The difference between the IGE and existing approaches is that the existing approaches learn a set of policies that represent optimal solutions for previously encountered tasks. In contrast, the IGE learns different policies for the same environment, but which represent different solutions for a trade-off between reward amount and time to reach the reward. The agent can then select the most appropriate solution, depending on changes in the environment or its objective. The IGE learns a set of policies that may be helpful in future tasks, rather than a set of policies that have been successful in past tasks.

1.2.2 Transfer of Skills and Options

Another approach to transfer knowledge is to de-construct tasks into sub-tasks, and to transfer the solutions for such sub-tasks. Consider, for example, a grid world with two rooms (A and B) that are separated by a wall in which there is a doorway (Fig. 1.3). The goal is to find a path from a start state in room A to a goal state in room B. Tasks differ in the location of the goal state in room B. The task can be de-constructed into sub-tasks. The first sub-task is to reach the doorway from the start state to enter room B. The second is to reach the goal-state in room B. The sub-task of reaching room B is common to all tasks in this environment and can be transferred between them to increase learning speed.

The architecture by Singh (1992) used a separate module for each possible sub-task. As with the IGE, each module is a Q-learning agent learning its own value function. Each module has an individual reward function that defines the sub-goal, for example to reach

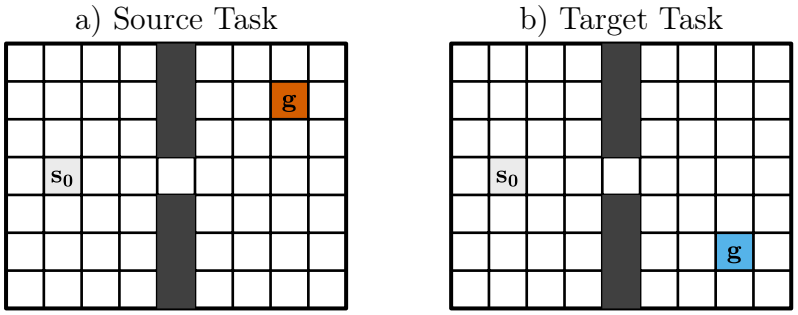


Figure 1.3: Path-finding problem with two tasks. The goal is to find the shortest path from the start state s_0 to the goal state. Both tasks have the same environment consisting of two rooms that are connected by a doorway. The agent starts in room A and has to find the shortest path to the goal state in room B. The position of the goal state differs between the source and target task. Nonetheless, the agent has to find in both tasks first the optimal path to enter room B through the doorway which could be used as a skill that can be transferred from the source to the target task.

the doorway to room B. Because the sub-modules use off-policy Q-learning, they can learn their value functions in parallel from the same observations. The agent learns then how to sequentially combine the sub-tasks to solve the overall task.

The concept of solving a task by decomposing it into sub-tasks was further developed in the options framework (Sutton, Precup, & Singh 1999). Options $o \in \mathcal{O}$ are temporally abstract actions representing a chain of actions that are combined to create a macro-action or a skill. Each option is defined by a triplet with:

$$(\mathcal{I}_o, \pi_o, \beta_o) \ ,$$

where $\mathcal{I}_o \subseteq S$ is a set of initial states in which the option can be started. $\pi_o = \Pr(a|s)$ is the intra-option policy that defines the actions of the agent while the option is active. $\beta_o : S \mapsto [0,\ 1]$ is a termination function that defines when an option finishes, based on the state of the agent or the duration the option was used.

Given a set of options, the agent can learn a policy $\pi(o|s)$ over options. In state s, the agent selects one of the possible options, i.e. an option with $s \in \mathcal{I}_o$. From this point, the policy π_o of the selected option is used until its termination condition $\beta_o(s_{t+k})$ is fulfilled. The problem of choosing the optimal option in a state is formalized by a semi-MDP (SMDP) in which actions can take different amounts of time (Puterman 1994). An optimal state-observation value function can be learned with a similar approach as Q-learning:

$$Q(s_t, o) \leftarrow Q(s_t, o_t) + \alpha \left(R_k + \gamma^k \max_{o_{t+k}} Q(s_{t+k}, o_{t+k}) - Q(s_t, o_t) \right) \ ,$$

where k is the number of steps that elapsed between s_t and its successor after option o_t terminated. $R_k = \sum_{i=0}^{k-1} \gamma^k r_{t+i}$ is the discounted reward sum during this time.

Options learned in source tasks can then be reused in target tasks (Perkins & Precup 1999). This is similar to the IGE, which also learns a set of skills that can be reused after a task change. Several approaches have been proposed to identify and learn useful

options for transfer learning. The following review distinguishes approaches based on their methods to discover options.

Sub-Goal Options

One of the most studied approaches for option discovery is to identify sub-goals that may be important, i.e. states or state regions in an environment. Sub-goals represent states that should be reached to solve the overall problem, such as a doorway between two rooms. For each sub-goal, an option is created with a policy that leads the agent to the associated states. The policy can be learned by having for each option a Q-function, similar to the modular architecture of the IGE. The values of an option are learned based on a pseudo-reward function that gives positive rewards if the sub-goal's states are reached.

Different approaches exist with respect to the questions of what constitutes a good sub-goal and how it can be identified. One of the salient concepts is the notion of bottleneck states that an agent tends to visit frequently on successful trajectories. In the room example, this would be the doorway between the two rooms that must be transited in any successful trajectory. One simple identification criterion is to measure the visiting frequency of states during the learning in a source task and to select states with a high frequency as sub-goals. The frequency can either be collected for a single task (Digney 1998) or determined over several subtasks (Stolle & Precup 2002). McGovern & Barto (2001) use the concept of diverse density to identify states that are frequently visited in successful trajectories, i.e. trajectories that reach the overall goal state of the source task, but not in unsuccessful ones. Asadi & Huber (2007) introduce a criterion based on a relative local increase in the visitation frequency for a state to detect states that are local attractors for trajectories.

Further approaches use methods from graph theory to identify bottlenecks and other sub-goals. They are based on the state transition graph. The graph has a vertex for each state and the edges represents possible transitions between them. The edges are usually weighted based on the observed transition frequency. Graph partitioning algorithms are then used to identify bottleneck states that have to be traversed to reach separate regions in the graph, such as a doorway. The approach by Menache, Mannor, Shimkin, et al. (2002) uses the Max-Flow/Min-Cut algorithm and operates on the whole state space. The L-Cut approach by Şimşek, Wolfe, & Barto (2005) operates on a local level, using graphs that include only recent state transition information. Şimşek & Barto (2009) use the concept of betweenness to measure the centrality of a certain state, i.e. how important it is so that different state regions can reach each other by the shortest paths. Clustering approaches separate the state space into regions that represent stronger spacially connected regions, such as the individual rooms in the room example (Mannor et al. 2004; Lakshminarayanan et al. 2016). Options are then learned to reach neighboring state regions from a region. The approach by Machado, Bellemare, & Bowling (2017) is based on the concept of Proto-Value-Functions (Mahadevan 2005; Mahadevan & Maggioni 2007). It identifies sub-goals based on the Laplacian matrix of a state transition graph that can be used to describe diffusion models in graphs.

Concepts from the developmental psychology of humans have also been utilized to to identify sub-goal states. Şimşek & Barto (2004) use the concept of relative novelty, in which a state becomes the sub-goal, if it leads after being visited to states that are

novel, i.e. that have not been visited often. The approach of Bonarini, Lazaric, & Restelli (2006) identifies states that are difficult to reach during random exploration, but that can reach other states easily. A different notion is the concept of saliency, in which states that produce significant changes, such as a strong change in a light level when a light switch is triggered, are used as sub-goals (Barto, Singh, & Chentanez 2004; Singh, Barto, & Chentanez 2005). Digney (1998) used a similar method by creating sub-goals for states that produced unexpected reward.

Other approaches identify sub-goals by analyzing the structure between state dimensions in multi-dimensional MDPs. For example, in the room example, the state space could comprise the local position in a room, and a different dimension could encode the room number. State variables have a certain structure that describes how they depend on each other, for example, being at the doorway in a room can change the room number in the next step. The HexQ approach (Hengst 2002; Hengst 2003) identifies states in which a lower dimension, for example the position in a room, changes an upper dimension, such as the room number. Such access states to other higher states become sub-goals. Which dimensions are upper or lower are determined by how often they change during a random exploration. In the room example, the position in a room changes more than the room number. VISA (Jonsson & Barto 2006) extends this approach by constructing a causal Bayesian network that represents how state variables influence each other to detect access states.

Sub-goals are also discovered by analyzing and de-constructing successful trajectories (Konidaris & Barto 2009; Konidaris, Kuindersma, et al. 2010). Sub-goals are created starting from the final goal state in a task. An option is created that reaches this state from predecessor states that are used as the initiation set of the option. Then the initiation states of this option become the sub-goal for the next option. This process is repeated until options exists that cover the trajectory from the start to the goal state.

Reusage of Sub-Policies

Another class of algorithms identifies sub-policies, i.e. policies in sub-regions of the state space, which can be reused for other tasks. The approach by Thrun & Schwartz (1995) predates the options framework. It identifies policies for sub-regions that provide a high performance over several tasks. Because the performance is optimal if individual sub-policies are created for every task, this approach introduces a description length measure that should be minimized. The description length quantifies the amount of memory needed to encode the policy of the skills, i.e. how many states the policy covers. Having individual skills for each task requires a lot of memory, whereas if skills are useful for several tasks, these can be reused to decrease the required memory.

PolicyBlocks (Pickett & Barto 2002) is a similar method for the options framework. Hawasly & Ramamoorthy (2013) identified continuous sub-regions in the state space that had similar policies for successful trajectories in several tasks with help of a Gaussian mixture model. Policies in these regions are then used as options. The approach by Brunskill & Li (2014) identifies options from policies of previous tasks based on a measure of sample efficiency, i.e. how many update steps the learning algorithm needs to find good solutions with a certain performance bound.

Parameterized Options and Composition Methods

Other approaches are more general and do not employ specific metrics that define what constitutes a good sub-goal. Instead these approaches separate a task into a given number of options so that the task can best be solved by them. The properties of an option, i.e. its initiation set, policy, and termination condition, are learned based on its performance in solving a task. The approach by Daniel et al. (2016) uses a probabilistic option framework that represents all option components as parameterized distributions. Given the desired number of options, the parameters of the distributions are optimized with an expectation maximization approach. Bacon, Harb, & Precup (2017) use also options with parameterized policies and termination conditions. A policy gradient approach is used to learn parameters of a given number of options so that the given task can be solved. Mankowitz, Mann, & Mannor (2016) use a similar approach to learn options that are responsible for separate regions in the state space.

Options can also be created by combining existing options or basic actions together. The following methods achieve this by using option models that define the outcome of an option in terms of expected reward and the expected state. Based on this information, the approaches can sequentially combine options in the best way to solve a task. This allows the identification of new high-level options as a combination of several low-level options. As for previous methods, options are created that help to solve certain tasks. Silver & Ciosek (2012) introduce such an approach for discrete state spaces. Sorg & Singh (2010) use a linear representation of options and models based on state features that can operate in continuous states spaces. The approach of Yao et al. (2014) learns reward-independent option-models using linear function approximation. This allows the model to evaluate options under different reward models, which could differ between tasks.

Conclusion

Options represent reusable skills that can be combined to solve tasks and that can be shared between tasks. The IGE shares features with many of the reviewed approaches. Options are usually represented as modules that have their own policies, like the modules of the IGE. Many approaches that identify sub-goals also hold a separate value functions that learn to reach its sub-goal. In contrast, the IGE does not de-construct a task into sequential sub-tasks. Instead the policies learned by different modules represent different solutions to the whole task in terms of trade-offs for reward amount and timing. Moreover, most existing approaches discover options based on the structure of the state space, for instance, by identifying bottlenecks, or they analyze which parts of previous optimal policies could be reused. The IGE discovers reuseable policies based on different discountings of reward, something that has not been explored in the existing literature.

1.3 The Horde Architecture

Although the idea of a modular framework with different discounting factors has not been explored for transfer learning, it has already been introduced by the Horde architecture (Sutton, Modayil, et al. 2011). The goal of the architecture is to allow a robot to perform nexting, which is a psychological concept describing the ability of humans and animals

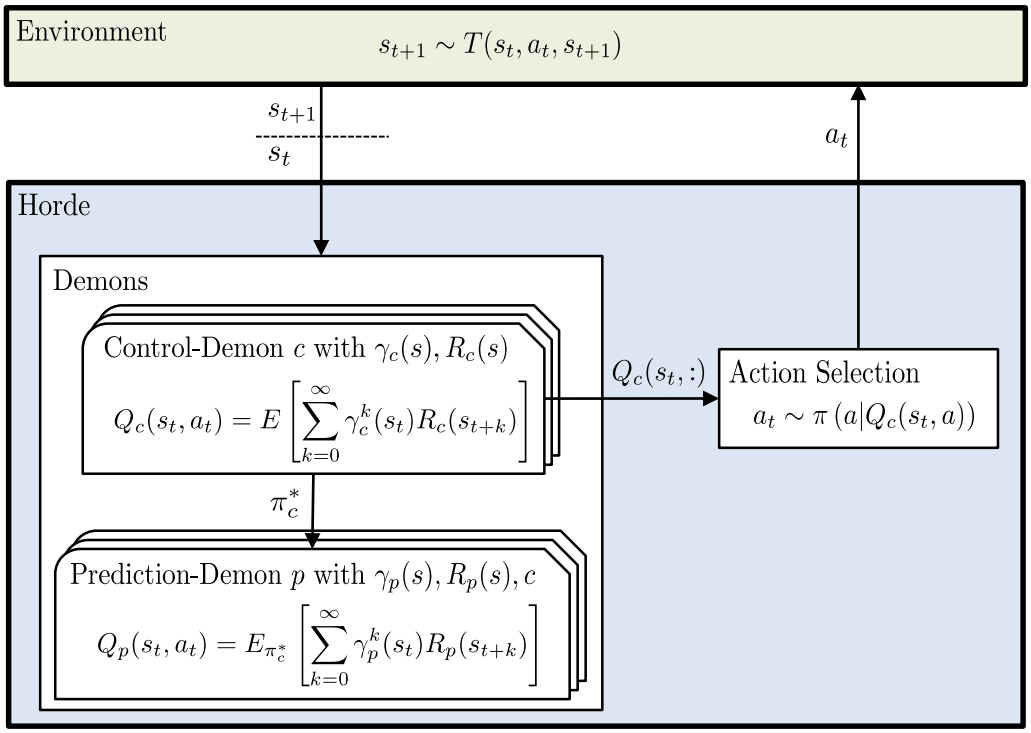


Figure 1.4: The Horde architecture consists of several control and prediction demons. Each demon has its own state-dependent discount-factor function $\gamma(s) \in [0, 1]$ and pseudo-reward function $R(s) \in \mathbb{R}$. Each prediction demon learns its Q-values for the policy of a certain control demon c, for example, to predict in how many steps the agent would bump into a wall using this policy.

to continually predict what will happen next in the near future. Horde consists of many independent modules called demons (Fig. 1.4). Each demon is an approximate value function with its own policy and pseudo-reward function. They can learn in parallel to predict different sensory events for a robot that interacts with its environment. Such events include, for example, whether the robot will bump into a wall, or the change of the light intensity around it. In extensions of the architecture, demons also have different discount factors to predict such sensory events on different time scales (White, Modayil, & Sutton 2012; Modayil, White, & Sutton 2014).

The Horde architecture distinguishes between control and prediction demons. Control demons learn policies that the robot should follow, for example, to seek some light source. Prediction demons predict outcomes of the policies that their associated control demons learn, for various sensory channels and events. Prediction demons have pseudo-reward functions that encode the measurement that should be predicted. For example, reward could be coupled to light intensity measured by a light sensor. The value function of the prediction demon would then learn the expected future discounted sum over this measurement, i.e. how much light can be expected if a certain policy is followed. The prediction is performed over different time scales by having demons with different discount factors. Smaller γ's would only predict the light intensity in the near future, whereas larger γ's would predict the light intensity further into the future.

Additionally, prediction demons can also have state-dependent discount factors $\gamma(s)$. These can be used to detect events and how far away they are. For example, the discount factor could be set to $\gamma = 1$ as long as the robot does not touch a wall, but after bumping into one it is set to $\gamma = 0$. If the pseudo-reward function gives a reward of 1 for every step, then the demon predicts in how many steps the robot will bump into a wall.

The main contribution of the work has been to show that a robot can learn thousands of such demons in real time with little computational effort. Value functions were represented by linear function approximators based on tile codings. As learning algorithm was a TD(λ) approach for function approximators used (Maei & Sutton 2010).

Similar to Horde, the IGE also has a modular architecture with value functions based on different discount parameters. The Horde architecture expands the IGE in some respects, such as having state-dependent discount factors. Nonetheless, the goal of the IGE is different from that of Horde. The goal of Horde is to predict future outcomes, not to control a robot or to select meaningful behavior. In contrast, the IGE was developed and analyzed with the intent of optimizing a reward function and learn behaviors that help the agent to adapt to different situations.

1.4 Inspiration from Neuroscience

Although the IGE uses an idea similar to the Horde architecture, its original inspiration comes from a hypothesis about decision-making and learning processes in the human brain. The study of decision-making and learning in humans and animals is closely linked with the field of reinforcement learning. Theories and ideas from both fields inspired each other (Sutton & Barto 1998; Niv 2009). The psychological theories of Pavlovian conditioning (Pavlov 1927) and instrumental conditioning (Thorndike 1911; Skinner 1935) formulated how animals predict events, including rewarding stimuli and how they select actions to maximize reward. Reinforcement learning provides an engineering approach to the same questions by asking how long-term future reward can be predicted and optimized. Connections between both fields span several areas.

For example, the Rescorla-Wagner model (Rescorla, Wagner, et al. 1972) and its extension, the temporal difference model (Sutton & Barto 1990) describe formally how animals and humans could predict stimuli or reward from preceding stimuli as described by Pavlovian conditioning. The models are rooted in the concept of the TD error (Eq. 1.4) and the value function as used in model-free reinforcement learning. Model-based reinforcement learning mechanisms are connected to the concept of cognitive maps (Tolman 1948). Cognitive maps are learned representations of the environment used to plan the best behavior to achieve a certain goal. The distinction between model-free and model-based processes is also used to differentiate between habitual and goal-oriented behaviors (Daw, Niv, & Dayan 2005). Habitual behavior is automatically triggered by specific stimuli. It is explained by a value function that provides a value that was learned over many trials for each action. Being in a certain state, animals are simply using the action with the highest value; thus, the action is triggered by the current state in a habitual sense. Goal-oriented behavior is more flexible and depends on the goal that an animal wants to achieve. Model-based processes have this property, using their environmental model to identify the best strategy to reach a goal.

Reinforcement learning mechanisms are not only linked to psychological models of decision-making and learning. In certain cases, they also inspire theories about brain processes and functions connected to decision-making (Doya 1999). The next section introduces evidence for model-free processes in the brain, which comprise a class of algorithms to which the IGE belongs. This is followed by a description of the brain research that inspired the IGE and an existing model in the field of human decision-making that is similar to it.

1.4.1 Model-Free Reinforcement Learning in the Brain

Model-free, value-based reinforcement learning mechanisms have been linked to brain processes involved in learning behavior. Neural correlates have been identified for two components of these processes: TD error computations and the existence of value functions.

The evidence for TD error computations in the brain comes from findings that activity of dopamine neurons in the substantia nigra (SN) and the ventral tegmental area (VTA) follow TD error characteristics (Schultz, Dayan, & Montague 1997). These neurons increase activity in response to unexpected rewards and show a reduction in activity at time points when rewards are expected, but are not received. Furthermore, fMRI studies show that activity in the projection areas of these neurons in the striatum and the frontal cortex is also correlated with reward prediction error computations (Pagnoni et al. 2002; Pessiglione et al. 2006). These neurons release the neurotransmitter dopamine when activated. Dopamine is therefore regarded as a TD error-learning signal that reinforces actions or states experienced prior to its release. Experiments that stimulate dopamine neurons using intracranial methods such as optogenetics, support this hypothesis. Rats prefer a lever that stimulates dopamine neurons over one that does not (Adamantidis et al. 2011) and they prefer locations where dopamine responses are elicited (Tsai et al. 2009). Moreover, human fMRI studies show that reward prediction error-related activity can be measured in participants that learn a task in contrast to non-learning participants (Schönberg et al. 2007). That dopamine is important for learning is also supported by findings that it modulates plasticity in the striatum (Reynolds & Wickens 2002). In summary, these findings suggest that dopamine neurons in the SN and VTA encode the TD error, which is projected into the striatum and the prefrontal cortex, via the neurotransmitter dopamine.

Another component of model-free, value-based algorithms is their use of a value function. Correlations between the firing pattern of striatal neurons and values of TD models have been identified in electrophysiological experiments in monkeys and rats (Samejima et al. 2005; Ito & Doya 2009). Animals performed decision-learning tasks and their behavioral choices were modeled with TD algorithms to compute the value function that animals seem to use. Electrical recordings of neurons showed a correlation between their activity and values computed by TD algorithms, providing evidence that animals may encode value functions in the striatum. This is further supported by human fMRI experiments, which also found a correlation of activity in the striatum and the prefrontal cortex with values based on TD computations (Kable & Glimcher 2007; Gläscher et al. 2010). In conclusion, the evidence for a neural representation of values and TD error supports the theory that the brain uses model-free, value-based mechanisms similar to TD methods.

1.4.2 Modular Discounting Structure in the Brain

Although the hypothesis for model-free, value-based brain processes has a strong experimental foundation, many questions remain. These include how parameters such as the learning rate α, or the temporal discount factor γ are controlled and what their neurobiological substrates are (Doya 2002). Tanaka, Doya, et al. (2004), Tanaka, Schweighofer, et al. (2007), and Schweighofer et al. (2008) investigated these questions in regard to the discount factor. Results of their experiments suggest a modular structure in the striatum, where each module learns values based on a different discount factor. Which modules influence behavior seems to be controlled by the neurotransmitter serotonin.

In their fMRI experiments participants had to choose between long-delay, high-reward versus short-delay, low-reward choices (Tanaka, Schweighofer, et al. 2007). Activity in the striatum was correlated with values computed for each choice and for different discount factors. The temporal discount factor γ defines how strongly the value of a choice gets discounted if it is delayed. A small γ setting prefers short-delay, low-reward choices, whereas a high γ prefers choices with a long-delays, but higher rewards. The experiment demonstrated that the dorsal part of the striatum correlates to values learned with high γ's, whereas, more ventral parts correlate with small γ's.

The experiments also showed that the serotonin level of participants influenced which of the areas are stronger correlated during decision-making. In participants with low serotonin levels, induced by a low tryptophan diet, a correlation between brain activity and values could only be found for ventral areas, i.e. the areas connected to low discount factors. Whereas, in participants with high serotonin levels, a correlation existed only for dorsal areas linked to large discount factors. Schweighofer et al. (2008) showed that serotonin level not only has an effect on brain activity, but also on the choice behavior of participants. Those with low serotonin levels preferred short-term choices, showing that they discounted future rewards more strongly, whereas, participants with higher serotonin levels preferred long-term choices.

The foregoing research suggests the potential existence of sub-modules in the brain for different γ settings, where each module learns its own value function. The IGE is based on this hypothesis. It consists of several modules each learning a value function with a different discount parameter. Similar to the effect of serotonin, one of the modules can be selected to use its policy to define the behavior of the agent.

1.4.3 The Dependent γ-Ensemble

Kurth-Nelson & Redish (2009) proposed a cognitive model based on the finding about the potential modular structure in the striatum. The model accounts for the discounting behavior observed in humans in temporal decision-making tasks. In such tasks, several alternatives are offered, among which a participant must choose, similar to the experiments by Tanaka, Schweighofer, et al. (2007). Each choice $c = (r, t)$ gives a certain reward r, such as money that is given after a certain amount of time t. Human choice behavior can be modeled using a discounted value approach (Frederick, Loewenstein, & O'Donoghue 2002). A value is computed for each choice based on a discount factor $d(t)$ that involves the time:

$$V(c) = d(t) \cdot r \ ,$$

and the choice with the highest value is selected. This is analogous to the value function in reinforcement learning that also discounts future expected reward. The correct form of the discount factor is sill an open question. One of the popular models is hyperbolic discounting:

$$d(t)_{\text{hyperbolic}} = \frac{1}{\kappa t + 1} \ , \tag{1.6}$$

where $\kappa \in \mathbb{R}^+$ controls the strength of discounting. In contrast, the discounting used in reinforcement learning, as used for the value function (Eq. 1.3), is exponential:

$$d(t)_{\text{exponential}} = \gamma^t \ ,$$

with $\gamma \in \mathbb{R}^+$ as the temporal discount factor. The hyperbolic discounting model fits often better the choice behavior observed in humans compared to exponential discounting (Frederick, Loewenstein, & O'Donoghue 2002). This observation stands in contrast to the hypothesis that the brain uses model-free, value-based reinforcement learning processes which are based on exponential discounting for such decision making processes.

The multi-micro-agent TD model of Kurth-Nelson & Redish (2009) provides a possible solution of this problem. It uses an ensemble of exponentially discounting reinforcement learning modules to model hyperbolic discounting. In the context of this thesis, the model is called the Dependent γ-Ensemble (DGE). The DGE's capacity for hyperbolic discounting is only part of the multi-micro-agent TD model that also has a state-belief approach in which each module has its own belief about its current state.

The DGE is composed of several modules (Fig. 1.5). Each module learns a state-action value function with an individual discount factor γ:

$$Q_\gamma(s_t, a_t) = \mathrm{E}\left[r_t + \gamma Q_\gamma(s_{t+1}, \underset{a_{t+1}}{\operatorname{argmax}}\, Q_E(s_{t+1}, a_{t+1}))\right] \ .$$

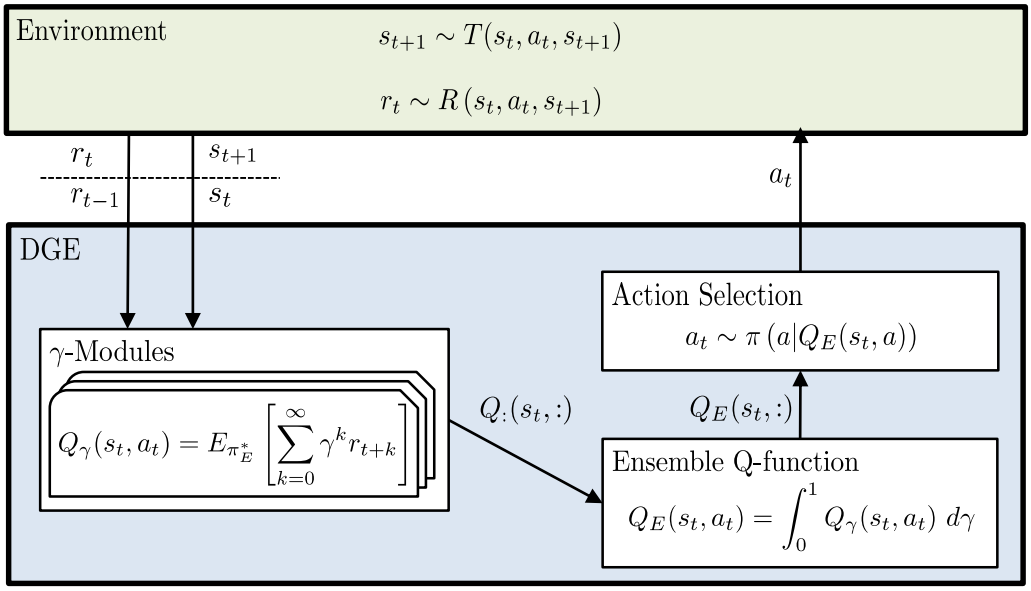


Figure 1.5: The Dependent γ-Ensemble framework.

The Q-function is similar to the definition of Q-Learning (Eq. 1.5). In contrast to Q-learning, it does not use the Q-value of the next state's (s_{t+1}) action that maximize its own value Q_γ. Instead, it uses the action that optimizes the Q-value defined over the ensemble of Q-functions Q_E. The ensemble's Q-value is defined as the integral over the Q-values of all modules:

$$Q_E(s,a) = \int_0^1 Q_\gamma(s,a)\ d\gamma\ . \tag{1.7}$$

Solving the integral yields the definition of the expected future reward sum maximized by the DGE:

$$\begin{aligned} Q_E(s,a) &= \textstyle\int_0^1 Q_\gamma(s,a)\ d\gamma \\ &= \mathrm{E}\left[\textstyle\int_0^1 \left(r_0 + \gamma^1 r_1 + \gamma^2 r_2 + \cdots + \gamma^n r_n\right) d\gamma\right] \\ &= \mathrm{E}\left[r_0 + \tfrac{1}{2} r_1 + \tfrac{1}{3} r_2 + \cdots + \tfrac{1}{n+1} r_n\right]\ . \end{aligned}$$

This shows that the discount factor for each reward is:

$$d(t)_{\mathrm{DGE}} = \frac{1}{t+1}\ ,$$

which is equal to the hyperbolic discounting factor with $\kappa = 1$ (Eq. 1.6).

Kurth-Nelson & Redish (2009) showed further that different discounting strategies are possible by introducing a weight $\omega(\gamma) \in \mathbb{R}$ for the Q-value of each module in the integral over the ensemble:

$$Q_E(s,a) = \int_0^1 \omega(\gamma)\ Q_\gamma(s,a)\ d\gamma\ .$$

Extending this ideas shows that true hyperbolic discounting is possible with the weight:

$$\omega(\gamma) = \frac{1}{\kappa}\gamma^{(\frac{1}{\kappa}-1)}$$

where $\kappa \in \mathbb{R}^+$ is a free parameter and not restricted to $\kappa = 1$ as for Eq. 1.7. Using this weight results in an ensemble Q-value that replicates hyperbolic discounting:

$$\begin{aligned} Q_E(s,a) &= \textstyle\int_0^1 \frac{1}{\kappa}\gamma^{(\frac{1}{\kappa}-1)} Q_\gamma(s,a)\ d\gamma \\ &= \mathrm{E}\left[\textstyle\int_0^1 \frac{1}{\kappa}\gamma^{(\frac{1}{\kappa}-1)} \left(r_0 + \gamma^1 r_1 + \gamma^2 r_2 + \cdots + \gamma^n r_n\right) d\gamma\right] \\ &= \mathrm{E}\left[r_0 + \tfrac{1}{\kappa+1} r_1 + \tfrac{1}{2\kappa+1} r_2 + \cdots + \tfrac{1}{n\kappa+1} r_n\right]\ . \end{aligned}$$

The DGE allows implementation of hyperbolic discounting based on an ensemble of exponentially discounting modules. The model can explain decision-making behavior observed in humans based on the hypothesis that the striatum has a modular structure (Tanaka, Schweighofer, et al. 2007). The model explains how model-free, value-based processes in the brain could give rise to the hyperbolic discounting observed in humans. The model can also be used to show pre-commitment behavior (Kurth-Nelson & Redish 2010). Furthermore, a connection between average reward and hyperbolic discounting makes it possible to maximize average reward under the condition that the tasks have a structure similar to that of multi-armed bandits (Reinke, Uchibe, & Doya 2015).

1.5 Conclusion

The IGE is a reinforcement learning method to solve multi-step decision-making problems where the problems are formalized as MDPs. The idea behind the IGE is very general, i.e. to learn several policies based on independent modules, each representing a reinforcement learner that optimizes the expected discounted reward sum with a different discount factor. It can therefore be implemented by model-free or model-based algorithms, and value-based or policy-based algorithms. The work in this thesis is based on Q-learning, which is a model-free, value-based TD approach. It allows the IGE to be used in tasks for which no model is given and to avoid the complication of a model-learning step. Furthermore, the value function can be used to deduct further information about the learned policies, and Q-learning's off-policy ability allows the modules to learn in parallel.

The IGE is part of the field of transfer learning, which uses knowledge gained from source tasks to improve learning in a target task. In contrast to existing methods, the IGE is the first algorithm that uses different discounting schemes to identify policies applicable to different task conditions or objectives.

Although the idea behind the IGE is new in the field of transfer learning, it was already introduced by Horde architecture. Horde uses different discounting modules purely for predictions on different timescales, whereas, the IGE uses them to identify different behaviors for the agent that help it to adapt to different situations.

The original inspiration for the IGE came from neuroscientific findings suggesting that the brain has a modular structure with different discounting modules. These findings also inspired a similar model, the DGE, which has been proposed as a model for the hyperbolic discounting behavior of humans in inter-temporal decision-making tasks. Unlike the IGE, modules of the DGE depend on each other. They can only learn a single behavior and not several, as the modules of the IGE.

Chapter 2

The Independent γ-Ensemble

The Independent γ-Ensemble (IGE) is a modular reinforcement learning framework. Each module has a state-action value function with a different discount factor γ. The values are learned via Q-learning. The IGE has two properties making it a suitable framework for adaptation to different task conditions and problems.

First, it learns different policies for a single environment, because of the different discount factors for each module. The policies differ in their preferences for the amount of reward and the time needed to gain rewards. This provides the agent with a set of policies that it can use to adapt its behavior to different task conditions. The second property of the IGE is its ability to decode information about the expected reward trajectory of its policies. It can identify how much reward can be expected from a policy and how much time is needed to obtain it. This allows the agent to select the most appropriate learned policy under a certain task condition.

The next section introduces the IGE framework (Section 2.1), followed by sections that discuss the two properties of the IGE (Sections 2.2 and 2.3). Two further sections discuss how discount parameters for the modules should be chosen (Section 2.4), and the effect of the IGE on the exploration behavior (Section 2.5).

2.1 The IGE Framework

The Independent γ-Ensemble (IGE) (Fig. 2.1) is composed of several independent γ-modules. The modules are Q-functions with different discount factors $\gamma \in (0, 1)$:

$$\begin{aligned} Q_\gamma(s_t, a_t) &= \mathrm{E}\big[r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} + \cdots\big] \\ &= \mathrm{E}\big[r_t + \gamma \max_{a_{t+1}} Q_\gamma(s_{t+1}, a_{t+1})\big] \ . \end{aligned} \tag{2.1}$$

The state values are defined by:

$$V_\gamma(s) = \max_a Q_\gamma(s, a) \ .$$

Each module has its independent policy, called a γ-policy, that is defined based on its values:

$$\pi_\gamma(a; s) \sim Q_\gamma(s, a) \ .$$

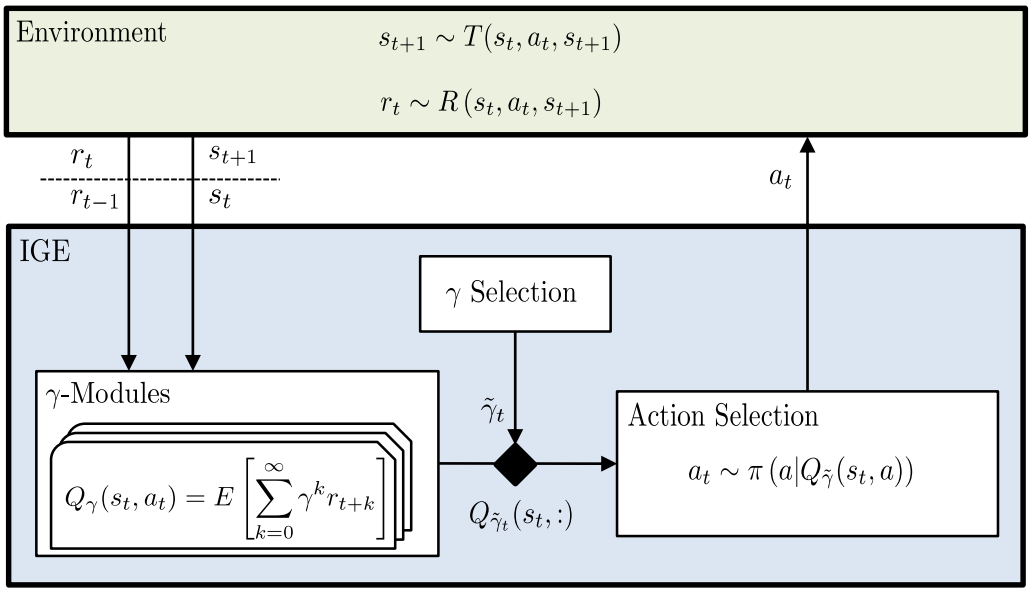


Figure 2.1: The Independent γ-Ensemble framework.

Probabilities for actions depend on the used action selection strategy, such as softmax or ϵ-greedy. The number of modules M and their discount parameters $\Gamma = (\gamma_1, \ldots, \gamma_M)$ are meta-parameters of the algorithm.

The values of each module are learned with a Q-learning approach (Algorithm 2.1). After an observation of (s_t, a_t, r_t, s_{t+1}) the values of all modules in Γ are updated by:

$$Q_\gamma(s_t, a_t) \leftarrow Q_\gamma(s_t, a_t) + \alpha \left(r_t + \gamma \max_{a_{t+1}} Q_\gamma(s_{t+1}, a_{t+1}) - Q_\gamma(s_t, a_t) \right)$$

with the learning rate $\alpha \in [0, 1]$. Although modules might have different policies, all modules can be updated by the same observations, because Q-learning is an off-policy algorithm that does not depend on the policy that produced the observation. Given enough exploration and a decaying learning rate α, γ-modules are guaranteed to converge to their optimal policies because they are Q-learning processes (Tsitsiklis 1994; Watkins & Dayan 1992).

The policy of the IGE agent is defined by the policy of one of its modules:

$$\pi(a|s) = \pi_{\tilde{\gamma}}(a|s) \ .$$

The active module $\tilde{\gamma}$ is either selected at the beginning of an episode or at each step. Which module is active and how the module selection works depends on the task that should be accomplished. Different selection methods and their application scenarios are discussed in later chapters.

2.2 Learning a Set of Policies

The independent modules of the IGE allow it to learn different policies, because the optimal policy π^*_γ of a module depends on its discount factor γ. Having a set of policies

Algorithm 2.1: Independent γ-Ensemble

Input:
Learning rate: $\alpha \in [0, 1]$
Sorted list of discount factors: $\Gamma = (\gamma_1, \ldots, \gamma_m)$ with $\gamma_i \in (0, 1)$

initialize $Q_\gamma(s, a)$ to zero
repeat (for each episode)
  initialize state s
  repeat (for each step in episode)
    `// selection of the active module`
    $\tilde{\gamma} \leftarrow \gamma$-module selection
    $a \leftarrow$ choose an action for s according to $\pi_{\tilde{\gamma}}(a|s)$ (e.g. ϵ-greedy)
    r, $s' \leftarrow$ take action a, observe outcome
    `// value update of all modules based on the observation`
    forall the $\gamma \in \Gamma$ **do**
      $Q_\gamma(s, a) \leftarrow Q_\gamma(s, a) + \alpha \left(r + \gamma \max_{a'} Q_\gamma(s', a') - Q_\gamma(s, a)\right)$
    end
    $s \leftarrow s'$
  until *s is terminal-state*
until *termination*

provides an agent with a behavioral repertoire for an environment to adapt to different situations and contexts. This section analyzes which policies can be learned by an IGE, and if a mapping from the discount factors to the type of policies exists. The policies that the IGE learns are called γ-policies.

The important aspect of the policies and the way in which they are analyzed is by the trajectories they produce. A trajectory τ_π is the states, actions, and rewards that are observed during a trial starting from state s_0 and following policy π:

$$\tau_\pi = \big((s_0, a_0, r_0), (s_1, a_1, r_1), \ldots, (s_{T-1}, a_{T-1}, r_{T-1}), s_T\big) \ .$$

The trajectories of a policy depend on the structure of the MDP, i.e. its transition probabilities and the reward function. The important property, that determines if the policies that produce the trajectories can be learned by the IGE, is the trajectories rewards, because the general objective of each γ-module is to maximize them. In each episode a policy can result in a different trajectory if the policy, the transitions, or the rewards in an MDP are probabilistic. To analyze which policies are learnable the expected reward per time step over all possible trajectories of a policy is used:

$$h_\pi(s_0) = \big(\mathrm{E}[r_0], \mathrm{E}[r_1], \ldots, \mathrm{E}[r_{T-1}]\big) \ ,$$

because it is the basis for calculating the Q-values (Eq. 2.1). For each state s in an MDP, a set of potential expected reward trajectories $H(s)$ exists for every possible policy π: $h_\pi(s) \in H(s)$.

The value of a state s for policy π is the sum over the discounted expected reward trajectory that the policy produces:

$$V_\gamma^\pi(s) = V_\gamma\big(h_\pi(s)\big) = \gamma^0 \mathrm{E}[r_0] + \gamma^1 \mathrm{E}[r_1] + \ldots + \gamma^{T-1} \mathrm{E}[r_{T-1}] \ . \tag{2.2}$$

The optimal policy for a module is the one that produces the trajectory that maximizes the module's value function for all states:

$$\forall s \in S: \quad \pi_\gamma^*(s) = \underset{\pi}{\operatorname{argmax}}\, V_\gamma\big(h_\pi(s)\big) \ .$$

In conclusion, analyzing which policies are learned by an IGE is connected to the analysis of the trajectories they produce.

The main questions in this section are, what reward trajectories an IGE learns, and whether a mapping from discount factors to the type of trajectories they learn can be defined. The next part analyzes these questions based purely on the theoretical set of possible expected reward trajectories, independent of MDPs in which they would be possible. Afterward, the relationship between the structure of MDPs and the reward trajectories that are learned for them is discussed. The last part describes this relationship in more detail for MDPs with several goal states which are the main application scenarios for the IGE explored in this thesis.

2.2.1 Type of Learned Reward Trajectories

This section analyzes the expected reward trajectories that an IGE learns, independently of the MDPs in which such trajectories could be possible. The analysis concentrates only on the types of trajectories that would be learned and their mapping to certain discount factors, given a theoretical set of expected reward trajectories H.

An expected reward trajectory is a solution for the IGE, if at least one $\gamma \in [0, 1]$ exists for which it has the maximum value:

$$\exists \gamma: \ \forall h' \neq h \in H: \ V_\gamma(h) > V_\gamma(h') \ . \tag{2.3}$$

A simple observation from this formulation is that the set of trajectories that can be learned by an IGE is based on their comparison with all possible trajectories. Thus, a certain discount factor does not guarantee learning of trajectories with criteria on specific expected reward values, such as having a certain reward sum. Instead, it learns a trajectory if it differs in a certain way from other trajectories in H.

The question becomes under what circumstances is a trajectory optimal for one module but not for another. This depends on the influence of the discount factor on the value for a trajectory, because the value defines whether it is learned by a module (Eq. 2.3). The discount parameter γ defines how strongly the expected reward for each future time step t is weighted in the sum of the value. Its influence can be represented as a weight ω_t:

$$V_\gamma\big(h_\pi(s)\big) = \omega_0 \mathrm{E}[r_0] + \omega_1 \mathrm{E}[r_1] + \omega_2 \mathrm{E}[r_2] + \ldots + \omega_{T-1} \mathrm{E}[r_{T-1}] \ , \text{ with } \ \omega_t = \gamma^t \ .$$

For $\gamma < 1$ the weights diminish exponentially to 0 over the future steps t:

$$\forall \gamma \in [0, 1): \ \lim_{t \to \infty} \gamma^t = 0 \ .$$

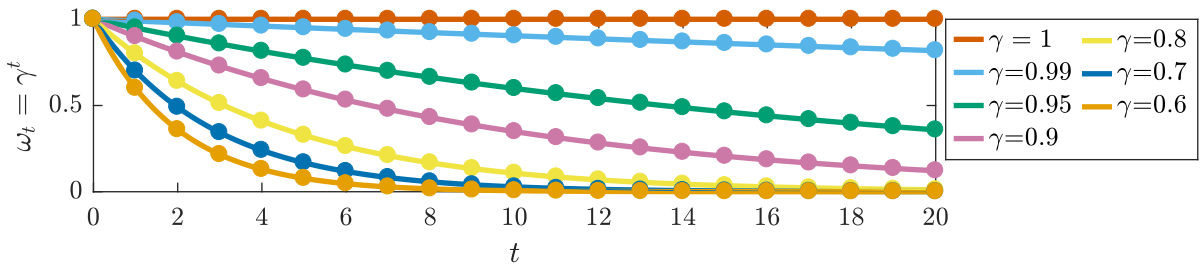


Figure 2.2: The discount parameter γ defines the weighting of the expected reward for each future step in the value sum. Small γ's reduce the weight for the expected reward of distant time steps. Large γ's do not reduce the weight of distant time steps as much.

The weights diminish faster to 0 the smaller γ is (Fig. 2.2). The result of the different weighting for different γ's is that large γ's take distant expected rewards into account more strongly. Thus, such modules maximize the long-term sum over the expected reward, whereas, for values based on small γ's, distant rewards have small weights and do not influence the sum as much. Their modules optimize the sum over expected rewards on a short-term scale.

A stronger criterion for the structure of the learnable set of expected reward trajectories, i.e. the height of their expected rewards and their distribution of the rewards over future time points, is difficult to formalize for MDPs in general. The value of a trajectory (Eq. 2.2) includes all expected rewards of all future time steps. Therefore, any difference in their height can influence whether one trajectory has a higher value than another. Nonetheless, it is possible to define a set of dominated trajectories that are not learnable by the IGE (Theorem 1).

Theorem 1. *An expected reward trajectory h_a dominates another trajectory h_b ($h_a \gg h_b$) if:*

$$\forall T \in \mathbb{N}^+ : \quad \sum_{t=0}^{T-1} r_{a,t} > \sum_{t=0}^{T-1} r_{b,t} \ .$$

Dominated trajectories cannot be learned by an IGE.

For example, the trajectory $h_a = (0, 2, 0)$ is dominating $h_b = (0, 0, 1)$ ($h_a \gg h_b$), because h_b results in a smaller reward and needs more steps. Trajectory h_b is therefore not learned by the IGE ($\forall \gamma \in [0, 1] : \ V_\gamma(h_a) > V_\gamma(h_b)$). Theorem 1 is only a sufficient condition and not necessary. Not all non-dominated trajectories are necessarily solutions for the IGE. For example, the two trajectories $h_a = (1, 0, 5)$ and $h_b = (0, 2, 0)$ do not dominate each other, but h_b is not a solution for the IGE ($\forall \gamma \in [0, 1] : \ V_\gamma(h_a) > V_\gamma(h_b)$).

In summary, the IGE learns policies that maximize the expected reward trajectories under different weightings of nearby and distant rewards. Strict restrictions on the form of learnable trajectories are difficult to formalize for general MDPs. In general, small discount factors learn trajectories that maximize short-term expected rewards, whereas large discount factors maximize long-term expected rewards.

2.2.2 Type of Learned Policies in General MDPs

The previous section discussed policies that an IGE learns in terms of their expected reward trajectories. This part discusses the influence of the structure in MDPs, i.e. their transition probabilities and rewards, on the set of possible trajectories that the IGE learns.

The policy that a module learns depends on the expected rewards that it produces and their time points: $h_\pi(s) = (\mathrm{E}[r_0], \mathrm{E}[r_1], \ldots, \mathrm{E}[r_n])$. The expected reward for a time step t depends on the expected reward for the states that can be reached by this time point:

$$\mathrm{E}\left[r_t\right] = \int_S \Pr(s_t|s,\pi)\mathrm{E}[R(s_t,a_t)]ds_t \ ,$$

where $\mathrm{E}[R(s_t,a_t)]$ is the expected reward for action a_t in state s_t, and $\Pr(s_t|s,\pi)$ is the probability of reaching state s_t at time point t if starting in state s and following policy π. It is defined by:

$$\Pr(s_t|s,\pi) = \prod_{k=1}^{t} \Pr(s_k|s_{k-1},a_{k-1}) \ ,$$

where $\Pr(s_k|s_{k-1},a_{k-1})$ is the transition probability function of the MDP. For simplicity, a greedy policy is assumed with $a_t = \pi(s_t)$. In summary, the expected reward trajectories depend on the transition probability function and the reward function of an MDP. Both are the factors that define which policies can be learned by the IGE.

Fig. 2.3 shows two MDPs that exemplify how certain transition and reward functions allow the learning of different policies and what they represent. In both MDPs, two

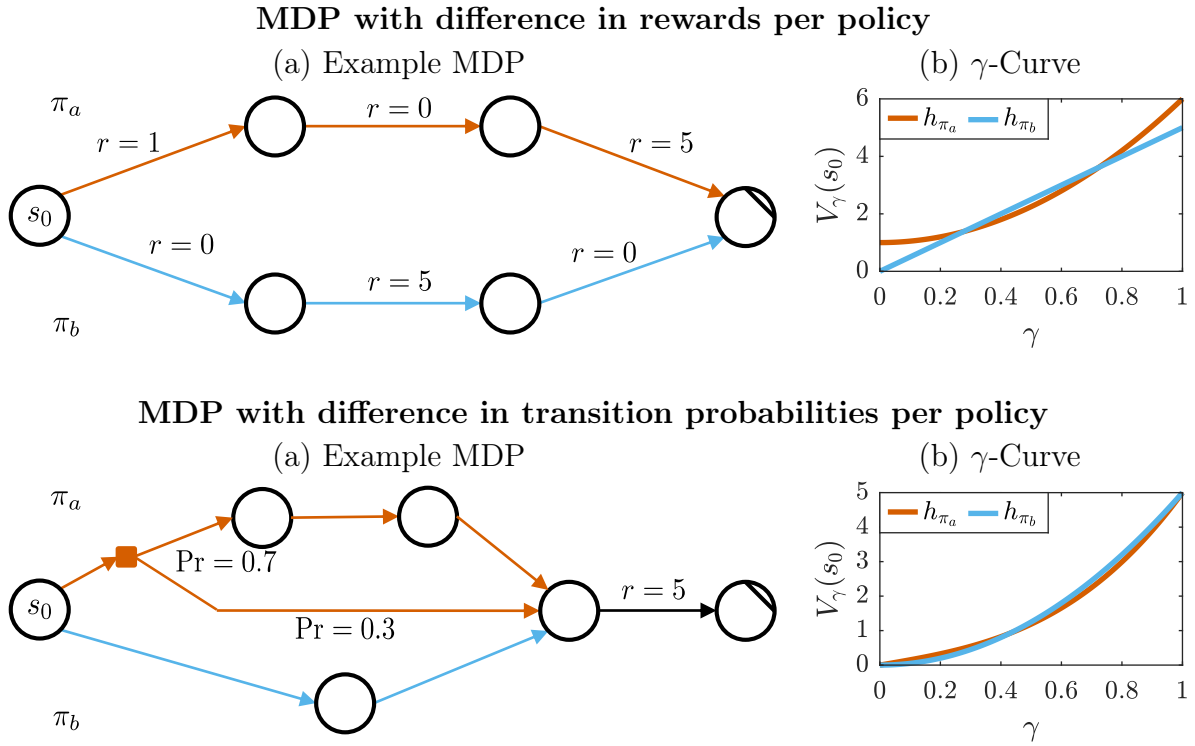


Figure 2.3: Two example MDPs that show how rewards and transition probabilities result in policies that can be learned by different γ-Modules.

policies are possible. Either the agent follows the upper route (π_a) or the lower route (π_b).

The first MDP is deterministic. The upper route results in an expected reward trajectory of $h_{\pi_a}(s_0) = (1, 0, 5)$, and the lower route in $h_{\pi_b}(s_0) = (0, 5, 0)$. The values of the two policies for different discount factors for state s_0 show that the IGE is learning both policies (Fig. 2.3, b). Modules with $0 < \gamma < 0.28$ learn π_a, because the trajectory of this policy has a higher immediate expected reward of $r_0 = 1$, compared to $r_0 = 0$ of π_b. For intermediate discount factors with $0.28 < \gamma < 0.72$, policy π_b is learned. The larger discount factors give more weight to intermediate rewards of $r_1 = 5$ for π_b compared to $r_1 = 0$ of π_a. For large γ's with $0.72 < \gamma < 1$, policy π_a is again the solution because over all time steps it has a higher expected reward sum. The policies that the IGE learns in this MDP represent different solutions to the trade-off between the reward sum that can be achieved in a certain number of steps and the amount of steps needed.

The second MDP shows how transition probabilities can induce the learning of different policies by an IGE. In this MDP, only the last state transition gives a reward of 5, all others give 0. The upper (π_a) and lower route (π_b) differ in their transition probabilities. The upper route is stochastic and reaches the goal state in 30% of all trials with 2 steps and in 70% of all trials with 4 steps. Its expected reward trajectory is $h_{\pi_a}(s_0) = (0, 1.5, 0, 3.5)$. The lower route is deterministic. It needs 3 steps to reach the goal, resulting in a reward trajectory of $h_{\pi_a}(s_0) = (0, 0, 5)$. Because $h_{\pi_a}(s_0)$ has a higher immediate expected reward for $t = 1$, it is the solution for smaller discount factors with $0 < \gamma < 0.43$. For larger γ's π_b is the solution because of its higher intermediate expected reward at $t = 2$. The policies that the IGE learns in this MDP represent different risk levels to reach the goal state within the shortest number of steps. The expected number of steps of π_a ($\mathrm{E}_{\pi_a}[N] = 0.3 \cdot 2 + 0.7 \cdot 4 = 3.4$) is higher than that of π_b ($\mathrm{E}_{\pi_b}[N] = 3$), but it has the probability of being shorter in 30% of the cases.

Both MDPs show that the IGE learns policies that represent different strategies. In the first MDP, the different strategies represent solutions for the trade-off between reaching a high reward sum and the number of steps needed. Smaller γ's provide solutions with fewer steps and larger γ's with greater rewards. In the second MPD, different strategies represent solutions for reaching the goal within the least number of steps that have different risk levels. Smaller γ's learn solutions that have a chance for using fewer steps, but also with the risk of having more.

2.2.3 Type of Learned Policies in MDPs with several Goal States

The next part concentrates on the analysis of policies and their trajectories in a class of MDPs that are used as the main application scenario for the IGE. These are episodic MDPs with several goal states. Depending on the distance of goal states and rewards that can be obtained for reaching them, the IGE learns policies to reach different goals. The policies represent solutions to the trade-off between the amount of reward an agent can gain and the number of steps required. The next part introduces an example of such an MDP. Afterward, the types of trajectories that can be learned in such MDPs and the relationship between γ and their learned trajectories is analyzed. The analysis is based on deterministic, goal-only-reward MDPs which allow a stronger theoretical analysis.

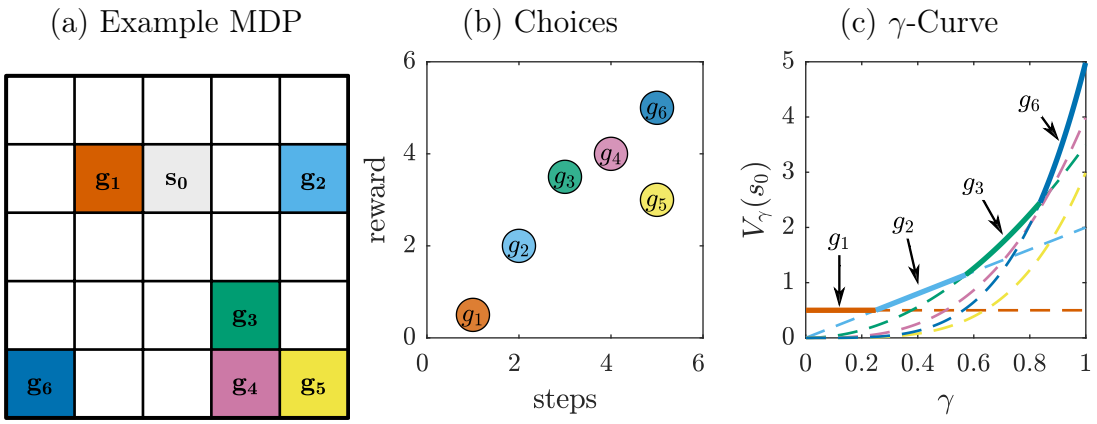


Figure 2.4: (a) A grid-world MDP with 6 goal states and one start state (s_0). (b) Each goal gives a different amount of reward and is a certain number of steps away from the initial state. (c) The values for the optimal trajectories to reach each goal state show that the IGE learns the policies to reach 4 of the goals.

Grid World Example

A grid-world example of an MDP with several goal states is shown in Fig. 2.4. It consists of 5×5 states and has 6 goal states. The agent starts in state s_0. It only receives a reward if it enters one of the goal states. Each goal gives a different amount of reward and can be reached with a certain minimal number of steps if the agent starts in s_0 (Fig. 2.4, b).

The γ-Curve shows the values of all γ-modules for each optimal trajectory to a goal state (Fig. 2.4, c). The maximum value shows which trajectory is optimal ($h^*_{\pi^*} = \text{argmax}_{h_\pi} V_\gamma(h_\pi)$). The IGE learns the policies to reach 4 of the 6 goals (Fig. 2.5). Reaching g_4 and g_5 are not in the set of learned solutions. Learned policies represent different solutions for the trade-off between the amount of rewards and the number of steps that are needed to achieve them. Modules with small γ's learn policies to reach goal states that are close to the start state, whereas modules with larger γ's learn policies to reach more distant goal states that provide greater rewards.

Deterministic, Goal-Only-Reward MDPs

The relationship between the discount factors and the trajectories they learn can be further analyzed for the class of deterministic, goal-only-reward MDPs. The value formulation for these MDPs can be simplified, because non-zero rewards are only given if a goal state is reached. This allows to formalize stronger restrictions on the possible γ-policies. First the MDPs are defined and the concept of a choice is introduced that is used to describe the optimal trajectories in such MDPs.

Definition 3. A deterministic, goal-only-reward MDP is an episodic MDP with a deterministic transition function:

$$T : S \times A \times S \mapsto \{0, 1\} \quad , \text{ and } \quad \forall s_t, a_t : \sum_{s_{t+1} \in S} T(s_t, a_t, s_{t+1}) = 1 \ ,$$

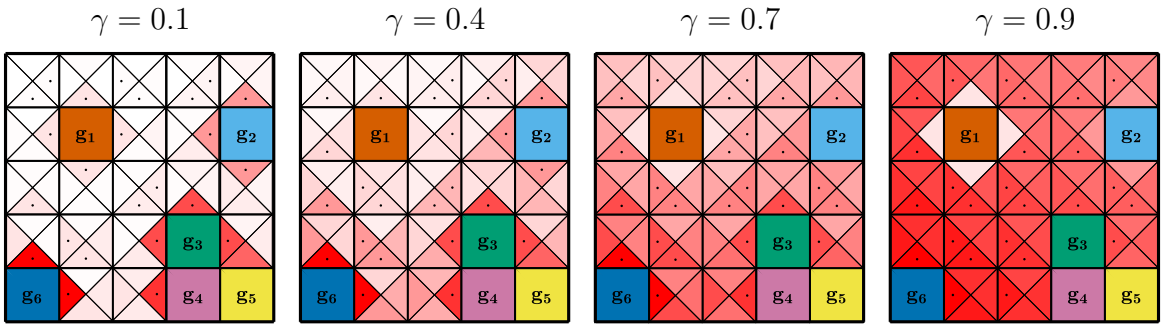


Figure 2.5: The γ-modules have different optimal Q-functions and therefore different policies for MDPs with several goal states, such as for the example in Fig. 2.4. The Q-values for each state and movement direction are shown as red coloration. The dots mark the optimal actions per state.

and where non-zero rewards are only given for transitions into goal states:

$$\forall s_{t+1} \notin S^G : \; R(s_t, a_t, s_{t+1}) = 0,$$

where $S^G \subset S$ is the set of goal states.

The goal states in such MDPs represent choices between which the agent can choose. Each choice describes the optimal trajectory to reach a certain goal state, i.e. the trajectory with smallest number of steps.

Definition 4. A choice for a goal state g is a tupel (r, n) with the expected reward $r = E[R(g)]$ and the minimum number of steps n to reach g from the current state s. The set of all possible choices $C(s)$ holds a choice for each reachable goal-state from state s. Goal states with the same reward r and number of steps n are combined into one choice.

Goal states with the same expected reward and steps are combined to simplify the proofs that build upon the concept of choices. If a γ-module learns the optimal policy to reach the goals of such a combined choice, it learns to go with equal probabilities to each goal state because their values are equal. In each episode the agent might take a different trajectory over states, but the expected reward trajectory h is the same. Because the expected reward trajectory is the important element of the analysis for policies that modules learn, the goal states can be combined into one choice without loss of generality.

In deterministic, goal-only-reward MDP's the possible set of trajectories $H(s)$ that the IGE could learn starting in state s is given by its choices $C(s)$. The set of choices can be ordered $(c_1 < \ldots < c_\mathrm{K})$, based on their number of steps from the shortest to the longest $n_i < n_{i+1}$. This also results in an ordering of their reward values: $r_i < r_{i+1}$.

The advantage of these types of MDPs is that the value formulation for the choices can be simplified, making it possible to analyze in more detail which choices the IGE can learn.

Theorem 2. *The value of a choice $c = (r, n)$ is given by:*

$$V_\gamma(c) = \gamma^{n-1} r \; . \tag{2.4}$$

Proof. The value of a choice is the value for the optimal trajectory to the goal state:

$$V_\gamma(c) = E[r_0 + \gamma r_1 + \ldots + \gamma^{n-1} r_{n-1}] \ .$$

In deterministic, goal-only-reward MDPs the length of the trajectory is fixed to n and all rewards before $t = n - 1$ are zero. Therefore, they can be excluded from the expression resulting in Eq. 2.4. □

The region of discount factors that would learn a certain choice can be defined (Theorem 3). This definition is the basis for analyzing the set of choices that the IGE learns and the mapping of the discount factors to the choices they learn.

Theorem 3. *The γ-region for which a choice c is the outcome in a deterministic, goal-only-reward MDP is defined by:*

$$\max_{\forall c_i < c} \gamma_E(c_i, c) < \gamma < \min_{\forall c_j > c} \gamma_E(c, c_j) \ , \tag{2.5}$$

where c_i are all choices that are smaller, and c_j are all choices that are larger than c. $\gamma_E(c_a, c_b)$ is the discount factor for which two choices ($c_a < c_b$) result in the same value ($V_{\gamma_E}(c_a) = V_{\gamma_E}(c_b)$). It is defined by:

$$\gamma_E(c_a, c_b) = \left(\frac{r_a}{r_b}\right)^{\frac{1}{n_b - n_a}} \ , \tag{2.6}$$

where $c_a = (r_a, n_a)$, and $c_b = (r_b, n_b)$. For discount factors $0 < \gamma_a < \gamma_E$ the values for choice c_a are larger than for c_b. For $\gamma_E < \gamma_b \leq 1$ the values of c_b are larger.

Proof. First, Eq. 2.6 is proven before Eq. 2.5 is shown. Given two non-dominated choices with $c_a < c_b$, it can be defined for which discount factor region one has the maximum value. γ_E (Eq. 2.6) is the boundary between both regions for which both choices have the same value:

$$V_{\gamma_E}(c_a) = V_{\gamma_E}(c_b) \quad \Leftrightarrow \quad \gamma_E^{n_a - 1} r_a = \gamma_E^{n_b - 1} r_b \quad \Leftrightarrow \quad \gamma_E^{n_b - n_a} = \frac{r_a}{r_b} \quad \Leftrightarrow \quad \gamma_E = \left(\frac{r_a}{r_b}\right)^{\frac{1}{n_b - n_a}} \ ,$$

showing Eq. 2.6. Following the same steps to determine the inequality between the two values shows that c_a is the outcome ($V_\gamma(c_a) > V_\gamma(c_b)$) for $0 < \gamma_a < \gamma_E$. Whereas, for $\gamma_E < \gamma_b \leq 1$ the outcome ($V_\gamma(c_a) < V_\gamma(c_b)$) is c_b.

This can be used to define the γ-region for which a choice is the outcome of γ-modules (Eq. 2.5). A choice is the outcome for a γ-module if its value is the maximum over all choices: $\text{argmax}_{c \in C} V_\gamma(c)$. The γ-region for which choice c has a higher value than any smaller choice c_i and any larger choice c_j is given by Eq. 2.5. □

Based on the definition of which γ-region has a certain choice as its outcome (Theorem 3), it is possible to define the set of choices that an IGE can learn in a deterministic, goal-only-reward MDP. The set is called the IGE choice set C_{IGE} and it is defined as the unique set of all choices that are the outcome of all possible γ-modules:

$$C_{\text{IGE}} = \left\{ \operatorname*{argmax}_c V_\gamma(c) \right\}_{\forall \gamma \in [0,1]} \ .$$

A choice is part of the IGE choice set if an existing γ-region, defined by Eq. 2.5, encodes it. This depends on the other choices in the MDP. It is therefore only possible to determine whether a choice is part of the IGE choice set by comparing it to the other choices in the MDP.

A choice is not part of the set if it is dominated by another choice (Theorem 4). The notion of domination follows that used in Theorem 1 for MDPs in general.

Theorem 4. *A choice c_b is dominated by another choice c_a ($c_a \gg c_b$) if it is longer $n_b \geq n_a$ and it has less reward $r_b \leq r_a$. A dominated choice is not part of the IGE choice set ($c_b \notin C_{IGE}$).*

Proof. The value of the dominated choice is smaller than the dominating choice's value for all discount factors with $\gamma \in (0, 1)$. If $n_a \leq n_b$ and $r_a \geq r_b$, then:

$$\gamma^{n_a-1} \geq \gamma^{n_b-1} \quad \Leftrightarrow \quad \gamma^{n_a-1} r_a \geq \gamma^{n_b-1} r_b \quad \Leftrightarrow \quad V_\gamma(c_a) \geq V_\gamma(c_b) \ .$$

Therefore, dominated choices are never outcomes of γ-modules. □

To analyze which choices are part of an IGE choice set, only non-dominated choices need to be considered. Based on the list of non-dominated choices ($c_1 < \ldots < c_\mathrm{K}$), the smallest and the largest choice, i.e. the choice with the smallest and largest number of steps, are part of the IGE choice set. They are the outcomes for discount factors near 0 and 1 (Theorems 5 and 6).

Theorem 5. *The non-dominated choice c_1 with the fewest steps is part of the IGE choice set ($c_1 \in C_{IGE}$).*

Proof. The non-dominated choice with the least number of steps is the outcome for modules with:

$$0 < \gamma < \min_{\forall c_i > c_1} \gamma_E(c_1, c_i) \ .$$

To prove its existence, it is enough to show the general case of:

$$\forall c_i > c_1 : \quad \gamma_E(c_1, c_i) > 0 \quad \Leftrightarrow \quad \left(\frac{r_1}{r_i}\right)^{\frac{1}{n_i - n_1}} > 0 \ ,$$

which is true because $x^y > 0$ for $x > 0$. □

Theorem 6. *The non-dominated choice c_K with the most steps is part of the IGE choice set ($c_\mathrm{K} \in C_{IGE}$).*

Proof. The last non-dominated choice c_K is the outcome of modules with:

$$\max_{\forall c_i < c_\mathrm{K}} \gamma_E(c_i, c_\mathrm{K}) < \gamma < 1 \ .$$

To prove its existence, it is enough to show the general case of:

$$\forall c_i < c_\mathrm{K} : \quad \gamma_E(c_i, c_\mathrm{K}) < 1 \quad \Leftrightarrow \quad \left(\frac{r_i}{r_\mathrm{K}}\right)^{\frac{1}{n_\mathrm{K} - n_i}} < 1 \ ,$$

which is true because $r_\mathrm{K} > r_i \Leftrightarrow \frac{r_i}{r_\mathrm{K}} < 1$ and $0 < x^y < 1$ for $0 < x < 1$ and $y > 0$. □

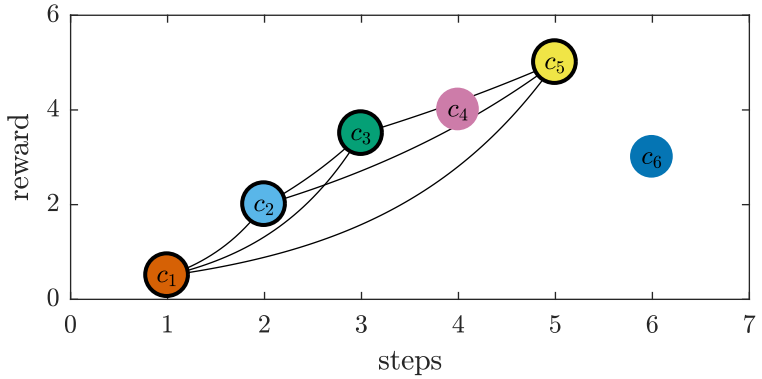


Figure 2.6: The reward boundary (Theorem 7) between all choice combinations shows which choices are part of the IGE choice set for the example MDP of Fig. 2.4. Choices c_1 and c_5 are in the set because they are the shortest and longest non-dominated choice (Theorems 5 and 6). Choices c_2 and c_3 fulfill the IGE choice set boundary. Choice c_4 is not the in the IGE choice set because it is below the boundary spanned by c_3 and c_5. Choice c_6 is excluded because it is dominated by c_3, c_4 and c_5.

Any other non-dominated choice between the smallest and largest choice ($c_2 < \ldots < c_{\mathrm{K}-1}$) has to fulfill a reward criterion that depends on the number of steps for that choice and the other non-dominated choices (Theorem 7). The criterion is an exponential boundary between each possible choice pair combination (Fig. 2.6).

Theorem 7. *Any non-dominated choice $c = (r, n)$ between the smallest and largest choice $c_1 < c < c_{\mathrm{K}}$ that fulfills the following reward criterion is part of the IGE choice set ($c \in C_{IGE}$):*

$$r > \max_{\forall c_a < c, \forall c_b > c} \exp\left(\frac{\log(r_a)(n - n_b) + \log(r_b)(n_a - n)}{n_a - n_b}\right) , \tag{2.7}$$

where $c_a = (r_a, n_a)$ are all smaller non-dominated choices, and $c_b = (r_b, n_b)$ are all larger non-dominated choices.

Proof. The criterion is defined by the maximum over the criterion of each possible pair (c_a, c_b) of non-dominated choices, where c_a is smaller and c_b is larger than c. If only such a pair of choices exists, then choice c is in the IGE choice set if a γ-region exists such that has it as outcome:

$$\gamma_E(c_a, c) < \gamma_c < \gamma_E(c, c_b) .$$

Thus, the choice c needs to fulfill:

$$\begin{aligned}
&\gamma_E(c_a, c) < \gamma_E(c, c_b) \;\Leftrightarrow\; \left(\frac{r_a}{r}\right)^{\frac{1}{n-n_a}} < \left(\frac{r}{r_b}\right)^{\frac{1}{n_b-n}} \;\Leftrightarrow\; r^{\frac{1}{n_b-n}+\frac{1}{n-n_a}} > r_a^{\frac{1}{n-n_a}} r_b^{\frac{1}{n_b-n}} \\
&\Leftrightarrow\; r^{\frac{n_b-n_a}{(n_b-n)(n-n_a)}} > r_a^{\frac{1}{n-n_a}} r_b^{\frac{1}{n_b-n}} \;\Leftrightarrow\; r > r_a^{\frac{(n_b-n)(n-n_a)}{(n_b-n_a)(n-n_a)}} r_b^{\frac{(n_b-n)(n-n_a)}{(n_b-n_a)(n_b-n)}} \\
&\Leftrightarrow\; \log r > \frac{(n_b - n)\log r_a + (n - n_a)\log r_b}{n_b - n_a} \\
&\Leftrightarrow\; r > \exp\left(\frac{(n_b - n)\log r_a + (n - n_a)\log r_b}{n_b - n_a}\right) .
\end{aligned}$$

This needs to be guaranteed for all possible choice pairs (c_a, c_b) in the MDP which results in Eq. 2.7 and finishes the proof of Theorem 7. □

In summary, the IGE choice set is the subset of choices that an IGE is able to learn. It consists of the choices with the fewest steps, the non-dominated choice with the most steps, and choices with a reward that is larger than the exponential reward bound defined by the other choices in the set.

The definition of the γ-region for which modules learn a certain choice (Theorem 3) also provides insight into the mapping of the choices that are learned by a certain γ-module. A direct mapping from a given γ to the choice that it learns, i.e. its number of steps n and reward r, is not possible. Instead, it can only guarantee that the choice associated with a γ-module has a certain reward and number of steps in comparison to the other choices in the MDP.

Theorem 3 shows that the choice $c = (r, n)$ that is learned by a module γ fulfills the following criteria in relation to smaller and larger choices:

$$\gamma_E(c_i, c) < \gamma < \gamma_E(c, c_j) \quad \Leftrightarrow \quad \left(\frac{r_i}{r}\right)^{\frac{1}{n-n_i}} < \gamma < \left(\frac{r}{r_j}\right)^{\frac{1}{n_j-n}} \ ,$$

where $c_i = (r_i, n_i)$ is the smaller and $c_j = (r_j, n_j)$ is the larger choice. The upper and lower criteria depend both on the quotient between the reward of choice c and the other choices ($\frac{r_i}{r}$, $\frac{r}{r_j}$) and the difference between their steps ($n - n_i$, $n_j - n$). Reformulating the lower criterion shows that the reward quotient depends on the step difference between choices:

$$\left(\frac{r_i}{r}\right)^{\frac{1}{n-n_i}} < \gamma \quad \Leftrightarrow \quad \frac{r_i}{r} < \gamma^{n-n_i} \ .$$

This means that choice c is guaranteed to be the outcome of γ if its reward quotient with all smaller choices is smaller than a certain value defined by the difference between the steps of the choices and γ. The criterion with choices larger than c is:

$$\gamma < \left(\frac{r}{r_j}\right)^{\frac{1}{n_j-n}} \quad \Leftrightarrow \quad \frac{r}{r_j} > \gamma^{n_j-n} \ .$$

In summary, the policy that a certain γ-module learns depends on other choices in the MDP. The important factors are the quotient between the rewards of the choices and the difference between their steps.

The IGE's ability to learn different policies in one framework for a single environment is the main difference between the IGE and classical methods such as Q-learning. Learned policies depend on the transition probabilities and rewards of an MDP. In MDP's with several goal states, the main application area of the IGE, the IGE learns policies to reach different goal states that have different distances and rewards. The policies represent solutions for the trade-off between amount of reward and the time to gain it. Modules with smaller discount factors learn policies to reach nearby goals, whereas policies of larger discount factors can reach distant goals that offer greater reward.

2.3 Decoding Information about Policies

The IGE learns different policies. This can be used to select the most appropriate policy for a given situation, such as to use a policy that results in fewer steps to reach reward, if the time to reach reward is important. To select the most appropriate policy, it would be helpful to know the outcome of it, i.e. what is its expected reward trajectory. The previous section showed that only a coarse mapping can be drawn from the discount factors to the expected trajectories that are their outcomes. In MDPs with several goal-states, smaller γ's result in gaining more rapid reward than larger γ's, which usually obtain more reward. But the discount factor does not give direct information about the length of a trajectory or its expected reward sum. Nonetheless, the IGE provides a way to decode such information from its values learned from several modules. At first, this possibility is discussed in the context of general MDPs. Although, it is theoretically possible to decode the full expected reward trajectory for policies that the IGE learns, such an approach has practical problems. For deterministic, goal-only-reward MDPs, these problems can be overcome, as discussed in the final part of this section.

2.3.1 General MDPs

The values of modules with similar discount factors are also similar ($\gamma_a \approx \gamma_b \rightarrow V_{\gamma_a}(s) \approx V_{\gamma_b}(s)$). Therefore, their policies are similar ($\pi_{\gamma_a} \approx \pi_{\gamma_b}$) and they often result in the same or similar expected reward trajectories ($h_{\pi_{\gamma_a}} \approx h_{\pi_{\gamma_b}}$). In deterministic environments, their trajectories are often the same ($h_{\pi_{\gamma_a}} = h_{\pi_{\gamma_b}}$). For example, in the MDP of Fig. 2.4 all modules with $0.57 < \gamma < 0.84$ learn the policy to reach goal state g_3 from s_0. All of them have the same expected reward trajectory.

Values of modules that have the same expected reward as outcomes can be defined by a matrix multiplication with:

$$V = G \times h,$$

$$\text{where } V = \begin{bmatrix} V_{\gamma_1}(s) \\ \vdots \\ V_{\gamma_T}(s) \end{bmatrix}, \; G = \begin{bmatrix} 1 & \gamma_1 & \gamma_0^2 & \cdots & \gamma_0^{n-1} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & \gamma_T & \gamma_T^2 & \cdots & \gamma_T^{n-1} \end{bmatrix}, \text{ and } h = \begin{bmatrix} E[r_0] \\ \vdots \\ E[r_n] \end{bmatrix}.$$

Based on this formulation, it is theoretically possible to decode the expected reward trajectory from the learned values with:

$$h = G^{-1} \times V \;, \tag{2.8}$$

where G^{-1} is the inverse of the γ-matrix. G is a Vandermonde matrix. The inverses of Vandermonde matrices are guaranteed to exist and they can be calculated by an explicit formula (Macon & Spitzbart 1958; Turner 1966). Therefore, given enough modules that learn policies for the same expected trajectory, the expected rewards of this trajectory could be decoded.

Unfortunately, this method has practical limitations making it difficult to apply to general MDPs. First, only in deterministic MDPs is the expected trajectory of modules with similar discount factors the same for two γ-modules ($h_{\pi_{\gamma_a}} = h_{\pi_{\gamma_b}}$). In stochastic MDPs, the trajectories are usually similar ($h_{\pi_{\gamma_a}} \approx h_{\pi_{\gamma_b}}$), but not identical because of the

stochastic transitions. Nonetheless, this approach might still provide an approximation of the expected trajectory for each module.

Second, the approach would first need to identify which modules have the same trajectories. This is a non-trivial problem, because only the values of each module, i.e. the γ-Curve as depicted for the example in Fig. 2.4 (c), are given. The curvature of the γ-Curve could provide a possible way to identify the regions that learned the same trajectory, but currently there is no practical method to do this.

Third, the method would require many modules, because it needs n modules to decode the expected reward trajectory for n steps into the future. The γ-regions that encode a certain policy are not known and they could have a small range. Therefore, many modules would be necessary to guarantee that enough modules learn the same policy and can be used to decode it.

The final problem is that the inverse G^{-1} of the γ-matrix is ill-conditioned for larger T (Gautschi 1990; Press et al. 2007). This means that small changes in the values V can result in big differences in the decoded trajectory h for Eq. 2.8. As a consequence, the approach becomes unstable to numerical errors such as rounding errors for small values. This effect is shown on trajectories with lengths of $T = (10, 20, 30)$. Only the last four

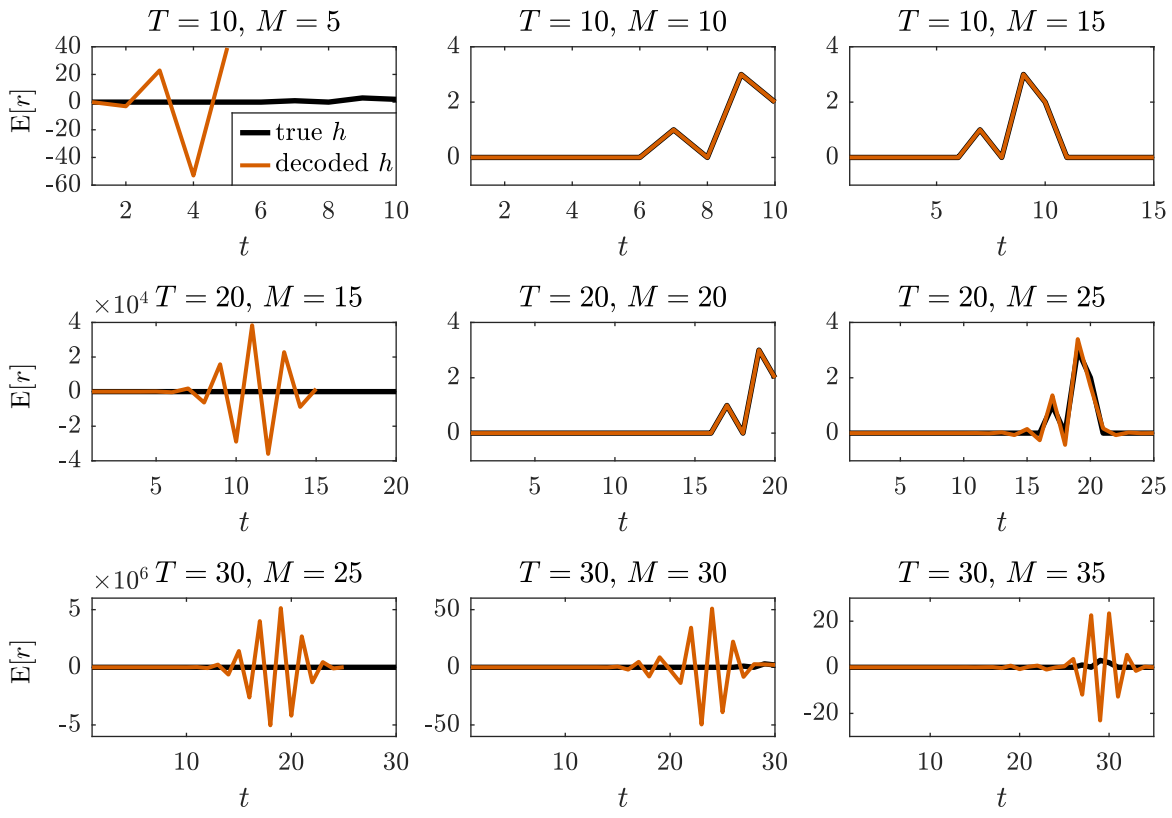


Figure 2.7: The difference between the true expected reward trajectory and the decoded trajectory with help of the inverse G^{-1} is large in cases where less modules are used than required (first column), and if the trajectory becomes too long (last row). The cases differ in the length of true trajectory T, and the number of modules used to decode the trajectory M .

time steps of each trajectory have non-zero expected rewards: $h = (r_0 = 0, \ldots, r_{T-4} = 1, r_{T-3} = 0, r_{T-2} = 3, r_{T-1} = 2)$. For each trajectory, values are calculated for γ-modules with discount factors that are linearly spaced between 0 and 1. The number of modules M is either five less than the length of the trajectory T, the same number, or five more modules. For each combination of the trajectory length and the number of modules the trajectory is decoded by Eq. 2.8 and compared to the original trajectory (Fig. 2.7). The results show that the decoding has large errors in two cases. If fewer modules are used than the number of steps of the original trajectory, and for the decoding of long reward trajectories ($T = 30$).

In summary, values of several modules of the IGE that learn to follow the same trajectory can be theoretically used to decode the expected reward trajectory. Nonetheless, the approach has practical limitations. It would require many modules to decode long trajectories, and it would first need to identify which modules follow the same trajectory. Moreover, the problem becomes ill-conditioned for long trajectories, introducing large errors in the decoded trajectory.

2.3.2 Deterministic, Goal-Only-Reward MDPs

Although, the approach for decoding the expected reward trajectory is limited to general MDPs, decoding can be done for deterministic, goal-only-reward MDPs, as shown in Fig. 2.4. In such MDPs, only the last step of a trajectory has a non-zero reward. Trajectories can therefore be represented as choices $c = (r, n)$ where r is the expected reward for entering the goal state after n steps. The values of two modules (γ_a, γ_b) that learn the trajectory to the same goal state can be used to decode r and n_s (Theorem 8).

Theorem 8. *If in deterministic, goal-only-reward MDPs, two modules (γ_a, γ_b) have the same choice $c = (r, n_s)$ as an outcome for state s, then the number of steps n_s can be decoded from their values by:*

$$n_s(\gamma_a, \gamma_b) = \frac{\log\big(V_{\gamma_a}(s)\big) - \log\big(V_{\gamma_b}(s)\big)}{\log(\gamma_a) - \log(\gamma_b)} + 1 \ , \tag{2.9}$$

and the expected reward r by:

$$r(\gamma_a, \gamma_b) = \frac{V_{\gamma_a}(s)}{{\gamma_a}^{n_s(\gamma_a,\gamma_b)-1}} \ , or \ \ r(\gamma_a, \gamma_b) = \frac{V_{\gamma_b}(s)}{{\gamma_b}^{n_s(\gamma_a,\gamma_b)-1}} \ . \tag{2.10}$$

Proof. The value of a module in a deterministic, goal-only reward environment is given by $V_\gamma(s) = \gamma^{n_s-1} r$. With values from two modules based on the same expected reward trajectory, the number of steps n_s to reach their goal state (Eq. 2.9) can be derived by:

$$\begin{matrix} V_{\gamma_a}(s) = {\gamma_a}^{n_s-1} r \\ V_{\gamma_b}(s) = {\gamma_b}^{n_s-1} r \end{matrix} \ \Leftrightarrow \ \begin{matrix} r = \frac{V_{\gamma_a}(s)}{{\gamma_a}^{n_s-1}} \\ r = \frac{V_{\gamma_b}(s)}{{\gamma_b}^{n_s-1}} \end{matrix} \ \Leftrightarrow \ \frac{V_{\gamma_a}(s)}{{\gamma_a}^{n_s-1}} = \frac{V_{\gamma_b}(s)}{{\gamma_b}^{n_s-1}} \ \Leftrightarrow \ \left(\frac{\gamma_a}{\gamma_b}\right)^{n_s-1} = \frac{V_{\gamma_a}(s)}{V_{\gamma_b}(s)}$$

$$\Leftrightarrow \ n_s - 1 = \log_{\left(\frac{\gamma_a}{\gamma_b}\right)} \frac{V_{\gamma_a}(s)}{V_{\gamma_b}(s)} \ \Leftrightarrow \ n_s = \frac{\log\big(V_{\gamma_a}(s)\big) - \log\big(V_{\gamma_b}(s)\big)}{\log(\gamma_a) - \log(\gamma_b)} + 1 \ .$$

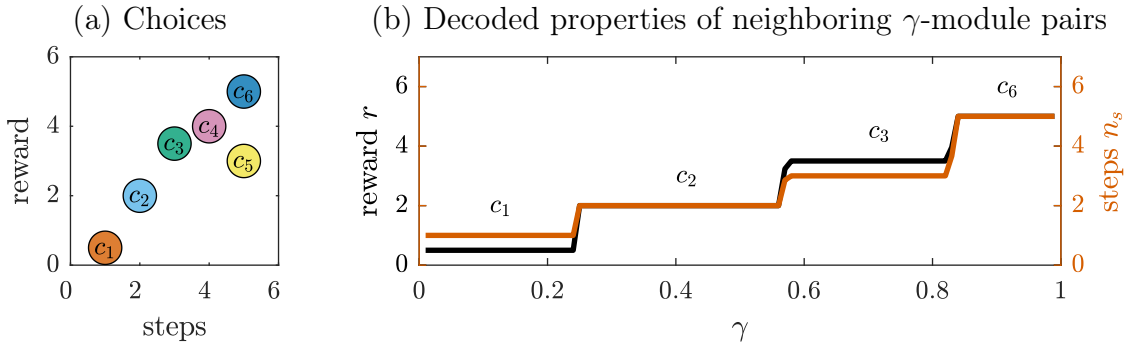


Figure 2.8: The reward and the number of steps can be correctly decoded for each learned choice in the MDP from Fig. 2.4. The modules that encode c_1, c_2, c_3, and c_6 can clearly be identified using the decoded information. Each choice is represented by a region where the number of steps n_s and the reward r are constant.

The number of steps n_s can be used with the value of one of the modules to compute r (Eq. 2.10) either by:

$$V_{\gamma_a}(s) = {\gamma_a}^{n_s-1} r \quad \Leftrightarrow \quad r = \frac{V_{\gamma_a}(s)}{{\gamma_a}^{n_s-1}} \quad , \quad \text{or} \quad V_{\gamma_b}(s) = {\gamma_b}^{n_s-1} r \quad \Leftrightarrow \quad r = \frac{V_{\gamma_b}(s)}{{\gamma_b}^{n_s-1}} \; .$$

□

Theorem 8 can be used to decode the expected reward r and number of steps n_s of two modules that have the same choice as outcomes. It can also be used to identify modules that have the same choices. In deterministic, goal-only-reward MDPs, a choice is guaranteed to be learned within a certain γ-region ($\gamma_L < \gamma < \gamma_U$) that has a lower and an upper bound (Theorem 3). Having a set of γ-modules $\Gamma = (\gamma_1, \ldots, \gamma_M)$ with $\gamma_i < \gamma_{i+1}$ the modules that have the same choice can be detected, by decoding n_s from each neighboring pair: $n_s(\gamma_i, \gamma_{i+1})$ (Fig. 2.8). If at least three neighboring modules exist in a γ-region for a choice c: $\gamma_L(c) < \gamma_a < \gamma_b < \gamma_c < \gamma_U(c)$, then the number of steps has to be equal for their neighboring pairs: $n_s(\gamma_a, \gamma_b) = n_s(\gamma_b, \gamma_c)$. The modules of all neighboring pairs with the same number of steps learn the same choice, because two choices cannot have the same number of steps. Otherwise, one would dominate the other (Theorem 4). The pair of modules on the border of the γ-regions, where the modules learn different choices can be detected. If γ_a and γ_b learn one choice and γ_c and γ_d another, then $n_s(\gamma_a, \gamma_b) \neq n_s(\gamma_b, \gamma_c) \neq n_s(\gamma_c, \gamma_d)$.

In summary, although decoding of the expected reward trajectory has practical limitations for general MDPs, it is possible in deterministic, goal-only-reward MDPs. Theorem 8 allows to identify γ-modules that learned the same trajectory and to decode their expected reward and number of steps. This information can help the agent to select the most appropriate policy for the task it wants solve in an environment.

2.4 Optimal Distribution of γ-Modules

The number of modules that an IGE can use is finite. This raises the question as to which discount factors should be used for the modules, i.e. how the discount factors should be

distributed. In many cases the goal is to learn a set of policies that differ in order to provide a diverse behavioral repertoire for the agent. The mapping from discount factors to the policies they learn depends strongly on the structure of target MDPs, i.e. their transition probability functions and reward functions. Thus, an optimal distribution of discount factors cannot be given for MDPs in general. As a general rule, discount factors should be spread out between 0 and 1 with enough space between them, because similar discount factors have similar values and often similar policies.

This section analyzes the distribution of discount factors for episodic MDPs with several goal states, the main application scenario for the IGE. The class of deterministic, goal-only-reward MDPs is used to analyze these cases (Definition 3). Their restrictions on the MDP structure allow to analyze the optimal distribution of discount factors. These MDPs have a set of choices that represent the optimal trajectories to each possible goal state. The question is how the γ-modules should be distributed to allow that all possible choices can be learned.

The analysis is accomplished by assuming a set of choices and then by identifying which discount factors would be necessary to learn all of these choices. A choice set consists of all choices $c_j = (r_j, n_j)$ starting at a certain number of steps m, where for each subsequent step j a choice exists: $n_j = m + j$. The rewards for each choice depend on its position in the set $r_j = f(j)$. Fig. 2.9 (a) shows an example where rewards depend linearly on the steps. The goal is to detect the γ-regions that are required to learn each choice. Having a module with a discount factor within each of the regions guarantees that

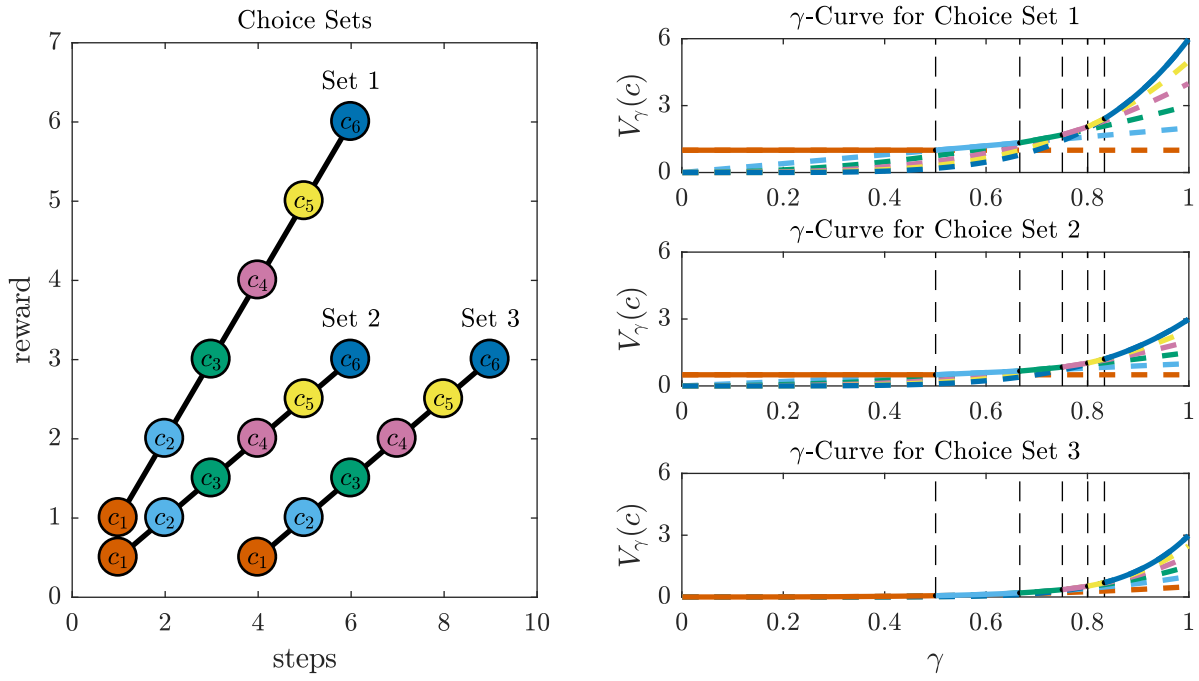


Figure 2.9: The mapping of γ-modules to choices is invariant to scaling in rewards and translation in steps of choice sets. (left) Three different choice sets, each with 6 choices. The rewards of Set 1 are double than Set 2 ($r_{\text{Set1}} = 2 \cdot r_{\text{Set2}}$). The steps of Set 3 are increased by two from Set 2 ($n_{\text{Set3}} = 2 + n_{\text{Set2}}$). (right) Nonetheless, the mapping between γ-modules and their preferred choice is the same for each choice set.

all choices will be learned.

The γ-regions depend on the relationship between reward and position $r_j = f(j)$. Theorem 9 determines the regions in the case of an exponential and linear relationship with $r_j = b \cdot j^a$, where b is a scaling factor and a determines the exponential influence. The relation is linear for $a = 1$.

Theorem 9. *For any deterministic, goal-only-reward MDP with choices that are defined by:*

$$r(j) = b \cdot j^a \ , \quad n(j) = m + j \ ,$$

where $m \in \mathbb{N}$, $b \in \mathbb{R}^+$, and $a \in R^+$, is the γ-region Γ_j for which all its discount factors γ_j encode the j'th choice $c_j = (r(j), n(j))$ defined by:

$$\Gamma_j = \left\{ \gamma_L(j) = \left(\frac{j-1}{j}\right)^a < \gamma_j < \gamma_U(j) = \left(\frac{j}{j+1}\right)^a \right\} \ , \tag{2.11}$$

where $\gamma_L(j)$ and $\gamma_U(j)$ are the lower and upper border of the region.

Proof. The γ region for which choice c_j is the outcome is defined by Theorem 3:

$$\gamma_L(j) = \max_{\forall c_i < c_j} \gamma_E(c_i, c_j) < \gamma_j < \gamma_U(j) = \min_{\forall c_k > c_j} \gamma_E(c_j, c_k) \ .$$

The lower border $\gamma_L(j)$ can be shown to be equal to $\left(\frac{j-1}{j}\right)^a$, starting with:

$$\begin{aligned}
\gamma_L(j) &= \max_{\forall c_i < c_j} \gamma_E(c_i, c_j) \\
&= \max_i \left(\frac{r(i)}{r(j)}\right)^{\frac{1}{n(j)-n(i)}} \\
&= \max_i \left(\frac{b \cdot i^a}{b \cdot j^a}\right)^{\frac{1}{(m+j)-(m+i)}} \\
&= \max_i \left(\frac{i}{j}\right)^{\frac{a}{j-i}} \qquad \text{s.t. } 0 < i < j \ .
\end{aligned}$$

The maximum value of this term is given by Lemma 9.1.

Lemma 9.1. *It holds that*

$$\max_i \left(\frac{i}{j}\right)^{\frac{a}{j-i}} = \left(\frac{j-1}{j}\right)^a \ , \tag{2.12}$$

for $a \in \mathbb{R}^+$, $i \in \mathbb{N}^+$, $j \in \mathbb{N}^+$, and $i < j$.

Proof. The maximum of Eq. 2.12 is at $i = j - 1$, because the derivative of $\left(\frac{i}{j}\right)^{\frac{a}{j-i}}$ is positive for all $i > 0$ and it is the largest possible i. Using it for Eq. 2.12 results in:

$$\left(\frac{j-1}{j}\right)^{\frac{a}{j-(j-1)}} = \left(\frac{j-1}{j}\right)^a \ ,$$

proving Lemma 9.1. The proof depends on the assumption that the derivative of Eq. 2.12 is positive for $0 < i < j$:

$$\frac{\partial}{\partial i}\left(\frac{i}{j}\right)^{\frac{a}{j-i}} = a\left(\frac{i}{j}\right)^{\frac{a}{j-i}} \left(\frac{i \log\left(\frac{i}{j}\right) + j - i}{i(j-i)^2}\right) \ . \tag{2.13}$$

The derivative (Eq. 2.13) is positive if:

$$i \log\left(\frac{i}{j}\right) + j - i > 0 \qquad \text{s.t. } i < j \ . \tag{2.14}$$

This can be shown using the derivative of this term with respect to i:

$$\frac{\partial}{\partial i}\, i \log\left(\frac{i}{j}\right) + j - i \ = \ \log\left(\frac{i}{j}\right).$$

This term is negative for all $0 < i < j$. Moreover, Eq. 2.14 is 0 for the point $i = j$:

$$j \log\left(\frac{j}{j}\right) + j - j = 0 \ .$$

Thus, Eq. 2.14 has to be positive for all $0 < i < j$, showing that the derivative Eq. 2.13 is positive for $0 < i < j$. This ends the proof of Lemma 9.1. □

The proof of the upper border $\gamma_U(j)$ follows similar steps:

$$\begin{aligned}
\gamma_U(j) &= \min_{\forall c_k > c_i} \gamma_E(c_j, c_k) \\
&= \min_k \left(\frac{r(j)}{r(k)}\right)^{\frac{1}{n(k)-n(j)}} \\
&= \min_k \left(\frac{b \cdot j^a}{b \cdot k^a}\right)^{\frac{1}{(m+k)-(m+j)}} \\
&= \min_k \left(\frac{j}{k}\right)^{\frac{a}{k-j}} \qquad\qquad \text{s.t. } 0 < j < k \ .
\end{aligned}$$

The minimum of this term is given by Lemma 9.2 which finishes the proof of the upper border.

Lemma 9.2. *It holds that*

$$\min_k \left(\frac{j}{k}\right)^{\frac{a}{k-j}} = \left(\frac{j}{j+1}\right)^a \ , \tag{2.15}$$

for $a \in \mathbb{R}^+$, $j \in \mathbb{N}^+$, $k \in \mathbb{N}^+$, and $j < k$.

Proof. The minimum of Eq. 2.15 is at $k = j + 1$, because the the derivative of $\left(\frac{j}{k}\right)^{\frac{a}{k-j}}$ is positive for all $k > j$ and it is the smallest possible k. Using it for Eq. 2.15 results in:

$$\left(\frac{j}{j+1}\right)^{\frac{a}{(j+1)-j}} = \left(\frac{j}{j+1}\right)^a \ ,$$

proving Lemma 9.2. The proof depends on the assumption that the derivative of Eq. 2.15 is positive for $k > j$:

$$\frac{\partial}{\partial k}\left(\frac{j}{k}\right)^{\frac{a}{k-j}} = -a\left(\frac{j}{k}\right)^{\frac{a}{k-j}} \left(\frac{k \log\left(\frac{j}{k}\right) + k - j}{k(k-j)^2}\right) \ . \tag{2.16}$$

The derivative (Eq. 2.16) is positive if:

$$k \log\left(\frac{j}{k}\right) + k - j < 0 \qquad \text{s.t. } j < k \ . \tag{2.17}$$

This can be shown using the derivative of this term with respect to k:

$$\frac{\partial}{\partial k}\, k \log\left(\frac{j}{k}\right) + k - j \;=\; \log\left(\frac{j}{k}\right) \;.$$

This term is negative for all $0 < j < k$. Moreover, Eq. 2.17 is 0 for the point $k = j$:

$$j \log\left(\frac{j}{j}\right) + j - j = 0 \;.$$

Thus, Eq. 2.17 has to be negative for all $k > j$, showing that the derivative Eq. 2.16 is positive for $k > j$. This ends the proof of Lemma 9.2. □

Having verified the lower and upper border ends the proof of Theorem 9. □

The γ-regions are independent of scaling in the reward function by the coefficient b and translations of the number of steps by m (Fig. 2.9). The regions only depend on the position of a choice in the set.

A similar relationship can also be shown for choice sets that follow a log distribution with:

$$r(j) = b \cdot \log(a) \;, \quad n(j) = m + j \;.$$

In this case the upper and lower border of the j'th choice are given by:

$$\Gamma_j = \left\{ \frac{\log(j-1)}{\log(j)} < \gamma_j < \frac{\log(j)}{\log(j+1)} \right\} \;. \tag{2.18}$$

For both cases, the exponential choice sets (Eq. 2.11) and the logarithmic choice sets (Eq. 2.18), is the upper γ-border for one choice the lower γ-border for the next:

$$\gamma_U(j) = \gamma_L(j+1) = \gamma_E(c_j, c_{j+1}) \;.$$

This means that for a set of N choices, N distinct γ-regions exist. To guarantee that every choice in a set can be learned, it is necessary that at least one module exists for each region.

The exact locations of the borders between regions are different for each choice set, but choices with more steps are learned by larger discount factors (Fig. 2.10). Furthermore, the size of the γ-region decreases with the number of steps ($\gamma_U(j) - \gamma_L(j) > \gamma_U(j+1) - \gamma_L(j+1)$). As a consequence, modules need to be more densely distributed for larger γ's close to 1.

In summary, the choice of discount factors of IGE modules, depends on the type of MDP and the types of policies that should be learned. The analysis of MDPs with several goal states shows that the distribution of discount factors should not be uniform. Instead, modules should be more densely distributed for larger discount factors, thereby allowing the IGE to learn policies that also reach goal states that need more steps to be reached.

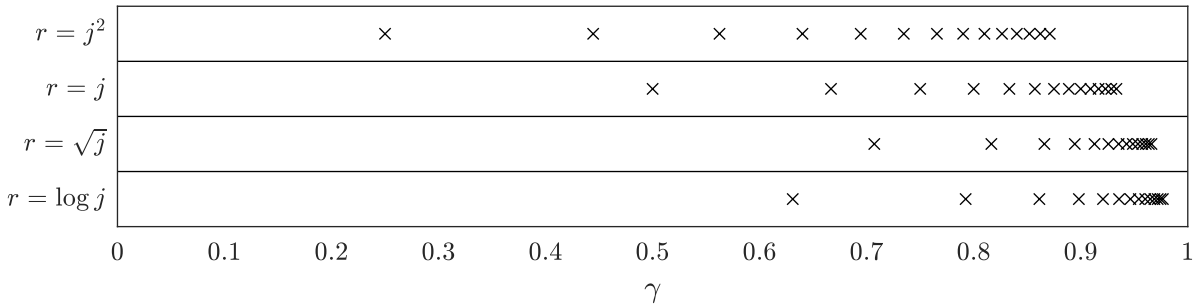


Figure 2.10: The boundaries $\gamma_E(c_j, c_{j+1})$ between the γ-regions of choices for different sets. Each set has a different relationship between the rewards r and the position of the choice j in the set. Each set has 15 choices: $C = (c_1, \ldots, c_{15})$. The boundaries show that modules should be denser distributed for larger γ's near 1 to cover each region with a module.

2.5 Effect of the Discount Factor on the Exploration

The discount factor of the active module $\tilde{\gamma}$ that the IGE uses can influence the exploration behavior of the agent. Usually, an agent explores the environment by using a suboptimal action for some transitions ($a_t \neq \text{argmax}_a Q_\gamma(s_t, a)$). The exploration strategy, i.e. the probability of different actions in a state, depends upon the values of the actions. Because discount factors have a strong influence on values, they can also influence exploration behavior.

The example MDP in Fig. 2.11 illustrates the influence of discount factors on exploration. The MDP has three different paths that can be reached from the initial state s_0. Each path is the optimal solution for a different γ-region (Fig. 2.11, b). Depending on the action selection strategy used, such as softmax or ϵ-greedy, the exploration behavior is influenced differently by the discount factor.

For the softmax action selection, the dependency between action probabilities and γ is more complex (Fig. 2.11, c). For γ-regions where a certain path is optimal, the action to reach it is the highest (for $\beta > 0$). Nonetheless, the probabilities between different actions change, depending on the γ in the γ-region. For example, the second path to g_2 is optimal for $0.34 < \gamma < 0.76$. The highest probability to choose action a_2 for this path is for $\gamma = 0.63$. Toward the upper and lower bounds of the γ-region, the probability of its use is reduced.

A second effect of the softmax action selection is that for larger discount factors, choices with more steps, but also higher rewards have a higher probability of being explored than shorter choices with smaller rewards. In the example for $\gamma = 0.9$, the probabilities are $a_1 = 0.01$, $a_2 = 0.19$, and $a_3 = 0.8$, showing that the shortest path (a_1) has a very low probability of being explored. This effect could be used to focus exploration on longer or shorter solutions in an MDP.

Moreover, the difference between probabilities for choosing each path is reduced for smaller γ's. For $\gamma = 0.1$, the probabilities are $a_1 = 0.39$, $a_2 = 0.32$, and $a_3 = 0.29$, showing that all paths have a high probability of being explored. The probability differences are stronger for larger γ's. Therefore, with a constant exploration factor β, exploration is stronger for small γ's than for larger ones. To allow weaker exploration for smaller γ's,

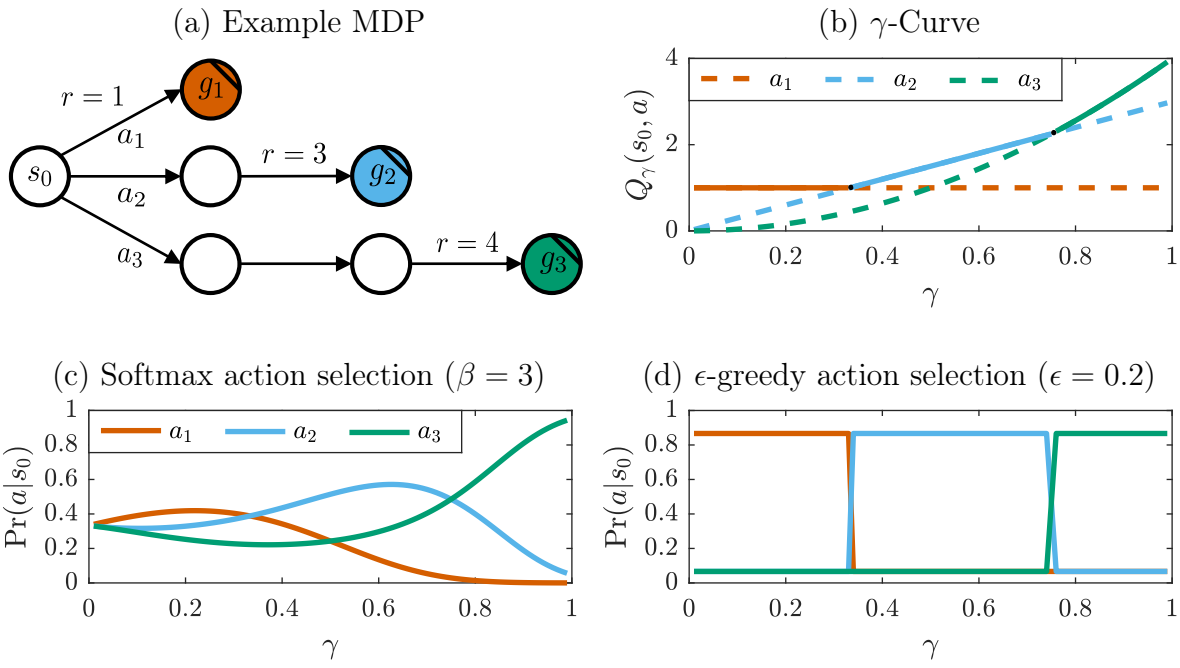


Figure 2.11: The example MDP shows the effect of the discount factor on the exploration of the agent. (a) The MDP has 3 different paths. (b) Each path is the optimal policy for a different region of discount factors. (c) The exploration behavior of the softmax action selection depends on the discount factor, whereas (d) for the ϵ-greedy action selection it only depends on the policy that is optimal in each γ-region.

the exploration factor β must be increased. As a result, the exploration factor should depend upon the discount factor: $\beta(\gamma)$.

The influence of the discount factor on exploration can be eliminated with an ϵ-greedy action selection. ϵ-greedy selects a random action with probability ϵ at each transition; otherwise it selects the action with the maximum value. Probabilities are therefore independent of individual value differences between actions. Only which action is optimal influences the probabilities (Fig. 2.11, d).

In summary, depending on the used approach, the choice of the discount factor has strong implications for the exploration behavior. For softmax, the exploration rate is strongly dependent upon the active γ-module. In contrast, the ϵ-greedy action selection has no such dependence. ϵ-greedy is therefore used for all application scenarios of the IGE in later chapters to avoid complications resulting from the choice of the active module and the agent's exploration behavior.

2.6 Conclusion

The IGE framework consists of independent modules, each learning a Q-function with a different discount factor. This allows the framework to learn a set of policies. The policies differ in their temporal preferences for expected future reward. Modules with small γ factors optimize rewards on a short-term time scale, whereas modules with larger γ's also consider the reward sum in the long-term. Depending on the structure of the

MDP, the policies represent different behaviors and possible strategies that an agent can use in the MDP. One type of environment where this ability is useful, is for MDPs with several goal states. The IGE learns policies in these environments to reach different goal states. Small γ's prefer proximal goals, whereas large γ's prefer more distant goals with greater rewards.

The IGE also allows to decode information about its learned policies. In theory, it can decode the full expected reward trajectory for policies that are learned by several modules. Unfortunately, this approach has practical limitations. Nonetheless, it is applicable for deterministic, goal-only-reward MDPs. This ability can be used to identify the most appropriate policy for a certain task the agent has to fulfill in an environment.

The study of deterministic, goal-only-reward MDPs also offers insight into the question of which discount factor parameters should be used for the modules. In MDPs with several goal states, it is beneficial to have modules more densely distributed for larger discount factors, because the γ-regions that learn policies to reach distant goals are denser distributed as γ approaches 1.

The IGE also affects the exploration behavior of the agent. Depending on the action selection method, exploration can vary greatly within a γ-region, even if the optimal policy for this region is the same. The ϵ-greedy action selection strategy is recommended in order to avoid this effect.

The following chapters introduce some application scenarios for the IGE and compare them to classical non-modular methods. Chapter 3 addresses tasks that change contextually. In such cases, the IGE provides a set of policies that can be used by the agent to adapt to the different contexts. In the Chapter 4, the agent has to adapt to different objectives. The agent can immediately adapt to a changed objective using one of the policies learned by the IGE. Moreover, the decoding ability of the IGE allows it to identify the most appropriate policy for a goal. Finally, Chapter 5 demonstrates the ability of the IGE to optimize average reward. In this case it can be shown that the IGE is guaranteed to learn and identify the optimal average reward policy in deterministic, goal-only-reward MDPs.

Chapter 3

Adaptation to Contextually Changing Tasks

The IGE has the ability to learn a set of behaviors for a task (Section 2.2). This can be an advantage for tasks that change depending on some context. The IGE can choose the most appropriate behavior for a given task context from its behavioral set, allowing the IGE to quickly adapt. In contrast, classical methods need to independently learn for each task context the appropriate behavior.

For example, in an energy foraging task an agent has to collect energy from different sources. Each source provides a different amount of energy and requires a different number of steps to reach. Sources that are far away provide more energy than nearby sources. The agent has an energy store and some backup energy. For each step that the agent performs its energy store reduces by a small amount. The goal of the agent is to collect the energy source that gives the highest energy payoff. The payoff is the gained energy from a source minus the energy it used to get there. Furthermore, the agent should avoid to deplete its energy store. If it reaches an energy level of zero, it can still move because of its backup energy, but the agent should learn to avoid this situation. The optimal energy source depends on the agent's energy level, which is the context of the energy foraging task. Having enough energy, the agent can reach distant energy sources that give a high payoff without reaching an energy level of zero. Whereas, if its energy level is low, going to a distant source can deplete its energy level and it would need to use backup energy which should be avoided. Thus, under low energy levels nearby energy sources are optimal. The task can be formalized as an MDP (Fig. 3.1). The agent receives positive rewards for reaching an energy source which are the goal states in the MDP. The strength of the rewards signal how much energy it collected from a source. For each action the agent receives a negative reward representing the energy that the action cost. If the energy level of the agent's energy store is 0, the agent receives for any further action a high negative reward.

The IGE can learn for the energy foraging task policies to reach different energy sources (Fig. 3.1, b). γ-Modules with high discount factors γ learn to reach distant sources, because they provide a high payoff. Modules with low γ's will prefer nearby sources, because they require fewer steps to be reached. Depending on the context, i.e. the agent's energy level, the IGE can select the most appropriate policy from its modules. The selection is done by a γ-map that defines which γ-module to use under which context.

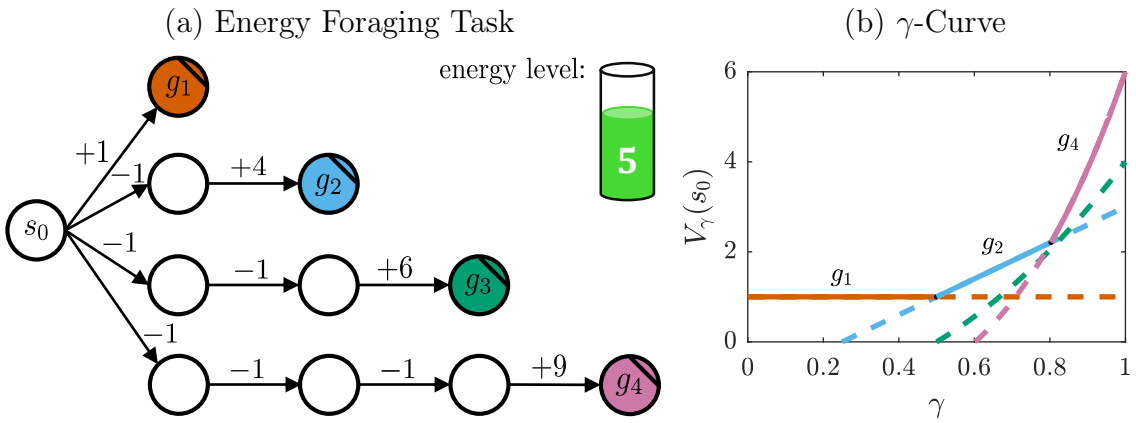


Figure 3.1: (a) An energy foraging task with 4 energy sources. The agent has an energy level of 5. For each step the agent loses energy. It gains energy for reaching a source. If the energy level reaches 0 the agent will receive a large negative reward per action. (b) The γ-Curve for the start state s_0 and each possible policy shows that the IGE can learn 3 (reaching g_1, g_2 or g_4) of the 4 possible strategies.

For the foraging task, the agent should use modules with large γ's to go to distant sources, if it has a high energy level. For low energy levels, it should use modules with small γ's to go to nearby sources. The IGE can outperform classical methods in such tasks, because it learns the different behaviors in parallel. This allows the IGE to choose quickly between the behaviors under different contexts. Classical methods need to independently learn the optimal behavior for each context.

This chapter provides several variants of tasks with varying contexts and variants of the IGE algorithm to solve them. Before the different scenarios and algorithms are introduced, the concept of a Context Adaptive MDP (CMDP) is given. CMDPs are used to define all tasks throughout this chapter. The general framework of IGE agents to solve CMDPs is introduced in Section 3.2. Afterward, the first application scenarios are discussed in Section 3.3. These are specific tasks for which a γ-mapping from context to the optimal γ-module can be theoretically derived. In Section 3.4, this is followed by a discussion about the application of the IGE to the general CMDPs for which the γ-mapping needs to be learned.

3.1 Context Adaptive Markov Decision Processes

The context of a task can be understood as a task modifier. It modifies the task and makes it necessary for an agent to adapt its behavior to solve the task optimally. For the purpose of the tasks and algorithms in this chapter, the context is known by the agent. A task with contexts is modeled by a regular MDP in which the context is represented as an extra state dimension.

Definition 5. A Context Adaptive Markov Decision Process (CMDP) is an MDP where the state space S is composed of environment states S^E and context states S^C:

$$\text{CMDP}\Big(A, S = \left(S^E \times S^C\right), T, R\Big) \ ,$$

where A is a finite set of actions, $T : S \times A \times S \mapsto [0, 1]$ is a transition probability function and $R : S \times A \times S \mapsto \mathbb{R}$ is a reward function.

In a CMDP the task space is defined by the environmental states S^E. The task changes between the contexts $s^C \in S^C$. The goal is to maximize the total expected reward sum per episode:

$$\max_{\pi} \mathrm{E}_{\pi,s_0} \left[\sum_{t=0}^{T-1} R\Big(\left(s_t^E, s_t^C\right), a_t, \left(s_{t+1}^E, s_{t+1}^C\right) \Big) \right] . \tag{3.1}$$

Combining the environment and the context space of a task into the state space of an MDP allows the context to alter a task in three ways. Either the reward function changes between contexts (Fig. 3.2, top), or the transition function changes (Fig. 3.2, bottom), or through a combination of both. The definition of a CMDP allows drastic changes of a task between contexts. In particular, changes in the transition probabilities can change a task so that it is hard to perceive a CMDP as one task under different contexts. For example, for a CMDP with two context states, the transition probability between the contexts can change in such a manner that under one context only a subset of environmental states can be reached. Whereas, under the other task context a non-overlapping, different subset of environmental states can only be reached. Thus, each context has a non-overlapping subset of states. In this case, policies learned under one context cannot be used for the other. In order to allow a transfer of knowledge between the contexts of a CMDP, the reward function and transition probabilities should be similar between contexts. For the CMDPs that are discussed as application scenarios for the IGE, the changes of the reward function and the transition probabilities between contexts are restricted in ways to allow the IGE to exploit their similarities.

The concept of context in reinforcement learning has been explored in the past. In the field of Multi-Armed Bandits (Sutton & Barto 1998) exists the concept of Contextual

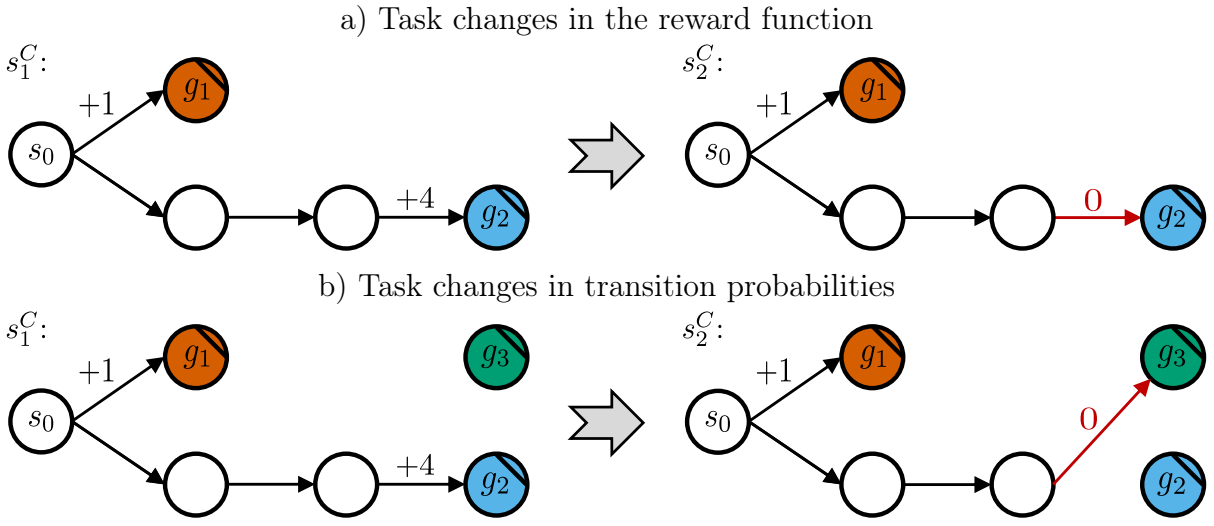


Figure 3.2: Tasks can change between different contexts (s_1^C and s_2^C) in a CMDP due to changes in a) the reward function R or b) transition probabilities T. Task changes can alter the optimal policy between contexts, such as from taking the long choice under s_1^C to the short choice under s_2^C.

Multi-Armed Bandits (CMABs) (Langford & Zhang 2008; Zhou 2015). CMABs have one state and several arms or actions. Each action has a stochastic reward function. The context of a bandit influences the reward distributions of the arms. This notion of context is similar between CMABs and the CMDP in this work. The difference is that bandits are not multi-step decision problems. Actions have no influence on the context state of a bandit in the next episode. Whereas, in CMDPs the task itself is a multi-step decision problem and actions can have an influence on the context state.

In the area of multi-step decision making a Contextual Markov Decision Process (Contextual MDP) exists (Hallak, Di Castro, & Mannor 2015). The Contextual MDP is defined as a set of separate MDPs. All of them have the same action and state space, but different transition dynamics and reward functions. A context state, which is usually hidden, defines in which MDPs the agent is for an episode. The main goal for Contextual MDPs is to identify the latent context state from interactions with the environment, so that the correct policy for the MDP can be applied. The CMDP differs in two ways to the Contextual MDP. First, the context is given to the agent. The goal is not to identify in which context the agent is, but to adapt quickly to a new context and to transfer knowledge between contexts. Second, the context is a state dimension and might change during an episode. In Contextual MDPs, the context does not change during an episode. In this regard, Contextual MDPs are a generalization of the CMABs (Langford & Zhang 2008; Zhou 2015).

The term context is also used to define Contextual Decision Processes (CDPs) (Jiang, Krishnamurthy, et al. 2017), but it is used in a very different manner compared to CMDPs. CDPs use the notion of contexts to describe an abstraction of the state space. The abstraction allows the description of regular MDPs and POMDPs under one framework. A context is either a normal state space in the case of MDPs or complete histories over observations in POMDPs.

3.2 The Context Adaptive Independent γ-Ensemble

Classical reinforcement learning approaches, such as Q-learning, can solve CMDPS, because CMDPs are just regular MDPs with an extra context dimension. The disadvantage of classical methods is that they need to learn over the full state space ($S^E \times S^C$) to identify the correct policy for each context. Q-learning, for example, needs to learn values for all combinations of environment, and context states, and actions:

$$Q : S^E \times S^C \times A \mapsto \mathbb{R} \ .$$

This results in a high demand for observations to learn the Q-function. The problem becomes especially apparent for problems with large task and context state spaces.

The next section introduces a variant of the IGE, the Context Adaptive Independent γ-Ensemble (CA-IGE), that can be used to overcome this problem by reducing the complexity of the learning problem. Afterward, the conditions that a task should fulfill, so that the CA-IGE can be successfully applied, are discussed in Section 3.2.2.

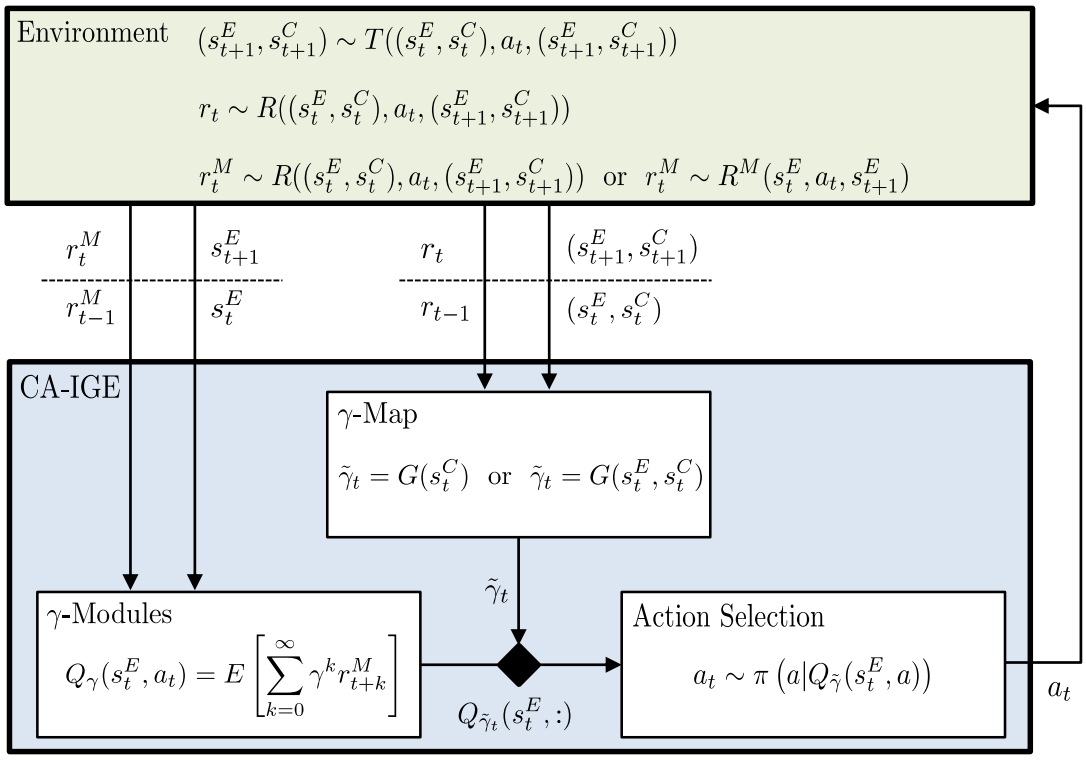


Figure 3.3: The general framework of the Context Adaptive IGE (CA-IGE). The Q-values of the γ-modules are only learned for environmental states S^E. Contexts s^C or environment-context state pairs (s^E, s^C) are used to select which module is active and used for the action selection.

3.2.1 The CA-IGE Framework

The CA-IGE (Fig. 3.3) learns different policies by its γ-modules only over the environmental state space S^E:

$$Q_\gamma : S^E \times A \mapsto \mathbb{R} \ ,$$

and uses the most appropriate policy for a certain context. Not having to learn the Q-values over the context dimensions reduces the complexity of the learning problem for the modules. A γ-map G provides a mapping from contexts or environment-context state pairs to the most appropriate module for a context:

$$G : S^C \mapsto \Gamma \quad \text{or} \quad G : S^E \times S^C \mapsto \Gamma \ ,$$

where Γ is the set of discount factors for which the CA-IGE has γ-modules. The CA-IGE selects an active module $\tilde{\gamma}$ either for each transition or only at the beginning of an episode. The Q-function of the selected module $\tilde{\gamma}$ is used for the action selection. For certain CMDPs the γ-map can be theoretically derived (Section 3.3), so that it guarantees to choose the optimal γ-module for any given context. For general CMDPs the map has to be learned (Section 3.4).

For most CMDPs in this chapter, the γ-Modules do not learn their values based on the reward function R of the CMDP. Instead, a separate reward function R^M is used. It

is either defined over environment states or environment-context state pairs:

$$R^M : S^E \mapsto \mathbb{R} \quad \text{or} \quad R^M : S^E \times S^C \mapsto \mathbb{R} \ .$$

The differentiation between the reward function R and the γ-module reward function R^M is necessary for many CMDPs, to allow the modules to learn a consistent set of policies over all contexts. CMDPs allow drastic changes in their reward function between contexts. In such cases, if modules learn their Q-functions based on this reward function, their optimal values Q^*_γ and policies change between contexts. This leads to an unlearning, and relearning of values if a context changes. To avoid this problem, the reward function R^M should be chosen to be consistent over contexts. How to choose R^M is further discussed in Section 3.2.2.

An advantage of the CA-IGE to classical reinforcement learning algorithms is its reduction of the complexity of the learning problem. It allows a faster learning with fewer observations. For CMDPs where the γ-map is theoretically derived, the reduction of the learning complexity is most apparent. The CA-IGE learns the Q-functions for its modules only over the environmental state space S^E. In contrast, classical methods such as Q-learning have to learn the Q-function over the full state space $S^E \times S^C$.

The task complexity can also be reduced for CMDPs where the γ-map is learned. The γ-modules learn different policies, i.e. the behavior of the agent in each environmental state s^E, for the task. To learn the γ-map not every individual action for each environment-context state pair needs to be learned. Instead, only which policy should be used. The reduction of the learning complexity by a CA-IGE can increase the learning speed significantly compared to classical methods, especially in tasks that have large state spaces.

The idea behind the CA-IGE is similar to the concept of transfer learning (Section 1.2). The source task is a certain context under which the agent is learning different policies. These policies are then transferred as knowledge to improve the learning in the target, a different task context.

3.2.2 Application Scenarios and Optimality

A disadvantage of the CA-IGE is that for many CMDPs its convergence towards the optimal policy cannot be guaranteed. Classical algorithms, such as Q-learning, can guarantee to converge to the optimal policy, because CMDPs are regular MDPs. Q-learning is proven to converge to optimal policy for MDPs (Watkins & Dayan 1992; Tsitsiklis 1994). Nonetheless, the reduction of the learning complexity can make the CA-IGE more efficient compared to classical algorithms. This section discusses the properties that a CMDPs should have so that the CA-IGE can be successfully applied. First, the conditions under which a CA-IGE can guarantee to optimally solve a CMPD are discussed. For some CMDPs these condition can be fulfilled, but not for all. Nonetheless, the conditions provide some insight into the properties of tasks that do not fulfill the conditions but for which the CA-IGE can be successfully applied.

The optimal policy for a CMDP in state (s^E, s^C) is defined as $\pi^*_{s^C}(s^E)$. The CA-IGE learns the Q-functions of its modules over the task space S^E. The optimal policy that a γ-modules learns for a certain task state s^E is defined as $\pi^*_\gamma(s^E)$. To choose an action for

a certain state (s^E, s^C), the CA-IGE selects a module according to the map $\tilde{\gamma} = G(s^C)$ or $\tilde{\gamma} = G(s^E, s^C)$. The action is then chosen according to the Q-function of the selected module. To guarantee that the CA-IGE can choose the optimal policy $\pi^*_{s^C}(s^E)$ for all states, one of the γ-modules must have the same optimal policy:

$$\forall s^E \in S^E, s^C \in S^C, \;\; \exists \gamma \in \Gamma : \;\; \pi^*_{s^C}(s^E) = \pi^*_{\gamma}(s^E) \; . \tag{3.2}$$

If this is the case, an optimal γ-mapping G^* can be either derived or learned.

The optimality criteria (Eq. 3.2) gives two conditions which should be fulfilled. First, the modules should converge to an optimal policy $\pi^*_{\gamma}(s^E)$. Second, for each optimal policy in the CMDP, a γ-module should exist that has the same policy. Based on these conditions, the properties of CMDPs to which the CA-IGE can be successfully applied can be defined.

Task Conditions allowing the Learning of Consistent Policies

Considering the first condition, i.e. that the modules converge to their optimal policy, the reward function used for the modules and the transition probabilities over the task space S^E should be invariant to context changes. For the γ-modules, the context space S^C is a hidden dimension. If the γ-modules learn their values from observation of the reward function R and the transitions function T of the CMDP, their optimal values and associated policies are defined as the expectation over the context states:

$$Q^*_{\gamma}(s^E, a) = \mathrm{E}_{s^C}\left[R\left(s^E, s^C, a\right) + \gamma \max_{a'} Q^*_{\gamma}\left(s^{E\prime}, s^{C\prime}, a'\right)\right] \; .$$

Unfortunately, learning the optimal values is difficult and often not possible for general CMDPs. For general CMDPs, the transition probabilities T and the reward function R can change drastically between contexts (Fig. 3.2). As a result, the observations $(s^E_t, r^M_t, s^E_{t+1})$, used to learn the Q-values of γ-modules, have a high variance over the contexts. The high variance makes it difficult for the γ-modules to converge to their optimal values. As a consequence, the transition probabilities of a CMDP between task states s^E should not vary between contexts. Moreover, the CMDP should allow to define a reward function R^M used by the γ-modules, that is also invariant to context changes. This excludes the variance of the observations over contexts for the γ-modules, and makes it possible for the modules to converge to their optimal policies.

One way to achieve that transition probabilities over task states s^E are equal between contexts is to separated the transition probability function T in two components:

$$T^E : S^E \times A \times S^E \mapsto [0, 1] \; , \text{ and } \; T^C : (S^E \times S^C) \times A \times S^C \mapsto [0, 1] \; .$$

The component T^E only defines the transition probabilities between task states S^E. The transitions are not influenced by context. The component T^C defines the transition probabilities between contexts. The transitions can still depend on the task states S^E. The foraging task (Fig. 3.1) includes such a separation. The agent can perform each transition in the task space S^E regardless of the context, its energy level. Nonetheless, other methods exist besides the separation of the transition function to ensure that the transition probabilities of the task space S^E are equal under each context. For example, in

interruption risk tasks (Section 3.3.1), the learning algorithm of the modules is altered in such a way that the transition probabilities only seem to be equal under all contexts for the γ-modules.

Besides the transition probabilities, the reward function R^M, used for the learning of the γ-modules, should not vary between contexts. One solution is to learn the values of modules only under the reward function R for a subset of contexts $\Phi \subset S^C$:

$$R^M(s^E, a) = R(s^E, \phi \in \Phi, a) \ ,$$

where the reward function over all contexts in Φ is equal:

$$\forall \phi_i, \phi_j \in \Phi : \quad R(s^E, \phi_i, a) = R(s^E, \phi_j, a) \ .$$

Observations from other contexts $s^C \notin \Phi$ are ignored by the modules. For the energy foraging task (Fig. 3.1) the reward function R^M can be defined as the reward function R while the agent has energy ($\forall s^C > 0 : \quad R^M(s^E, a) = R(s^E, s^C, a)$). The reward function R does not vary for positive energy levels. It only changes when the agent depleted its energy ($s^C = 0$). Then a strong punishment is added for each further action.

Having a reward function R^M that does not vary allows for the values of the γ-modules to converge. Additionally, it guarantees that the optimal policy $\pi^*_{s^C}$ for the contexts Φ will be learned by modules with $\gamma \approx 1$. A γ-mapping with $\forall \phi \in \Phi : \ G(\phi) \approx 1$ guarantees that the optimal policy is then used for the contexts Φ. For other contexts, optimality cannot be guaranteed. However, if the correct contexts Φ are chosen, the IGE can outperform classical methods through an increased learning speed.

A drawback of learning the γ-modules only under certain contexts Φ is that observations under other contexts ($s^C \notin \Phi$) are not used to learn the modules Q-values. The agent will need more time to learn, reducing its performance. Nonetheless, the CA-IGE variants that use this approach can still outperform classical algorithms (Section 3.4.3). Moreover, for some tasks it is possible to reformulate the learning algorithm (Section 3.3.1), or the reward function (Sections 3.3.2, 3.3.3 and 3.4.3) to allow observations under all contexts to be used to update the γ-modules.

Task Conditions allowing the Learning of Helpful Policies

The second condition of the optimality criteria for the CA-IGE (Eq. 3.2) is that for each optimal policy in the CMDP, a γ-module should exist that learns this optimal policy. For a subset of CMDPs it is possible to guarantee this condition (Section 3.3), but for most CMDPs, the CA-IGE cannot learn the optimal policy for each context. The CA-IGE can only use policies that are learned by the γ-modules. Some optimal policies for certain contexts of a CMDP are not learned, because they are not optimal for any γ-module. For example, the energy foraging task (Fig. 3.1) has four possible energy sources the agent can reach. Each is optimal for a certain energy level. Nonetheless, the γ-Modules learn only three of the four options (Fig. 3.1, b). As a result, the agent cannot choose the optimal policy for each energy level. Classical algorithms such as Q-learning are guaranteed to converge to the optimal policy. However, if the CA-IGE learns a relatively good set of policies, then its advantage of reducing the learning complexity of a tasks can result in an advantage over classical algorithms. At the moment, the theoretical conditions for

CMDPs so that the CA-IGE can have such an advantage are missing. Some informal constraints can be given by analyzing the properties of the IGE.

The γ-modules of the IGE learn different policies based on their difference in the discounting of reward over time. Modules with large γ's prefer larger rewards, which might take longer to reach. Modules with smaller γ's prefer less distant rewards, but which are smaller. Thus, the CA-IGE should be used in tasks where a trade-off between the reward strength and the time to reach rewards exist. Depending on the context, either longer strategies or shorter strategies should be optimal. The energy foraging task (Fig. 3.1) is an example for such tasks. It has four strategies with different time requirements to reach reward. Strategies that need longer provide more reward. This results in a trade-off between reward strength and the time to reach it. If the agent has enough energy, the longest strategy gives the highest payoff. For lower energy levels, shorter strategies are optimal. In such environments, the CA-IGE can learn the different strategies by different γ-modules and it can then use them under different contexts.

That the CA-IGE is most helpful for tasks with a trade-off between reward strength and required time gives also some insight into the choice of the reward function R^M for the γ-modules. Usually, a reward function is chosen which uses the rewards of certain contexts $\Phi \subset S^C$. A context should be chosen that allows the CA-IGE to learn many different strategies considering the trade-off between reward and time. Given a certain context, the IGE will not only learn the strategy with the highest payoff, but also other strategies that give less reward in less time. Therefore, the modules should learn under a context that has longer strategies as optimal solutions. For the foraging task, these are the contexts under which the agent has energy. If the agent has energy, it can reach every possible reward without receiving any punishment. As a result, it can learn not only the longest strategy with the highest payoff, but also other strategies which are helpful under reduced energy levels.

Conclusion

In summary, the CA-IGE can provide advantages over classical algorithms in CMDPs which allow the γ-modules to learn a set of policies that are helpful under different contexts. To allow the modules to learn a consistent set of polices over all contexts, the transition probabilities between task states S^E should not depend on the context. Moreover, the reward function R^M used for the modules should also not vary between contexts. In order for the CA-IGE to learn policies that are helpful, the task should have a trade-off between the reward value of different strategies and the time needed to reach the reward. In such tasks, the γ-modules can learn different policies which represent the solution to different trade-offs between reward and time.

The next two sections introduce different CMDPs and variants of the CA-IGE algorithm to solve them. First, CMDPs for which the optimal γ-map can be theoretically derived are discussed in Section 3.3. Afterward, CMDPs for which a γ-map needs to be learned are introduced in Section 3.3.

3.3 Derivation of γ-Mappings

For some types of CMDPs a direct mapping from context to the optimal γ-module can be theoretically derived. This section introduces three types of CMDPs with direct γ-mappings. The first type are Interruption Risk MDPs. They have at each step a risk that the task gets interrupted. The context determines the probability of these interruptions. The second type, Reward Risk MDPs, are similar. For them exists a risk that all future reward becomes zero. In the third class of environments, Constant Punishment MDPs, a constant punishment per step is given. In this case, the context is the strength of the punishment.

3.3.1 Interruption Risk Tasks

During interruption risk tasks a constant risk at each time point exists that the task will be interrupted and ends (Fig. 3.4). The interruption risk is constant per episode, but it can change between episodes. The probability of an interruption is the context of the task. The agent has to adapt to different risk probabilities. An optimal γ-map from context to the module with the optimal policy for it can be derived for such tasks.

For example, an agent might perform a base task, such as cleaning for a house keeping robot or collecting energy for a space exploration probe. It has to abort this task if another task is given to it. For the base task, different possible strategies might exist that the agent can learn. Some strategies have a low time effort, but also give only a low payoff. Whereas, other strategies need a lot of time before a high payoff can be achieved. Every time the agent receives a task, it has to end its base task and perform the assigned task. Depending on factors, such as the time of the day, the probability that a task gets assigned might differ. If there is a high risk of a task assignment, the agent should prefer strategies for its base task that have a low time effort to reduce the risk of being interrupted before finishing. Whereas, if there is a low risk of a task assignment, the agent can perform a long term strategy which gives a high payoff. The risk of a task assignment is the context under which the agent performs its base task. It could be either given to the agent or learned, for example by observing how often a task gets assigned during certain hours of the day.

This section first introduces the concept of Interruption Risk MDPs (IR-MDP) which are used to formalize interruption risk tasks. Next, the Interruption Adaptive IGE (IR-

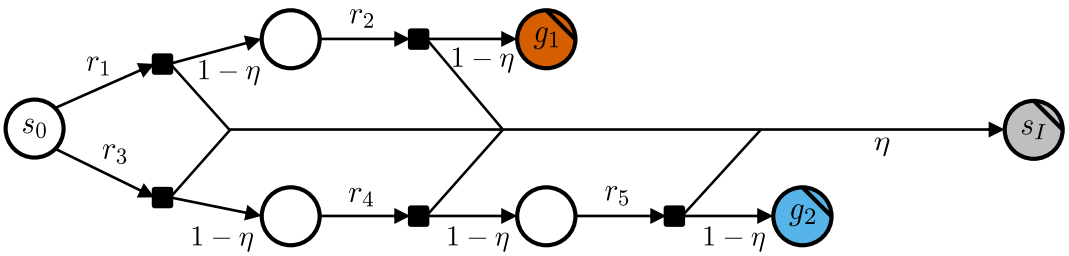


Figure 3.4: Illustration of an interruption risk MDP with 2 choices (g_1, g_2). At each step a risk with the probability η exists that the agent is interrupted and ends in the terminal state s_I.

IGE) is introduced which is a CA-IGE variant to solve IR-MDPs. For the IR-IGE a direct mapping from the interruption risk to the optimal γ-module, which should be used for this risk, can be constructed. The IR-IGE is guaranteed to solve IR-MDPs optimally. Afterward, the IR-IGE is compared to classical Q-learning in a IR-MDP which shows that it can outperform Q-learning.

Interruption Risk MDPs and the Interruption Risk IGE

Definition 6. An Interruption Risk MDP (IR-MDP) is a CMDP$(A, S^E \times S^C, T, R)$ that has an interruption state $s^I \in S^E$ as part of its environmental state space. At each state transition a probability of $\eta \in [0, 1]$ exists that the agent will be interrupted and transitions into the interruption state s_I:

$$\forall s \in S, a \in A: \quad T(s, a, s^I) = \eta \ ,$$

where η is constant during an episode. The context during each episode is the interruption risk:

$$s^C = \eta \ .$$

IR-MDPs can be optimally solved with an Interruption Risk IGE (IR-IGE) (Algorithm 3.1). The γ-modules of the IR-IGE learn their values based on the reward function of the IR-MDP:

$$R^M(s, a, s') = R(s, a, s') \ .$$

The learning algorithm for each module is modified compared to the standard Q-learning update. After transitions in which the agent gets interrupted and enters the interruption state s^I, the Q-value for the state from which the agent came is not updated. Thus, the agent learns the Q-functions as if there is no interruption risk, i.e. with risk $\eta = 0$. Using the mapping $\gamma = 1 - \eta$ from context η to γ-modules with such Q-functions guarantees that the agent selects for each context the optimal policy (Theorem 10).

Theorem 10. *An IR-IGE with a γ-map of*

$$G^*(\eta) = 1 - \eta \ ,$$

has the optimal policy $\pi^*_{s^C}(s^E)$ for all states $(s^C, s^E) \in S$ of an IR-MPD, under the assumption that the γ-modules converged to their optimal Q-functions Q^*_γ and use their greedy policy.

Proof. The goal of the IR-IGE is to optimize the expected reward sum per episode starting in a state s_0 and following the optimal policy π^*:

$$\mathrm{E}_{\pi^*,s_0}\left[\sum_{t=0}^{T-1} r_t\right] = \mathrm{E}_{\pi^*,s_0}[r_0] + \mathrm{E}_{\pi^*,s_0}[r_1] + \mathrm{E}_{\pi^*,s_0}[r_2] + \cdots + \mathrm{E}_{\pi^*,s_0}[r_{T-1}] \tag{3.3}$$

The expected reward for a time step t, in the case that no interruption risk exists ($\eta = 0$), is defined as $\mathrm{E}^{\eta=0}_{\pi^*,s_0}[r_t]$. The expected reward at t under a certain risk η is either 0 if the agent was interrupted, or it is the expected reward if there is no interruption $\mathrm{E}^{\eta=0}_{\pi^*,s_0}[r_t]$:

$$\begin{aligned}
\mathrm{E}_{\pi,s_0}[r_t] &= Pr(s_t = s_I|t) \cdot 0 + Pr(s_t \neq s_I|t) \cdot \mathrm{E}^{\eta=0}_{\pi^*,s_0}[r_t] \\
&= Pr(s_t \neq s_I|t) \cdot \mathrm{E}^{\eta=0}_{\pi^*,s_0}[r_t] \\
&= (1-\eta)^t \cdot \mathrm{E}^{\eta=0}_{\pi^*,s_0}[r_t] \ ,
\end{aligned}$$

Algorithm 3.1: Interruption Risk IGE

Input:
Learning rate: $\alpha \in [0, 1]$
Sorted list of discount factors: $\Gamma = (\gamma_1, \ldots, \gamma_m)$ with $\gamma_i \in (0, 1)$

initialize $Q_\gamma(s, a)$ to zero
repeat (for each episode)
  initialize state s, context η
  `// map context to the active module` γ^*
  $\gamma^* \leftarrow 1 - \eta$
  repeat (for each step in episode)
    choose an action a for s derived from Q_{γ^*} (e.g. ϵ-greedy)
    take action a, observe r, s'
    `// only update Q-values if the algorithm does not end in the interruption state`
    if $s' \neq s_I$ **then**
      forall the $\gamma \in \Gamma$ **do**
        $Q_\gamma(s, a) \leftarrow Q_\gamma(s, a) + \alpha \left(R(s, a) + \gamma \max_{a'} Q_\gamma(s', a') - Q_\gamma(s, a)\right)$
      end
    end
    $s \leftarrow s'$
  until *s is terminal-state*
until *termination*

where the probability of not being interrupted is $Pr(s_t \neq s_I | t) = (1 - \eta)^t$ (See Fig. 3.4). Therefore, the expected reward sum (Eq. 3.3) of the optimal policy ϕ^* for each context η can be reformulated to:

$$\mathrm{E}_{\pi^*, s_0}\left[\sum_{t=0}^{T-1} r_t\right] = \sum_{t=0}^{T-1} (1 - \eta)^t \cdot \mathrm{E}^{\eta=0}_{\pi^*, s_0}[r_t] \ . \tag{3.4}$$

Because the modules of an IR-IGE learn their Q-functions as if no interruption risk exist, there optimal Q-values are:

$$Q^*_\gamma(s_0, a) = \sum_{t=0}^{T-1} \gamma^t \cdot \mathrm{E}^{\eta=0}_{\pi^*, s_0}[r_t] \ , \tag{3.5}$$

Using a mapping of $\gamma = 1 - \eta$, the optimal Q-values (Eq. 3.5) and the expected reward sum of the optimal policies for each context (Eq. 3.4) are equal. Therefore, after the Q-functions converged to their optimal values, using a module with $\gamma = 1 - \eta$ under context η will result in the optimal policy for the IR-IGE. This proves that the IR-IGE is optimal for all interruption risks η using the γ-map $G^*(\eta) = 1 - \eta$. □

Experiments

The IR-IGE was compared to a standard Q-learning algorithm in a grid world variant of an IR-MDP (Fig. 3.5, a). The grid world has five goal states which provide a different amount of reward (Fig. 3.5, b). At each time step a risk of η exists that the task is interrupted and the agent transitions into the interruption state s^I. The expected reward for reaching each goal state with an optimal policy depends on η (Fig. 3.5, c). For different risks η are different goal states optimal. The environmental states S^E of the IR-MDP are the positions in the grid world. The context states S^C are the risk levels η.

Each experimental run consisted of four phases with 5000 episodes per phase. The phases differed in their risk level η. Phase 1 had a small risk ($\eta = 0.03$), followed by Phase 2 that had a medium risk ($\eta = 0.15$). Phase 3 had no risk ($\eta = 0$) and Phase 4 had an high risk ($\eta = 0.3$). Fig. 3.5 (c) shows that the optimal choice changed between each phase with c_5 for Phase 1, c_2 for Phase 2, c_5 for Phase 3 and c_4 for Phase 4. The experiment tested if and how fast the IR-IGE could adapt to changes in a risk level compared to traditional Q-learning.

Both algorithms used a learning rate of $\alpha = 0.1$ and an exponentially decaying ϵ-greedy action selection with $\epsilon_{init} = 50$, $d = 0.998$ and $\epsilon_{min} = 0$. For the IR-IGE, ϵ stayed at zero after the first phase. For Q-learning, ϵ was reset after each phase to allow it to explore and update its Q-values to the new risk level. The IR-IGE did not need to update its Q-values after phase transitions, because it can generalize policies learned under the first risk context for all other contexts.

The results of Phase 1 show that the IR-IGE and Q-learning behaved similar (Fig. 3.6). Both learned the optimal choice with the same number of episodes. In the successive phases, the IR-IGE outperformed Q-learning. The IR-IGE could immediately adapt to a new risk level by using the policy of a different module learned during the first phase.

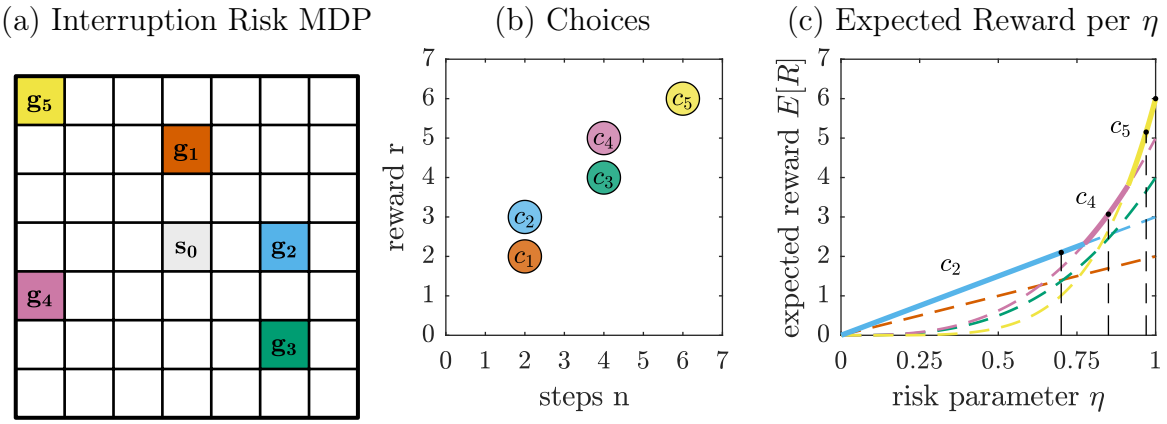


Figure 3.5: (a) The Interruption Risk MDP used for the experiments consisted of one start state s_0 and 5 terminal states $(g_1, \ldots, g_5)$. (b) Each terminal state represents a different choice with different rewards and a number of minimum steps to reach them. (c) The expected reward of a choice depends on the risk η that the task gets interrupted. Choice c_2 has the highest expected reward for a high risk. For intermediate and low risks, choices c_4 and c_5 are optimal. The black points and dashed lines show the risk levels for the different experimental phases.

Q-learning needed to learn a new policy for each phase.

The IR-MDP used for the experiments (Fig. 3.5) is deterministic in its state transitions over the grid world positions. The only stochastic element is the interruption risk. Thus, for the γ-modules of the IR-IGE, the task is deterministic, because Q-values are only updated if there is no interruption. As a result, the learning rate of the IR-IGE can be set to $\alpha = 1.0$ and the exploration time can be reduced to $d = 0.985$.

The learning rate in MDPs with a deterministic transition probability and reward function can be set to $\alpha = 1$, because no variance over the observation of state transitions and rewards exists. The Q-values are the expectation over the future reward sum. In stochastic environments the value has to be updated in small increments over several steps with $\alpha < 1$ to allow the value to converge over iterations to the true expectation. In deterministic environments, the value can be immediately updated with $\alpha = 1$, because their exists no variance and the observed rewards and state transitions in each step are equal to the expectation.

The higher learning rate and reduced exploration allowed the IR-IGE to learn the task faster (Fig. 3.6). Q-learning could benefit from the higher α and lower exploration only

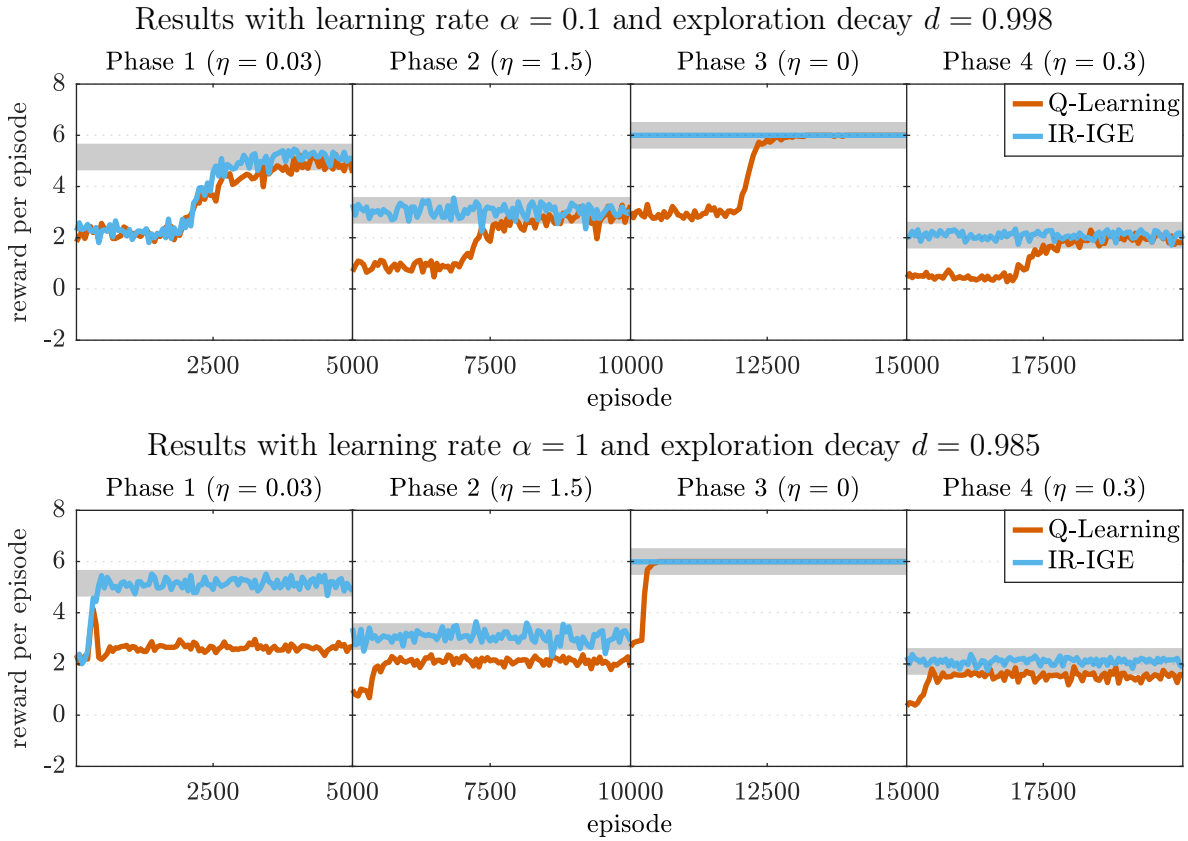


Figure 3.6: In contrast to Q-learning, the IR-IGE adapted immediately to changing risk levels in Interruption Risk MDPs. The algorithms were compared on the IR-MDP from Fig. 3.5 with four different risk levels (η). IR-IGE could immediately adapt to the optimal policy for each risk level after Phase 1 as shown by the reward sum per episode. The reward sum is the mean over 100 agents per algorithm. The optimal reward per phase is indicated by the gray line. Q-learning needed to adapt to each risk level.

in Phase 3 which had no risk. In phases which included risk, Q-learning could not learn the optimal policy, because its values could not converge under the high α setting.

The IR-IGE outperformed classical Q-learning in the experiments. IR-IGE immediately adapted to different interruption risks. Moreover, it allowed to increase the learning rate and to reduce exploration, because it did not suffer from the increased stochasticity that the interruption risk adds to a task.

Discussion

The IR-IGE is guaranteed to be optimal for tasks with an interruption risk for each time step. For the theoretical analysis and the experiments the interruption risk η is given as context to the agent. For many tasks, this interruption risk may not be known. Nonetheless, it can be learned. For example, in tasks where the risk depends on the hour of the day h, the interruption risk $\eta(h)$ for each hour can be learned. Only the frequency of interruptions per time step needs to be observed:

$$\eta(h) = \frac{\text{number of interruptions during hour } h}{\text{number of state transition during hour } h} \ .$$

This makes the application of the IR-IGE to many scenarios with interruption risk possible.

One scenario is tasks where the agent has to control machines which have different failure rates because of their different age. If a machine fails, the task the agent performs is interrupted. The IR-IGE can adapt immediately between different machines. Another scenario is where the agent has to interrupt its task due to an incoming requests to perform a different task as described in the introduction to this section. Depending on external factors this interruption risk can vary. In summary, the IR-IGE can be applied to a multitude of task scenarios in which an agent needs to adapt its strategy to perform a task because the task might get interrupted.

It should be noted that IR-MDPs can also be solved by another modular Q-learning approach. Similar to the IR-IGE, it has an η-module for each potential interruption risk η. All modules learn from the observations (s_t, a_t, r_t, s_{t+1}) during an episode. Each module's discount factor is near one ($\gamma \approx 1$). Each module samples at each transition if an interruption will take place according to its associated risk η. If it sampled that an interruption will happen, the module pretends that the episode ended in the interruption state and updates the last Q-value with a reward of 0: $Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha(0 - Q(s_t, a_t))$. If the observation from the environment ends in the interruption state, then, like the IR-IGE, the Q-values of the last previous state are not updated. By this method, each η-module learns its Q-values as if under a interruption risk of η. Depending on the current interruption risk of the task, the associated module's Q-function is used to define the policy of the agent. Although, this framework can solve IR-MDPs, it has a disadvantage to the IR-IGE. For all η-modules with $\eta > 0$ the task has an increased stochasticity due to their interruption risk. As a result, the learning rate α should be low enough to allow the convergence of the Q-values. This is not the case for the IR-IGE. For the modules of the IR-IGE, the interruption risk does not increase the stochasticity over the observation from which they learn. Thus, a larger learning rate α can be used improving the learning speed. This gives the IR-IGE an advantage over the the η-module approach.

3.3.2 Reward Risk Tasks

Reward risk tasks are another task variant for which an optimal mapping from context to the optimal γ-module can be derived. These tasks are similar to the interruption risk tasks (Section 3.3.1). Instead of a risk for task interruption, reward risk tasks have a risk that all future reward becomes zero without the knowledge of the agent (Fig. 3.7). More formally, for each time step t exists a risk of η that the reward for all future time steps is zero. Similar to the interruption risk tasks, the risk probability is the context of the task. The tasks are formulated by Reward Risk MDPs (RR-MDPs) which have an observable state space and a hidden state space. The observable space is the task without any risk that future reward becomes zero. Any MDP can be used for it. The hidden state space is used to define if the current and all future reward is zero.

A variant of the CA-IGE, the Reward Risk Adaptive IGE (RR-IGE), solves RR-MDPs optimally for different risk levels. The important property of the RR-IGE is that even in the case that all future reward became zero, it still learns as if it would still receive rewards. This can be understood as if the agent knows what reward it would have received. This allows the γ-modules to learn a consistent set of policies over all risk levels and that an optimal map from the reward risk η to the optimal γ-module can be constructed with $\gamma = 1 - \eta$.

For example, in a foraging task the agent learns to find the way to different food sites. The food sites have different distances and reward values. Distant sites offer more food. Furthermore, the agent has to compete with other agents that it cannot observe. Other agents could get to a food site and eat everything before the agent can reach the site. If the agent reaches a food source that has already been taken it gets no reward. Nonetheless, the agent could still get information about the potential reward it could have received if another agent hadn't reached the site first. Remains of the food such as peels could provide such information. From this information the agent can still learn that this food site has a high payoff, as long as no other agent reaches it first. This risk that a food site has already been taken by another agent can be modeled as a probability per time step. This probability could depend, for example, on the number of agents in the environment. If there are more agents, then there is a higher risk that another agent already reached

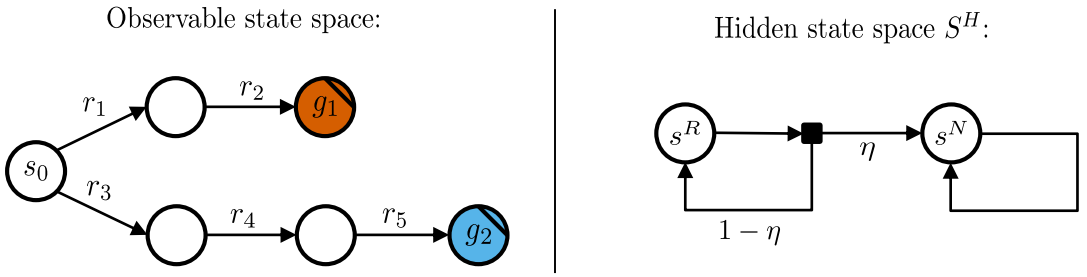


Figure 3.7: Illustration of a Reward Risk MDP with 2 choices (g_1, g_2). The states are separated in two spaces. The observable state space can be any MDP. The hidden state space S^H consists of 2 states which determine if the agent receives reward (s^R) or not (s^N). For each step a risk with the probability η exists that the hidden state changes from s^R to s^N and that the agent will not receive further rewards.

a food site. Depending on this risk level, the agent has to adjust its choice which food source it wants to reach.

The next part defines the RR-MDPs used to formalize reward risk tasks and the RR-IGE to solve them. It also provides the proof that the RR-IGE is optimal for RR-MDPs. The RR-IGE is then compared to classical Q-learning in a grid world RR-MDP, showing that the RR-IGE outperforms classical Q-learning.

Reward Risk MDPs and the Reward Risk IGE

Definition 7. A Reward Risk MDP RR-MDP$(A, S^E \times S^C, T, S^H, T^H, R)$ is a partially observable variant of a CMDP. It has an observable environmental and context state space $S^E \times S^C$ with a transition probability function $T : S^E \times A \times S^E$. The hidden part of the RR-MDP consists of two states $S^H = (s^R, s^N)$ and a transition probability function $T^H : S^H \times A \times S^H$ between them. At the beginning of an episode the agent starts in the hidden state s^R. At each transition a risk of η exists with which the hidden state changes to s^N:

$$\forall a \in A : \; T^H(s^R, a, s^N) = \eta \; ,$$

with η being constant during an episode. The hidden state s^N is absorbing. The reward function R depends on the hidden state in which an agent ends after a transition:

$$R\left((s_t^E, s_t^H), a_t, (s_{t=1}^E, s_{t+1}^H)\right) = \begin{cases} R^O(s_t^E, a_t, s_{t=1}^E) & | \; s_t^H = s^R \\ 0 & | \; s_t^H = s^N \end{cases} \; ,$$

where $R^O : S^E \times A \times S^E \mapsto \mathbb{R}$ is a reward function over the observable environmental state space S^E. The context during each episode is defined by the risk probability:

$$s^C = \eta \; .$$

RR-MDPs can be optimally solved by a Reward Risk Adaptive IGEs (RR-IGE) (Algorithm 3.2). The RR-IGE is a variant of the CA-IGE. Its γ-modules learn the reward function based on the observable reward function R^O:

$$R^M(s_t^E, a_t, s_{t+1}^E) = R^O(s_t^E, a_t, s_{t+1}^E) \; . \tag{3.6}$$

This allows the modules to learn their values as if no risk $\eta = 0$ exists in the RR-MDP. It can be proven that this allows the RR-IGE to learn the optimal policy by using the mapping $\gamma = 1 - \eta$ from context η to γ-modules.

Theorem 11. *An RR-IGE with a γ-map of*

$$G^*(\eta) = 1 - \eta \; ,$$

has the optimal policy $\pi^*_{s^C}(s^E)$ for all states $(s^C, s^E) \in S$ of an RR-MPD, under the assumption that the γ-modules converged to their optimal Q-functions Q^*_γ and use their greedy policy.

Proof. The goal of the IR-IGE is to optimize the expected reward sum per episode if starting in a state s_0 and following the optimal policy π^*:

$$\mathrm{E}_{\pi^*,s_0}\left[\sum_{t=0}^{T-1} r_t\right] = \mathrm{E}_{\pi^*,s_0}[r_0] + \mathrm{E}_{\pi^*,s_0}[r_1] + \mathrm{E}_{\pi^*,s_0}[r_2] + \ldots + \mathrm{E}_{\pi^*,s_0}[r_{T-1}]$$

The expected reward for a time step t, in the case that no reward risk exists ($\eta = 0$), is defined as $\mathrm{E}^{\eta=0}_{\pi^*,s_0}[r_t]$. The expected reward at t under a certain risk η is either 0 if the hidden state s^N is reached, or it is the expected reward as if there is no risk $\mathrm{E}^{\eta=0}_{\pi^*,s_0}[r_t]$:

$$\begin{aligned}\mathrm{E}_{\pi,s_0}[r_t] &= Pr(s_t = s_I|t) \cdot 0 + Pr(s_t \neq s_I|t) \cdot \mathrm{E}^{\eta=0}_{\pi^*,s_0}[r_t] \\ &= Pr(s_t \neq s_I|t) \cdot \mathrm{E}^{\eta=0}_{\pi^*,s_0}[r_t] \\ &= (1-\eta)^t \cdot \mathrm{E}^{\eta=0}_{\pi^*,s_0}[r_t] \ ,\end{aligned}$$

where the probability of not reaching the hidden state s^I is $Pr(s_t \neq s_I|t) = (1-\eta)^t$ (See Fig. 3.7). Therefore, the expected reward sum (Eq. 3.6) of the optimal policy ϕ^* for each context η can be reformulated to:

$$\mathrm{E}_{\pi^*,s_0}\left[\sum_{t=0}^{T-1} r_t\right] = \sum_{t=0}^{T-1}(1-\eta)^t \cdot \mathrm{E}^{\eta=0}_{\pi^*,s_0}[r_t] \ . \tag{3.7}$$

Algorithm 3.2: Reward Risk IGE

Input:
Learning rate: $\alpha \in [0, 1]$
Sorted list of discount factors: $\Gamma = (\gamma_1, \ldots, \gamma_m)$ with $\gamma_i \in (0, 1)$

initialize $Q_\gamma(s, a)$ to zero
repeat (for each episode)
  initialize state s, context η
  `// map context to the active module` γ^*
  $\gamma^* \leftarrow 1 - \eta$
  repeat (for each step in episode)
    choose an action a for s derived from Q_{γ^*} (e.g. ϵ-greedy)
    take action a, observe r^M, s'
    forall the $\gamma \in \Gamma$ **do**
      `// learn values based on reward function` R^M
      $Q_\gamma(s,a) \leftarrow Q_\gamma(s,a) + \alpha\left(r^M(s,a) + \gamma \max_{a'} Q_\gamma(s',a') - Q_\gamma(s,a)\right)$
    end
    $s \leftarrow s'$
  until s *is terminal-state*
until *termination*

Because the modules of an IR-IGE learn their Q-functions from reward function R^O as if no reward risk exist, their optimal Q-values are:

$$Q^*_\gamma(s_0, a) = \sum_{t=0}^{T-1} \gamma^t \cdot \mathrm{E}^{\eta=0}_{\pi^*, s_0}[r_t] \ , \tag{3.8}$$

Using a mapping of $\gamma = 1 - \eta$, the optimal Q-values (Eq. 3.8) and the expected reward sum of the optimal policies for each context (Eq. 3.7) are equal. Therefore, after the Q-functions have converged to their optimal values, using a module with $\gamma = 1 - \eta$ under context η will result in the optimal policy for the RR-IGE. This proves that the RR-IGE is optimal for all interruption risks η using the γ-map $G^*(\eta) = 1 - \eta$. □

Experiments

The RR-IGE was compared to Q-learning in a grid world variant of an RR-MDP (Fig. 3.8). The grid world has the same layout as the grid world of the interruption risk experiments (Fig. 3.5). For each step a risk of η exists that future reward becomes zero. Each experimental run was divided in four phases with different risk levels (Phase 1: $\eta = 0.03$, Phase 2: $\eta = 0.15$, Phase 3: $\eta = 0$, Phase 4: $\eta = 0.3$). Each phase consisted of 5000 episodes.

The RR-IGE has been compared to two Q-learning approaches. The Q-learners differ in their state space information. The state space of the first approach consists of the position p of the agent in the grid world and the reward risk ($s = (p, \eta)$). The second approach has a step counter t as state information ($s = (t, p, \eta)$) as well. The counter t holds the number of steps that the agent performed since the start of the episode. This allows the Q-learner to incorporate information about the changed expectation of the future reward if it visits the same state at different time points. This is important

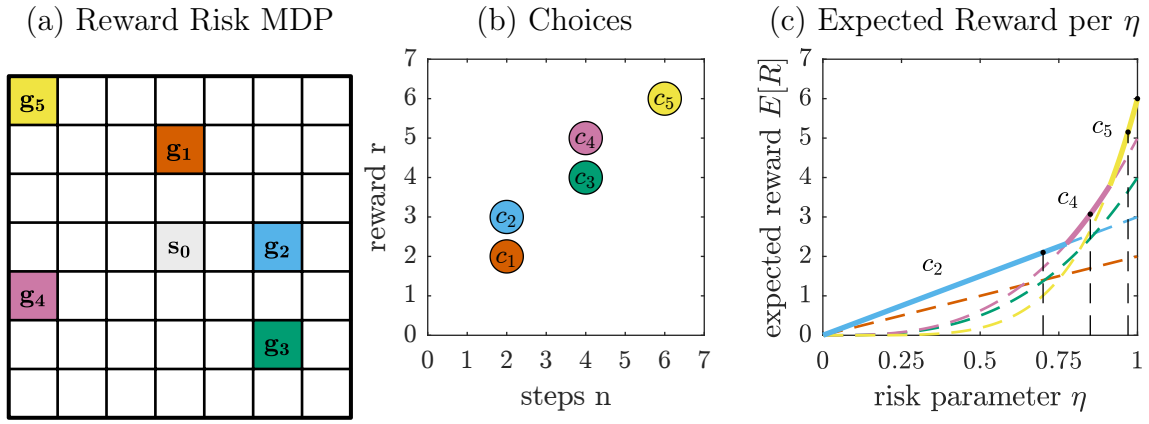


Figure 3.8: (a) The reward risk MDP used for the experiments consisted of one start state s_0 and 5 terminal states $(g_1, \ldots, g_5)$. (b) Each terminal state represents a different choice with different rewards and a number of minimum steps to reach them. (c) The expected reward of a choice depends on the risk $(1 - \eta)$ at each step that future reward becomes zero. For a high risk (low η) choice c_2 gives the highest expected reward. Whereas for intermediate and low risk choices c_4 and c_5 are optimal. The black points and dashed lines show the risk levels of the different experimental phases.

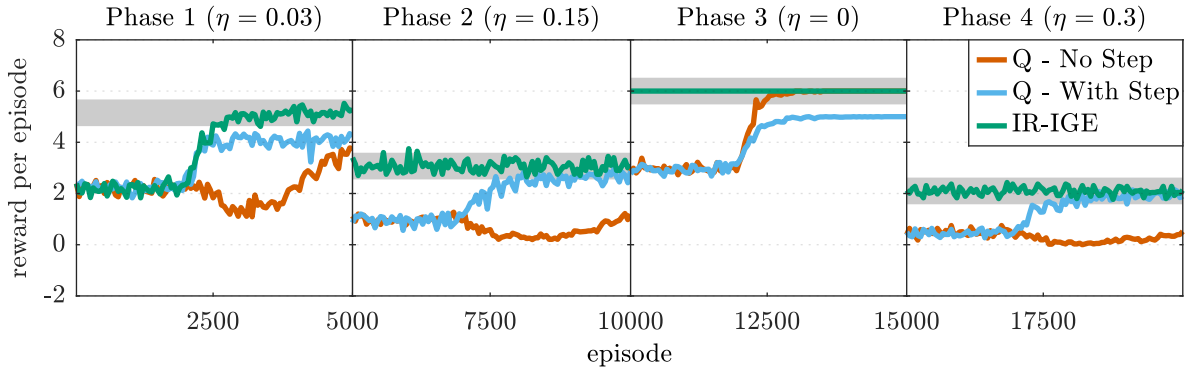
because the probability that the agent gets no more reward, i.e. that hidden state s^N was entered, is higher when the agent visits the same state at a later time point again. For example, the agent starts in state s_0 in Fig. 3.8. In the first run, the agent goes to goal state g_1 by the trajectory (up, up). In the next run, the agent goes also to goal state g_1, but by the trajectory (left, up, right, up). Both trajectories have the state north of s_0 as last state before entering the goal state g_1. Nonetheless, the number of steps before entering this state differ between these two trajectories and therefore the risk that the reward for entering the terminal state is zero is different. As a result, information about the time is needed to differentiate the two situations.

For the RR-IGE, the position in the grid world is used as the environmental states S^E for the γ-modules. The reward risk level η is used as context information S^C.

All agents used a learning rate of $\alpha = 0.1$ and an exponentially decaying ϵ-greedy action selection with $\epsilon_{init} = 50$, $d = 0.998$ and $\epsilon_{min} = 0$. For the RR-IGE ϵ stayed at zero after the first phase. For both Q-learners ϵ was reset after each phase to allow them to explore and update their Q-values to the new risk level.

The results show that the RR-IGE outperformed both Q-learning approaches in all

(a) Results with learning rate $\alpha = 0.1$ and exploration decay $d = 0.998$



(b) Results with learning rate $\alpha = 1$ and exploration decay $d = 0.985$

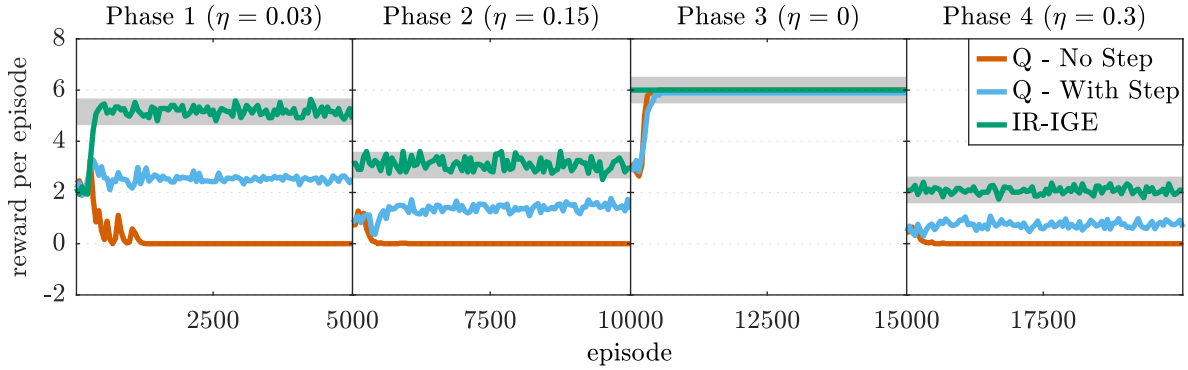


Figure 3.9: In contrast to Q-learning without and with time step information, the RR-IGE adapts immediately to changing risk levels in RR-MDPs and can utilize a higher learn rate of $\alpha = 1$. The algorithms were compared in a grid world task (Fig. 3.8) with four different risk levels (η). The mean over 100 runs per algorithm are shown. The optimal reward per phase is indicated by the gray region.

phases (Fig. 3.9). After Phase 1, the RR-IGE could immediately adapt to different risk levels. Both Q-learning approaches needed to learn a new Q-function for each risk level. The Q-learner without information about the current time step could not find the optimal policy for any phase with risk (Phase 1, 2, and 4). The agent could learn the optimal policy only in Phase 3 that has no risk. Q-learning with step information can learn the optimal policy for all phases. Nonetheless, in Phase 1 and 3 which have the farthest goal state g_5 as optimal solution, the agent could not learn the optimal policy to reach it for each of the 100 experimental runs. The problem is its increased state space as a result of using the time step information. The increased state space would require more exploration to consistently find the optimal policy.

For the γ-modules of the RR-IGE the task is deterministic because the rewards given to them do not include the risk. As a result, the learning rate of the RR-IGE can be set to $\alpha = 1.0$ and the exploration time can be reduced be setting the exploration decay to $d = 0.985$. This allowed the RR-IGE to learn the optimal policy for each of its modules faster (Fig. 3.9, b, Phase 1). Both Q-learning methods had a decreased performance in phases with risk (Phases 1,2 and 4) for these learning parameters. The task is stochastic for them under risk and it would require a smaller α to learn it. Only in Phase 3, that has no risk, could the Q-learning approaches improve their performance. Nonetheless, they are still outperformed by the RR-IGE, because it could immediately adapt its policy in this phase.

In summary, the RR-IGE outperformed the classical Q-learning methods, because of three reasons. First, the RR-IGE can utilize its knowledge learned under one context in other contexts. Whereas, Q-learning needed to relearn its optimal policy for each context. Second, Q-learning needs to incorporate the time step information into its state space to find the optimal policy. This increases the complexity of the task and the need for more exploration. Finally, for the modules of the RR-IGE the task is deterministic. Thus, a higher learning rate ($\alpha = 1$) can be used, increasing the learning speed.

Discussion

The RR-IGE can directly adapt to different risk levels η in reward risk tasks. It learns different policies by its γ-modules as if there is no risk in the environment. Given some reward risk, the RR-IGE can directly map the risk to the γ-module with the optimal policy by $\gamma = 1 - \eta$. Given that the modules learned their optimal Q-functions, the mapping is guaranteed to result in the optimal policy.

As for interruption risk tasks, the risk η is given to the agents as task context, but it can also be learned by observing the task. However, compared to interruption risk tasks, the calculation of the risk can be more complicated. It depends on the reward function of the task. If the reward function cannot be 0 ($R^O \in \mathbb{R} \setminus 0$) unless the hidden state s^N is reached, the calculation is simple. In such tasks, it can be clearly detected when all future reward becomes zero, i.e. when hidden state s^N is reached, because at this time point the reward is 0 ($r_t = 0$). This allows to calculate the probability by observing how often this happens, for example during a certain time of the day with:

$$\eta = \frac{N^{r \to 0}}{N^{r \neq 0}} \ ,$$

where $N^{r\to 0}$ is number of episodes in which the reward went to zero ($r_t = 0$), and $N^{r\neq 0}$ the sum over the number of steps in all episodes until the reward went to zero.

For tasks that allow rewards to be zero, even if the state s^N is not reached, the calculation is more complicated, because it cannot be observed when the hidden state changes to s^N. One possible solution for such situations is to use a maximum likelihood estimation (Bishop 2006) for η:

$$\underset{\eta}{\operatorname{argmax}}\, \mathcal{L}(\eta; D) \ , \ \text{with} \ \ \mathcal{L}(\eta; D) = Pr(D|\eta) \ ,$$

where D is observed data from the task. The goal is to find the risk probability η that has the highest probability of explaining the collected experimental data. For example, in tasks such as those used for the experiments (Fig. 3.9), where reward is only given at the end of an episode, the likelihood can be computed by the following method. For each observed episode, the final reward r and the number of steps n are collected as data $D_i = (r_i, n_i)$. The probability for receiving reward ($r \neq 0$) after an episode of n steps is:

$$Pr(r \neq 0|n) \ = \ (1-\eta)^n \ ,$$

which can be derived from the transition probabilities T^H of the hidden state space S^H (Fig. 3.7). The probability for not receiving reward ($r = 0$) is

$$\begin{aligned} Pr(r = 0|n) &= 1 - Pr(r \neq 0|n) \\ &= 1 - (1-\eta)^n \ . \end{aligned}$$

Based on these probabilities the likelihood of a risk level η for data D can be calculated by:

$$\begin{aligned} \mathcal{L}(\eta; D) &= \prod_{i=1}^{|D|} \mathbb{I}(r_i = 0) Pr(r_i \neq 0|n_i) + \mathbb{I}(r_i \neq 0) Pr(r_i = 0|n_i) \\ &= \prod_{i=1}^{|D|} \mathbb{I}(r_i = 0)(1-\eta)^{n_i} + \mathbb{I}(r_i \neq 0)\left(1 - (1-\eta)^{n_i}\right) \end{aligned} \ ,$$

where $\mathbb{I}$ is the indicator function. The reward risk η which maximizes this likelihood should be used as context for the RR-IGE to select the most appropriate γ-module.

Another assumption that is not fulfilled for many potential tasks is that the reward function R^O is given to the agent. As a solution the agent could learn a model $\hat{R}^O$ of the reward function R^O based on the observations (s_t, a_t, r_t, s_{t+1}), but ignoring the observation with zero reward ($r_t = 0$). The model of the reward function could be for example learned by a Robbins-Monro approximation algorithm:

$$\begin{aligned} &\hat{R}^O_{\text{init}}(S, A) = 0 \ , \\ &\hat{R}^O_{t+1}(s_t, a_t) = \begin{cases} \hat{R}^O_t(s_t, a_t) & | \ r_t = 0 \\ \hat{R}^O_t(s_t, a_t) + \alpha_{\hat{R}} \left(r_t - \hat{R}^O_t(s_t, a_t) \right) & | \ r_t \neq 0 \end{cases} \ , \end{aligned}$$

where $\alpha_{\hat{R}}$ is the learning rate. By ignoring zero rewards ($r_t = 0$) the update for $\hat{R}^O_{k+1}$ will ignore observations if the hidden state s^N is reached. One problem with this method is that the learned reward function $\hat{R}$ will have a bias if the function $R^O(s, a)$ is a distribution over rewards with a non-zero probability for zero as outcome, because these will be ignored.

Similar to the interruption risk tasks, a different modular Q-learning algorithm can be also constructed to solve RR-MDPs. The algorithm has for each potential risk level η

an associated η-module. Each module learns from the observations (s_t, a_t, r_t, s_{t+1}) during an episode. The modules sample at each transition if the current reward and all future rewards will be zero, i.e. if hidden state s^N is reached, according to their associated risk η. If this is the case, then the module will pretend that all future reward during the episode is zero. By this, each η-module learns the optimal policy for its risk level. Depending on the reward risk for the current episode, the associated module's Q-function is used to define the policy of the agent. Nonetheless, similar to the alternative modular framework for IR-MDPs, for this framework the stochasticiy for η-modules with $\eta > 0$ is also increased. A lower learning rate α is required to learn the correct values compared to the RR-IGE. This decreases its learning speed. Therefore, the RR-IGE will have a better performance compared to the alternative modular approach.

3.3.3 Constant Punishment Tasks

The third variant of tasks for which an optimal mapping from context to γ-modules can be derived are constant punishment tasks. In these tasks, for each time step a punishment p is given to the agent that is constant during an episode (Fig. 3.10). The strength of the punishment is the task context. It can change between episodes. Constant punishment tasks are deterministic, goal-only-reward MDPs (Definition 3).

An example for a constant punishment task is an agent that has to collect energy from different possible sources with different payouts. For each action the agent looses some energy. The energy $p \in \mathbb{R}^-$ that the agent losses is fixed for an episode, but it depends on some factors such as the weight of the agent which changes depending on its load. The total energy that the agent gains in an episode is $t \cdot p + R$, where t is the number of time steps to reach the source and R is the energy that the agent gains from it. Depending on the punishment strength p, a different energy source might be optimal. If the punishment is high, then the agent should prefer nearby sources which do not need a lot of steps. Whereas, for a low punishment, more distant energy sources which have more energy provide a higher payoff.

Constant punishment tasks can be solved with a Constant Punishment IGE (CP-IGE). The CP-IGE can outperform classical Q-learning. Nonetheless, a different modular Q-learning framework, which is introduced in the discussion section, can be constructed that has the same advantages as the CP-IGE. Additionally, this framework is more general, because it can solve stochastic tasks. Therefore, the CP-IGE does not provide a practical algorithm for solving constant punishment tasks. However, it provides an interesting theoretical case for the ability of the IGE to handle different problem settings.

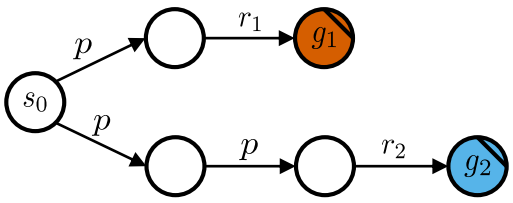


Figure 3.10: A Constant Punishment MDP with two goal states (g_1, g_2). For each step the agent receives a punishment p until it reaches a goal state.

The next section defines Constant Punishment MDPs (CP-MDPs) used to formulate constant punishment tasks. The CP-IGE is introduced and its optimality for solving CP-MDPs is proven. Following this, the experimental results from applying the CP-IGE to a CP-MDP are discussed, showing that it outperforms classical Q-learning.

Constant Punishment MDPs and the Constant Punishment Adaptive IGE

Definition 8. A Constant Punishment MDP CP-MDP$(A, S^E \times S^C, T, R)$ is a deterministic variant of a CMDP. Its transition probability function is defined over the environmental state space S^E and it is deterministic:

$$\forall s_i^E, a \; \exists s_j^E : \quad T(s_i^E, a, s_j^E) = 1 \; , \quad \text{and} \quad \sum_{s_k^E \in S^E} T(s_i^E, a, s_k^E) = 1 \; .$$

The reward function is defined over the environmental state space S^E. It gives for each transition a punishment p, unless a goal state S^G is reached:

$$R\left(s_i^E, a, s_j^E\right) = \begin{cases} p & \mid s_j^E \notin S^G \\ R^G(s_j^E) & \mid s_j^E \in S^G \end{cases} \; ,$$

where the punishment is $p \in \mathbb{R}^-$ is constant during an episode and the rewards for the goal states are $R^G : S \mapsto \mathbb{R}^+$. The context is the punishment p of an episode:

$$s^C = p \; .$$

The Constant Punishment Adaptive IGE (CP-IGE) can be used to solve CP-MDPs (Algorithm 3.3). The γ-modules of the CP-IGE learn the Q-values based on a modified reward function R^M:

$$R^M\left(s_t^E, a, s_{t+1}^E\right) = \begin{cases} 0 & \mid s_{t+1}^E \notin S^G \\ \exp\left(\mathrm{E}\left[R^G\left(s_{t+1}^E\right)\right]\right) & \mid s_{t+1}^E \in S_G \end{cases} \; . \tag{3.9}$$

An important point is that in the case of reaching a goal state the expectation over the reward for entering the state is used. If the reward function for goals is deterministic, then it can be directly applied. For stochastic rewards, an approximation of their expectation has to be learned which will be further discussed in the discussion section. An optimal mapping from the punishment context p to the appropriate γ-module with $\gamma = \exp(p)$ can be derived based on the reward function R^M.

Theorem 12. *An CP-IGE with a γ-map of*

$$G^*(\eta) = \exp(p) \; ,$$

has the optimal policy $\pi^*_{s^C}(s^E)$ for all states $(s^C, s^E) \in S$ of an CP-MPD, under the assumption that the γ-modules converged to their optimal Q-functions Q^*_γ and use their greedy policy.

Proof. Because CP-MDPs are deterministic and for each step a constant punishment is given until a goal state is reached, the expected reward sum for a policy π starting in state s_0 can be expressed as:

$$\mathrm{E}_{pi,s}\left[\sum_{t=0}^{T-1} r_t\right] = (n(s, s^G) - 1) \cdot p + \mathrm{E}\left[r_{s^G}\right] \ , \tag{3.10}$$

where $n(s, s^G)$ is the number of steps to reach goal state s^G starting from state s and r_{s^G} is the reward for reaching the goal state.

The CP-IGE learns the values of its γ-modules based on the modified reward function R^M (Eq. 3.9). The function gives a reward of 0 for all steps and the exponential over the expectation of the reward for reaching a goal state. Using the reward function R^M simplifies the definition of the Q-values to

$$\begin{aligned} Q_\gamma(s,a) &= \mathrm{E}\left[r_0\right] + \gamma \mathrm{E}\left[r_1\right] + \gamma^2 \mathrm{E}\left[r_2\right] + \cdots + \gamma^T \mathrm{E}\left[r_{T-1}\right] \\ &= 0 + \gamma 0 + \gamma^2 0 + \cdots + \gamma^{n(s,s^G)-1} \exp\left(\mathrm{E}\left[r_{s^G}\right]\right) \\ &= \gamma^{n(s,s^G)-1} \exp\left(\mathrm{E}\left[r_{s^G}\right]\right) \ . \end{aligned}$$

The log transformation of the Q-values:

$$\begin{aligned} \log\left(Q_\gamma(s,a)\right) &= \log(\gamma^{n(s,s^G)-1} \exp(\mathrm{E}\left[r_{s^G}\right])) \\ &= (n(s, s^G) - 1) \cdot \log(\gamma) + \mathrm{E}\left[r_{s^G}\right] \ , \end{aligned} \tag{3.11}$$

Algorithm 3.3: Constant Punishment IGE

Input:
Learning rate: $\alpha \in [0, 1]$
Sorted list of discount factors: $\Gamma = (\gamma_1, \ldots, \gamma_m)$ with $\gamma_i \in (0, 1)$

initialize $Q_\gamma(S^E, A)$ to zero
repeat (for each episode)
  initialize state s^E, context p
  `// map context to the active module` $\tilde{\gamma}$
  $\tilde{\gamma} \leftarrow \exp(p)$
  repeat (for each step in episode)
    $a \leftarrow$ choose an action for s^E derived from $\log(Q_{\tilde{\gamma}})$ (e.g. ϵ-greedy)
    r^M, $s^{E'} \leftarrow$ take action a, observe outcome
    forall the $\gamma \in \Gamma$ **do**
      `// learn values based on the exponential of the reward function` R
      $Q_\gamma(s^E, a) \leftarrow Q_\gamma(s^E, a) + \alpha \left(r^M + \gamma \max_{a'} Q_\gamma(s^{E'}, a') - Q_\gamma(s^E, a)\right)$
    end
    $s^E \leftarrow s^{E'}$
  until s^E *is terminal-state*
until *termination*

and the expected reward sum (Eq. 3.10) are equal for $\gamma = \exp(p)$. The logarithm is a monotonically increasing function. Thus, the actions that maximize the Q-function also maximize their log transformation ($\text{argmax}_a\, Q(s,a) = \text{argmax}_{a'}\, log(Q(s,a'))$). As a result, the module with $\gamma = \exp(p)$ also optimizes the reward sum (Eq. 3.10). This proves that the γ-map $G^*(p) = \exp(p)$ will choose the module with the optimal greedy policy for each punishment context p. □

Because the logarithm of the Q-values (Eq. 3.11) results in the expected reward sum that should be optimized, the CP-IGE uses the log of the Q-functions for the action selection (Algorithm 3.3). Nonetheless, also the direct Q-values could be used, because the logarithm is a monotonic increasing function, meaning that the action with the highest value has also the highest value after the log transformation.

Experiments

The CP-IGE was compared to a standard Q-learning algorithm in a grid world variant of a CP-MDP with five goal states (Fig. 3.11, a). Reaching a goal state gives a positive reward (Fig. 3.11, b). Otherwise, for each step a punishment p is given. Therefore, the expected reward and the optimal goal state depends on p (Fig. 3.11, c).

Each experimental run consisted of four phases with 1200 episodes per phase. The phases differed in their punishment p. Phase 1 had a small punishment ($p = -0.2$), followed by Phase 2 that had an intermediate punishment ($p = -0.75$). Phase 3 had no punishment ($p = 0$) and Phase 4 had a high punishment ($p = -1.3$). Fig. 3.11 (c) shows that the optimal choice changed between each phase with c_5 for Phase 1, c_4 for Phase 2, c_5 for Phase 3 and c_2 for Phase 4. The experiment tested if, and how fast, the CP-IGE can adapt to changes in punishment compared to traditional Q-learning.

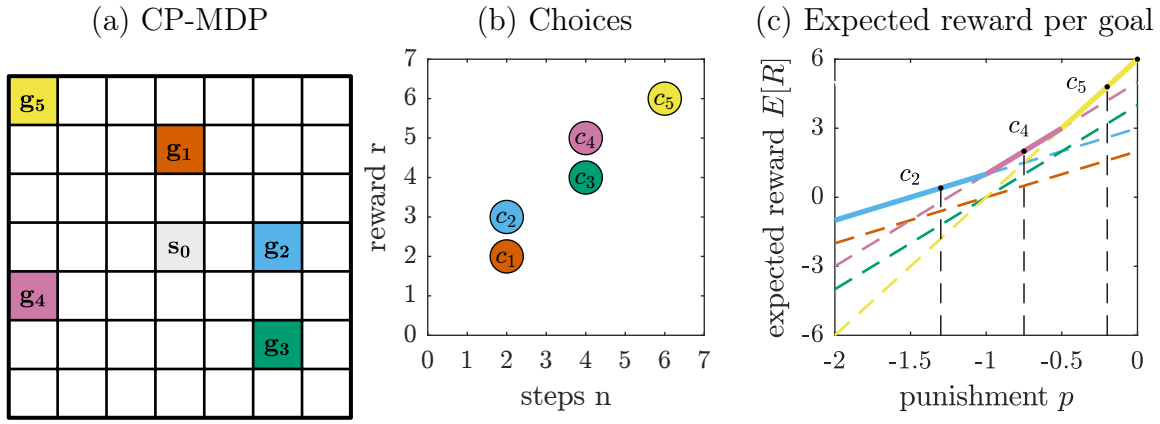


Figure 3.11: (a) The Constant Punishment MDP used for the experiments consisted of one start state s_0 and 5 goal states ($g_1, \ldots, g_5$). (b) Each goal state represents a different choice with different rewards and a number of minimum steps to reach them. (c) The expected reward of a choice depends on the punishment p for each step. For a high punishment choice c_2 has the highest expected reward. Whereas, choices c_4 and c_5 are optimal for intermediate and low punishment. The black points and dashed lines show the punishment levels of the experimental phases.

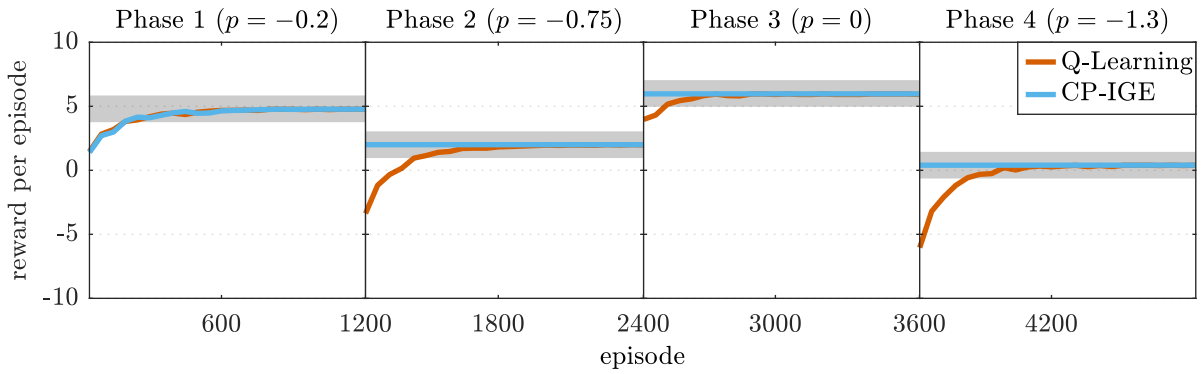


Figure 3.12: The CP-IGE adapts immediately to changing punishments. Q-learning has to learn the optimal policy for each phase independently.

The CP-IGE uses the position x in the grid world as environmental state space for its γ-modules ($s^E = x$). The punishment level p is given as context information ($s^C = p$). Q-learning uses as state space the combination of both ($s = (x, p)$). Both algorithms used a learning rate of $\alpha = 1$ and an exponentially decaying ϵ-greedy action selection with $\epsilon_{init} = 1$, $d = 0.995$ and $\epsilon_{min} = 0$. For the CP-IGE, ϵ stayed at zero after the first phase. For Q-learning, ϵ was reset after each phase to allow it to explore and update its Q-values to the new punishment level. The CP-IGE did not need to update its Q-values after phase transitions.

The results show that CP-IGE and Q-learning behaved similar for Phase 1 (Fig. 3.12). Both learned the optimal choice c_5 with the same number of episodes. In the successive phases where punishment exists, CP-IGE outperformed Q-learning. The CP-IGE could immediately adapt to the new punishment levels. Q-learning needed to learn a new policy for each punishment level.

Discussion

The CP-IGE can guarantee to be optimal under different constant punishments in deterministic and goal-only-reward tasks. It can outperform classical Q-learning in such tasks, because it can generalize policies learned under one punishment level to other levels.

The punishment level is given to the CP-IGE, but it can also be simply observed. As long as the agent does not enter a goal state the punishment is the reward that the agent receives ($p = R(s_t, a_t)$).

A problem for the framework exists for tasks where the reward that the agent receives by reaching a goal state is stochastic. Then, the reward function used for the γ-modules (Eq. 3.9) has to give directly the expectation over the reward ($\mathrm{E}\left[R^G\left(s^E_{t+1}\right)\right]$). This expectation is usually not given for a task. However, the expectation could be learned in an extra step based on observations (s_t, a_t, r_t, s_{t+1}). For each goal state s^G the expectation $\hat{R}^G$ over its reward can be learned by a Robbins-Monro approximation algorithm:

$$\hat{R}^G_{k+1}(s_t, a_t) = \hat{R}^G_k(s_t, a_t) + \alpha_{\hat{R}} \left(r_t - \hat{R}^G_k(s_t, a_t) \right) \ .$$

The Q-modules could then use $\hat{R}^G$ to define the rewards for entering goal states ($r^M_t =$

$\exp(\hat{R}^G(s_t, a_t)))$. This allows the framework to solve tasks with stochastic reward functions.

Although, the CP-IGE can guarantee optimality for CP-MDPs, its practical application is not recommended, because a different Q-learning algorithm that is more general can be constructed. This Q-learner is also optimal and, in contrast to the CP-IGE, it can handle environments with stochastic transitions. Moreover, it does not need to model the expectation $\hat{R}^G$ if a stochastic reward function is used for goal states. This Q-learner has a module for each potential punishment level p. The reward function for each module depends on its associated punishment level p with:

$$R^M\left(s_t^E, a, s_{t+1}^E\right) = \begin{cases} p & \mid s_{t+1}^E \notin S^G \\ r_t & \mid s_{t+1}^E \in S_G \end{cases} ,$$

where the module receives the punishment for normal transitions and the regular reward for entering a goal state. The modules can learn in parallel from the same observations similar to the IGE. Each module learns the optimal policy for its associated punishment level. This allows this Q-learner to learn the optimal policy for each punishment level in parallel, even in stochastic environments. The policy of the agent is then based on the module which is associated with the current punishment level. Although, this approach outperforms the CP-IGE because it is more general, the CP-IGE still provides an interesting theoretical case for the abilities of the IGE.

3.4 Learning of γ-Mappings

The previous section discussed special Context MDPs for which an optimal mapping from context to γ-modules can be derived. This section discusses how the CA-IGE can be applied to general CMDPs in which a γ-mapping needs to be learned. Three variants of the CA-IGE algorithm are introduced that can learn γ-mappings. For the CMDPs where a γ-mapping has to be learned, the convergence of the CA-IGE to the optimal policy cannot be guaranteed. In contrast, classical algorithms such as Q-learning, which learn over the whole state space $(S^E \times S^C)$ of the CMDP, are guaranteed to converge to the optimal policy. Nonetheless, the CA-IGE can outperform classical Q-learning in the rate of learning, because it reduces the complexity of CMDPs. For tasks where the agent needs to adapt to changing task conditions the rate of learning is often more important than the final performance.

The problem of learning a γ-map can be formulated as an MDP with the set of γ-Modules Γ as its action set:

$$\text{G-MDP}(\Gamma, S^E \times S^C, T, R) , \tag{3.12}$$

where the state space $(S^E \times S^C)$, the reward function R, and the transition probability function T are the same as for the CMDP. The goal is find a mapping G, the policy for the G-MDP, to maximize the expected reward sum for each episode (Eq. 3.1). Because the G-MDP is an MDP, the γ-map can be learned by any reinforcement learning algorithm that can solve MDPs. Section 3.4.1 introduces a model-free, temporal difference method and two policy search methods to learn the γ-map. The CA-IGE variants are compared to classical Q-learning in grid-world versions of the energy foraging task in Section 3.4.2.

Two variations for the reward function R^M of the γ-modules are analyzed in Section 3.2.2. The first uses the general reward function R from the overall MDP, but updates values only for observations from a subset $\Phi \in S^C$ of contexts. Although, this increases the number of needed observations to learn the value functions, the CA-IGE still outperforms classical Q-learning. The second variation decomposes the reward function R into a component R^E that takes only the task state space S^E into account, and R^C that takes only the context S^C into account. This allows the γ-modules to learn from all observation in the task, by using only the reward function R^E. This improves the performance of the CA-IGE significantly compared to the version which learns only under certain contexts and to classical algorithms.

3.4.1 Algorithms to Learn γ-Mappings

Three variants of the CA-IGE are introduced to learn the γ-mapping from context to the most appropriate γ-module. Because the learning of the γ-map G is an MDP (Eq. 3.12), all variants are reinforcement learning methods. The first variant (Q-IGE) uses Q-learning to learn the mapping. The other two variants (PGPE-IGE and EPHE-IGE) use policy search methods.

The Context Adaptive Q-IGE

The Context Adaptive Q-learning Independent γ-Ensemble (Q-IGE) is a CA-IGE variant that use a Q-learning approach to learn the γ-map (Algorithm 3.4). Its γ-map G is based on a learned value function ($Q^G : S \times \Gamma \mapsto \mathbb{R}$) over the state space S and the modules Γ. The state space is either the task and context space ($S = (S^E \times S^C)$) or only the context space ($S = S^C$) of the CMDP. The value for a state-module pair $Q^G(s, \gamma)$ is the sum over the expected discounted future rewards, if the γ-module's greedy policy is used in state s and subsequent steps follow the mapping G. It can be formulated in form of the Bellman equation:

$$\begin{aligned} Q^G(s,\gamma) &= \mathrm{E}_G\left[\sum_{t=0}^{T-1} \gamma_G^t r_t\right] \\ &= R(s, \pi_\gamma(s)) + \gamma_G \cdot \mathrm{E}_{s'}\left[\max_{\gamma'} Q^G(s', \gamma')\right] \ , \end{aligned}$$

where $\gamma_G \in [0, 1]$ is a discount factor. The map $G(s)$ is defined by the greedy action over the Q^G function:

$$G(s) = \underset{\gamma}{\operatorname{argmax}}\, Q^G(s, \gamma)$$

A Q-learning approach is used to learn the optimal value function Q^{G*}. It should be noted that the optimal mapping G^* solves the G-MDP (Eq. 3.12) optimally, but not necessarily the CMPD.

The Q-IGE learns for each state-γ pair a value. The IGE may have a large set of γ-modules. If a standard Q-learning approach would be used to learn the values, it would take many observations and a lot of exploration to learn for each γ-module the values. However, the Q-IGE can be altered in two ways to allow an efficient learning of the values, and exploration.

First, the Q-IGE has a parallel update of values for γ-modules with the same greedy policy after one observation. The Q-IGE selects the γ-module $\tilde{\gamma}_t$ to select an action a_t

Algorithm 3.4: Context Adaptive Q-IGE

Input:
Learning rates: $\alpha_G \in [0,1]$, $\alpha_Q \in [0,1]$
Discount factor for γ-mapping: $\gamma_G \in [0,1]$
Sorted list of discount factors: $\Gamma = (\gamma_1, \ldots, \gamma_m)$ with $\gamma_i \in (0,1)$

initialize Q^G to zero
repeat (for each episode)
  initialize state (s^E, s^C)
  repeat (for each step in episode)
    `// select the active` γ`-module after each transition`
    $\tilde{\gamma} \leftarrow$ select module for state (s^E, s^C) based on Q^G (e.g. ϵ-greedy)
    $a \leftarrow$ select action for state s^E based on $Q_{\tilde{\gamma}}$ (e.g. ϵ-greedy)
    (r^M, r), $(s^{E'}, s^{C'}) \leftarrow$ take action a, observe outcome
    `// identify all` γ`-modules for which action` a `is optimal`
    $\hat{\Gamma} \leftarrow (\gamma_i)_{\forall \gamma_i:\ a\ =\ \mathrm{argmax}_u\, Q_{\gamma_i}(s^E,u)}$
    `// update` Q^G `for all modules in` $\hat{\Gamma}$
    forall the $\gamma \in \hat{\Gamma}$ **do**
      $Q^G(s^E, s^C, \gamma) \leftarrow Q^G(s^E, s^C, \gamma) + \alpha_G \left(r + \max_\gamma \gamma_G Q^G(s^{E'}, s^{C'}, \gamma) - Q^G(s^E, s^C, \gamma)\right)$
    end
    forall the $\gamma \in \Gamma$ **do**
      $Q_\gamma(s^E, a) \leftarrow Q_\gamma(s^E, a) + \alpha_Q \left(r^M + \gamma \max_{a'} Q_\gamma(s^{E'}, a') - Q_\gamma(s^E, a)\right)$
    end
    $(s^E, s^C) \leftarrow (s^{E'}, s^{C'})$
  until *s is terminal-state*
until *termination*

based on its policy. It performs the action and observes the resulting reward r_t and the new state s_{t+1}. Then, it updates the values for all modules $\hat{\Gamma}$ for which action a_t follows their greedy policy:

$$\hat{\Gamma} = (\gamma_i)_{\forall \gamma_i \in \Gamma:\ a_t\ =\ \mathrm{argmax}_{\hat{a}}\, Q_{\gamma_i}(s^E,\hat{a})}\ .$$

All these modules are updated, because all their greedy policies would take action a_t and result in the same observation. Thus, the values of many γ-modules can be updated in parallel after one observation.

Second, the Q-IGE explores over greedy policies and not γ-modules. An ϵ-greedy selection strategy is used. First, it samples if with probability ϵ an exploration step should be performed. If not, it will use the γ-module with the maximum value ($\tilde{\gamma} = \mathrm{argmax}_\gamma\, Q^G(s,\gamma)$). In the case of an exploration step, one approach would be to select a random module as active γ-module $\tilde{\gamma}$. However, there is a problem with this approach

because many of the modules usually have the same greedy policy. For example, out of 100 modules, 95 might have a_1 as greedy action and 5 have a_2 as greedy action. If the policy of a random module would be explored, action a_1 would have a probability of 0.95 and a_2 of 0.05 to be used. This is problematic because the goal of the random exploration is to choose between different policies randomly in an uniform manner. Each policy should have the same probability. To solve this problem, the ϵ-greedy module selection explores over policies and not modules. For an exploration step, the greedy actions of all modules are collected in a set without repetition. For the given example of 100 modules with two greedy actions, the set would be (a_1, a_2). From the set of greedy actions, one of them is randomly selected. Then, one of the modules which has this action as greedy action is randomly selected. This avoids that policies, i.e. greedy actions, are oversampled just because more modules have them as their greedy policy.

The parallel update of values for γ-modules with the same greedy policy, and the exploration over greedy policies and not γ-modules allow the Q-IGE to learn the γ-map with a large set of γ-modules.

The Context Adaptive PGPE and EPHE-IGE

The Q-IGE uses value-based Q-learning to learn the γ-mapping. The next two variants of the CA-IGE are policy-search methods. The first is the Context Adaptive Parameter Exploring Policy-Gradients Independent γ-Ensemble (PGPE-IGE) based on the PGPE (Sehnke et al. 2010). The second approach is the Context Adaptive EM-based Policy Hyperparameter Exploration Independent γ-Ensemble (EPHE-IGE) based on the EPHE (Wang, Uchibe, & Doya 2017). Both algorithms are policy search approaches for deterministic policies. Their goal is to optimize the deterministic γ-map G which is defined by a parameter vector $\theta \in \Theta$:

$$G(s^C, \theta) = \theta_1 + \frac{1 - \theta_1}{1 + \exp(-\theta_2(s^C - \theta_3))} \ . \tag{3.13}$$

The map G has a sigmoidal form. See Fig. 3.14 (d) for an example.

Standard policy search methods like REINFORCE (Williams 1992) use stochastic policies $\pi(a_t|s_t, \theta)$ which are parameterized by θ. Before both algorithms are introduced, the general approach of stochastic policy search methods are reviewed to provide an understanding why a deterministic and not a stochastic policy is used.

The goal of policy search methods is to find the parameter that maximizes the objective function J which is the expected reward sum of the agent:

$$J(\theta) = \int_H p(h|\theta) R(h) dh \ ,$$

where $h = [s_0, a_0, r_0, \ldots, s_{T-1}, a_{T-1}, r_{T-1}, s_T]$ is a trajectory history, and $R(h) = \sum_{t=0}^{T-1} r_T$ is the reward sum of a trajectory. In MDPs the probability of a trajectory is given by:

$$p(h|\theta) = p(s_0) \prod_{t=0}^{T-1} p(s_{t+1}|s_t, a_t) \pi(a_t|s_t, \theta) \ , \tag{3.14}$$

where $p(s_0)$ is the initial state distribution, and $p(s_{t+1}|s_t, a_t)$ is the transition probability (usually denoted as T in MDPs). Standard policy search methods such as REINFORCE

maximize the objective J iteratively. They follow the gradient $\nabla_\theta J(\theta)$ with respect to θ using gradient ascent:

$$\theta_{k+1} = \theta_k + \alpha \nabla_\theta J(\theta) \ ,$$

where α is the step size. If the policy $\pi(a_t|s_t, \theta)$ is stochastic the gradient can be calculated by

$$\nabla_\theta J(\theta) = \mathrm{E}_H \left[\sum_{t=0}^{T-1} \nabla_\theta \log \pi(a_t|s_t, \theta) R(h) \right] \ .$$

To compute the gradient, several trajectories H are collected for the current parameter θ_k. The gradient $\nabla_\theta J(\theta)$ is then computed to updated the parameters. However, stochastic policies induce a high variance into the sampling because at each time point an action a_t gets sampled. To compensate for this, many samples are required to evaluate a single policy parameter θ.

To avoid the high variance problem of stochastic policies, a deterministic policy for the γ-map (Eq. 3.13) is used. The probability for a trajectory (Eq. 3.14) has to be modified for deterministic policies to:

$$p(h|\theta) = p(s_0) \prod_{t=0}^{T-1} p(s_{t+1}|s_t, a_t = \pi(a_t|s_t, \theta)) \ .$$

As a result, the derivative of it is given by:

$$\nabla_\theta a_t = \pi(a_t|s_t, \theta) = \left. \frac{\partial p(s_{t+1}|s_t, a_t)}{\partial s_t} \frac{\partial a_t}{\partial \theta} \right|_{a_t = \pi(a_t|s_t, \theta)} \ ,$$

requiring the knowledge of the state transition probability (Deisenroth, Neumann, & Peters 2011). For CMDPs, the state transition probability is not given to the agent.

To solve this problem, the PGPE and EPHE use a multivariate Gaussian distribution over the policy parameters θ:

$$p(\theta|\mu, \sigma) = \mathcal{N}(\mu, I\sigma)$$

The goal of the algorithms changes to optimize the meta-parameters $\rho = (\mu, \sigma)$ of this distribution:

$$J(\rho) = \int_\Theta \int_H p(h|\theta) p(\theta|\rho) R(h) dh d\theta \ .$$

PGPE uses stochastic gradient ascent $\rho_{k+1} = \rho_k + \alpha \nabla_\rho J(\rho)$ to maximize $J(\rho)$. The gradient is given by

$$\nabla_\rho J(\rho) = \int_\Theta \int_H p(h|\theta)(R(h) - b) \nabla_\rho p(\theta|\rho) dh d\theta \ ,$$

where b is a reward baseline. The baseline is used to decrease the variance of the gradient estimation. EPHE uses an idea from EM-based Policy Search (Peters & Schaal 2007). It maximizes a lower bound of the objective function $J(\rho)$ by a new parameter distribution.

The general framework of both algorithms is identical (Algorithm 3.5). The policy parameter distribution is for both initialized by the hyper-parameters μ^{init} and σ^{init}. The

Algorithm 3.5: PGPE-IGE and EPHE-IGE

Input:

Inital γ-map hyper-parameters: $\mu_{init} \in \mathbb{R}^N, \sigma_{init} \in \mathbb{R}^N$

Number of samples: $\#samples, \#subSamples$

γ-Module learning rate: $\alpha_Q \in [0, 1]$

Sorted list of discount factors: $\Gamma = (\gamma_1, \ldots, \gamma_m)$ with $\gamma_i \in (0, 1)$

initialize $\mu \leftarrow \mu_{init}, \sigma \leftarrow \sigma_{init}$

repeat

  for $i \leftarrow 1$ *to* $\#samples$ **do**

    `// draw the policy parameters for the γ-map`

    draw $\theta^i \sim \mathcal{N}(\mu, I\sigma^2)$

    `// evaluate the policy parameters by several episodes`

    for $j \leftarrow 1$ *to* $\#subSamples$ **do**

      initialize state s^E, context s^C

      $r^j_{episode} \leftarrow 0$

      repeat (for each step in episode)

        `// get active module from the γ-map`

        $\tilde{\gamma} \leftarrow G(s^C, \theta^i)$

        $a \leftarrow$ choose an action for s^E derived from $Q_{\tilde{\gamma}}$ (e.g. ϵ-greedy)

        (r^M, r), $(s^{E'}, s^{C'}) \leftarrow$ take action a, observe outcome

        forall the $\gamma \in \Gamma$ **do**

          $$Q_\gamma(s^E, a) \leftarrow Q_\gamma(s^E, a) + \alpha_Q \left(r^M + \gamma \max_{a'} Q_\gamma(s^{E'}, a') - Q_\gamma(s^E, a)\right)$$

        end

        $r^j_{episode} \leftarrow r^j_{episode} + r$

        $(s^E, s^C) \leftarrow (s^{E'}, s^{C'})$

      until s^E *is terminal-state*

    end

    $r^i_{samples} \leftarrow \frac{1}{\#subSamples} \sum_j r^j_{episode}$

  end

  `// update the hyper parameters by Algorithm 3.6 or 3.7`

  $[\mu, \sigma] \leftarrow$ update of the policy distribution with $(\mu, \sigma, \theta, r_{samples})$

until *termination*

Algorithm 3.6: PGPE-IGE - update of the policy distribution

Input:

Policy distribution: μ, σ

Sampled parameters and corresponding returns: θ, r

Step size parameters: α_μ, α_σ

$T \leftarrow [t_{ij}]_{ij}$ with $t_{ij} = \theta_i^j - \mu_i$

$S \leftarrow [s_{ij}]_{ij}$ with $s_{ij} = \frac{t_{ij}^2 - \sigma_i^2}{\sigma_i}$

$r \leftarrow [(r^1 - b), \cdots, (r^N - b)]^T$ with $b = \frac{1}{|r|} \sum_i r_i$

$\mu' \leftarrow \mu + \alpha_\mu T r$

$\sigma' \leftarrow \sigma + \alpha_\sigma S r$

return μ', σ'

Algorithm 3.7: EPHE-IGE - update of the policy distribution

Input:

Policy distribution: μ, σ

Sampled parameters and corresponding returns: θ, r

$b \leftarrow \frac{1}{|r|} \sum_i r_i$

$r \leftarrow [\max(0, r^1 - b), \cdots, \max(0, r^N - b)]^T$

$\mu' \leftarrow [\mu_i]_i$ with $\mu_i = \frac{\sum_{n=1}^N [r^n \theta_i^n]}{b}$

$\sigma' \leftarrow [\sigma_i]_i$ with $\sigma_i = \sqrt{\frac{\sum_{n=1}^N [r^n (\theta_i^n - \mu_i')^2]}{b}}$

return μ', σ'

update of the policy parameter distribution is done iteratively. At each iteration several parameters θ are sampled from the parameter distribution:

$$\theta^i \sim \mathcal{N}(\mu, I\sigma) \ .$$

The number of parameters is defined by $\#samples$. The parameters are evaluated by performing several episodes and collecting the reward sum of each episode. Usually, only one episode per parameter is executed (Sehnke et al. 2010; Wang, Uchibe, & Doya 2017). For the application of the PGPE and EPHE to the energy foraging task (Section 3.4.2) several episodes are sampled. The foraging task has varying initial energy levels over episodes. Hence, to evaluate how a γ-map with the parameters θ^i performs for different energy levels, it has to be evaluated over several episodes. The number of sampled episodes per parameter is given by $\#subSamples$. The reward sum r^i of parameter θ^i is given by the average reward sum over the collected episodes:

$$r^i = \frac{1}{\#subSamples} \sum_j r_{episode}^j \ .$$

Based on the sampled parameters θ^i and their evaluated reward r^i, the hyper-parameters μ and σ are updated.

The update of the hyper-parameters differs between the PGPE and the EPHE . The PGPE (Algorithm 3.6) calculates the gradient of the objective function and uses gradient ascent to update the hyper parameters. The gradient ascent uses for each hyper-parameter (μ, σ) separate step size parameters $(\alpha_\mu, \alpha_\sigma)$. The baseline b is set to the average reward over the sampled reward sums ($b = \frac{1}{|r|}\sum_i r_i$). The EPHE (Algorithm 3.7) has no further parameters. It uses an expectation maximization approach to update policy parameter distribution directly, only based on the sampled parameters and their returns and not on the past distribution.

3.4.2 Experimental Design

The performance of the CA-IGE algorithms that learn γ-maps (Q-IGE, PGPE-IGE and EPHE-IGE) has been compared to classical Q-learning in energy foraging tasks. This section describes the tasks and the experimental procedure. The later part describes the learning parameters for each algorithm.

The Energy Foraging Tasks

The tasks are grid world variants of the discussed energy foraging example (Fig. 3.1). The agent has to forage for energy sources in a grid world. Each task is formulated as a CMPD$(A, (S^E, S^C), T, R)$. The position of the agent in the grid world is the task state space S^E. The agent can move by the actions $A =$ (north, east, south, west). The task can be stochastic, i.e. a probability of η exist for each transition that the agent moves in a random direction, instead of its intended direction. The agent has an energy level $s^C \in \mathbb{R}^+$, which is the task context. It decreases for each action that the agent takes by $p \in \mathbb{R}^-$ until it reaches 0. If the energy level reaches zero the agent receives a strong negative punishment $p_0 \in R^-$ for each further movement. If the agent reaches an energy source, i.e. a goal state s_G that gives energy r_G, it can refill its energy level. The change of the energy level s^C during a transition is measured by the reward function R with:

$$s^C_{t+1} = \max\left(0, s^C_t + R(s^E_{t+1}, s^C_{t+1})\right) \ .$$

The reward function R is associated with the gain and loss of energy during a transition. It depends only on the state s_{t+1} to which the agent transitions:

$$R(s^E_{t+1}, s^C_{t+1}) = R^P(s^E_{t+1}) + R^C(s^C_{t+1}) \ . \tag{3.15}$$

It is composed of two components. The first component R^P depends on the position of the agent:

$$R^P(s^E) = \begin{cases} r_G & | \text{ if } s^E \in S^G \ \ (s^E \text{ is a goal state}) \\ p & | \text{ if } s^E \notin S^G \ \ (s^E \text{ is not a goal state}) \ , \end{cases}$$

where r_G is the reward for reaching goal state s^G, and p is the loss of energy the agent has for each action not ending at an energy source. The second reward component R^C only depends on the energy level of the agent:

$$R^C(s^C) = \begin{cases} p_0 & | \text{ if } s^C = 0 \ \ (\text{agent reached zero energy}) \\ 0 & | \text{ if } s^C > 0 \ \ (\text{agent has energy}) \ . \end{cases}$$

It gives the agent a strong negative reward for each action if it reaches a zero energy level. The goal of the agent is to maximize the reward function per episode:

$$\max_{\pi} \mathrm{E}_{\pi} \left[\sum_{t=0}^{T-1} R(s_t, a, s_{t+1}) \right] .$$

As a result, the agent tries to gain as much energy as possible during an episode without reaching an energy level of zero.

Different foraging tasks have been used to evaluate the algorithms under different conditions. The tasks differed in their layout, i.e. the number, location, and strength of energy sources, and in their settings of the stochasticity and the punishment strengths.

A simple, a moderate and a complex grid-world layout (Fig. 3.13) have been chosen to test how task complexity influences the performance of each algorithm. For each layout, the agent starts in a fixed start position s_0. An episode ends, either if the agent reaches one of the goal states $g_i \in S_G$, or after a maximum of 30 steps. The complexity of the layouts increases by an increasing number of states and goal states, and by an increasing number of steps needed to reach the farthest goal. A more complex layout needs a stronger exploration to learn the optimal policies to reach all goal states. The simple layout has 36 states with 4 goal states the start state. The farthest goal state can be reached within 6 steps by an optimal policy and if the task has no stochasticity. The moderate layout has 49 states with 6 goal states. The most distant goal being 9 steps away from the start state. The complex layout has 100 states with 8 goal states. The farthest goal needs 12 steps to be reached.

Five different task settings for the stochasticity and the punishments were used for each layout (Table 3.1). The different settings test if the algorithms show a consistent learning behavior under different task scenarios. In the first setting, the task is deterministic, the movement punishment is set to $p = -1$ and the zero-energy punishment is $p_0 = -9$. In the second setting, stochasticity of $\eta = 0.1$ is introduced. The third setting has an increased stochasticity of $\eta = 0.15$. For the fourth setting the movement punishment is increased to $p = -1.5$ and in the fifth setting the zero-energy punishment is increased to $p_O = -14$. Both have a stochasticity of $\eta = 0.1$.

The task layouts were designed so that the IGE is not able to learn the optimal policy for the tasks. Every goal state is an optimal solution for a certain energy level between 1 and 20 with which the agent starts in the start state s^0. Thus, to be optimal an agent should learn the policy to reach each goal state, but the IGE can only learn to reach a subset of goals as shown by the γ-Curves for the deterministic setting (Fig. 3.13). As a result, a classical algorithm, such as Q-learning, could outperform the IGE variants, because it is able to learn the optimal policy. Nonetheless, the experimental results showed that the IGE variants learned faster and that they usually found a better final policy than classical Q-learning (Section 3.4.3).

All algorithms were tested in each combination of layout and task setting. For each algorithm 100 runs were performed to measure their mean performance. Each run consisted of $20,000$ episodes. The agent's initial energy level was pseudo randomly selected between 1 and 20 at the beginning of each episode. The distribution was uniform and over natural numbers: $s^C_{t=0} \sim U(1, 20)$. It was pseudo random so that for every block of 20 episodes all energy levels between 1 to 20 were used once. This procedure allowed the

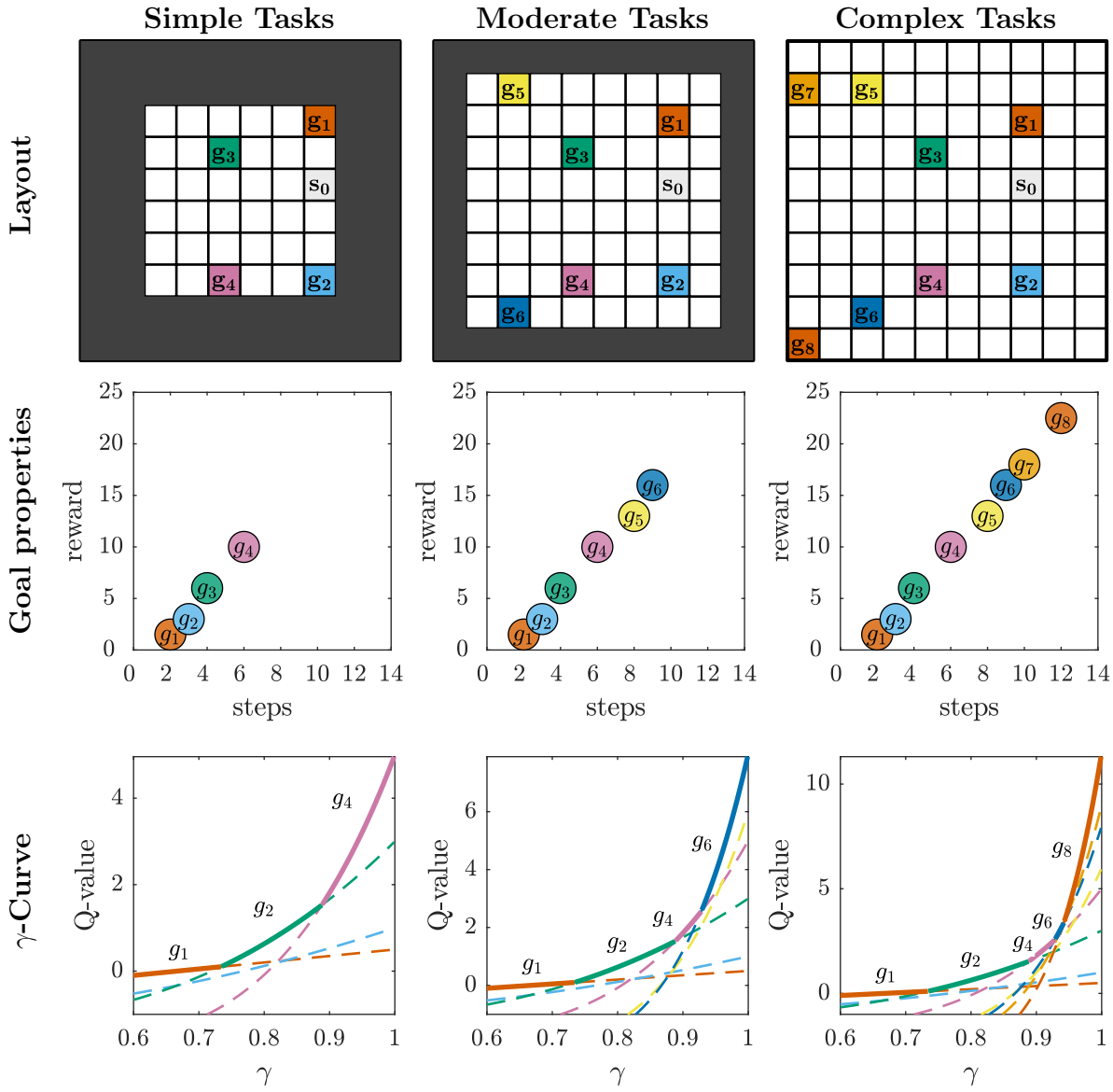


Figure 3.13: The algorithms were evaluated on three task layouts. The layouts increase in complexity. The number of states increases from 36 to 64 to 100. The number of goal states from 4 to 6 to 8. The distance to the farthest goal increases from 6 to 9 to 12 steps. The IGE cannot learn to reach every goal state from the start state s_0. The γ-curve (bottom) for the deterministic setting of the task (see Table 3.1) shows that the IGE modules can learn for the simple task layout to reach 3 out of 4, for the moderate 4 of 6, and for the complex setting 5 of 8 goals.

Settings	**Stochasticity** η	**Movement punishment** p	**Zero energy punishment** p_0
1) deterministic	0	-1	-9
2) stochastic	**0.1**	-1	-9
3) high stochastic	**0.15**	-1	-9
4) high p	0.1	**-1.5**	-9
5) high p_0	0.1	-1	**-14**

Table 3.1: Each task layout was tested under five settings. The settings differ in their tasks stochasticity and punishment strengths.

policy search algorithms (PGPE and EPHE-IGE) to draw 20 sub-samples, each with a different initial energy level, for every sampled policy parameter θ^i.

Learning Parameters

The learning parameters of the algorithms were manually optimized to achieve a high final performance with a reasonable exploration strategy for the stochastic setting ($\eta = 0.1, p = -1, p_0 = -9$) of the complex task layout.

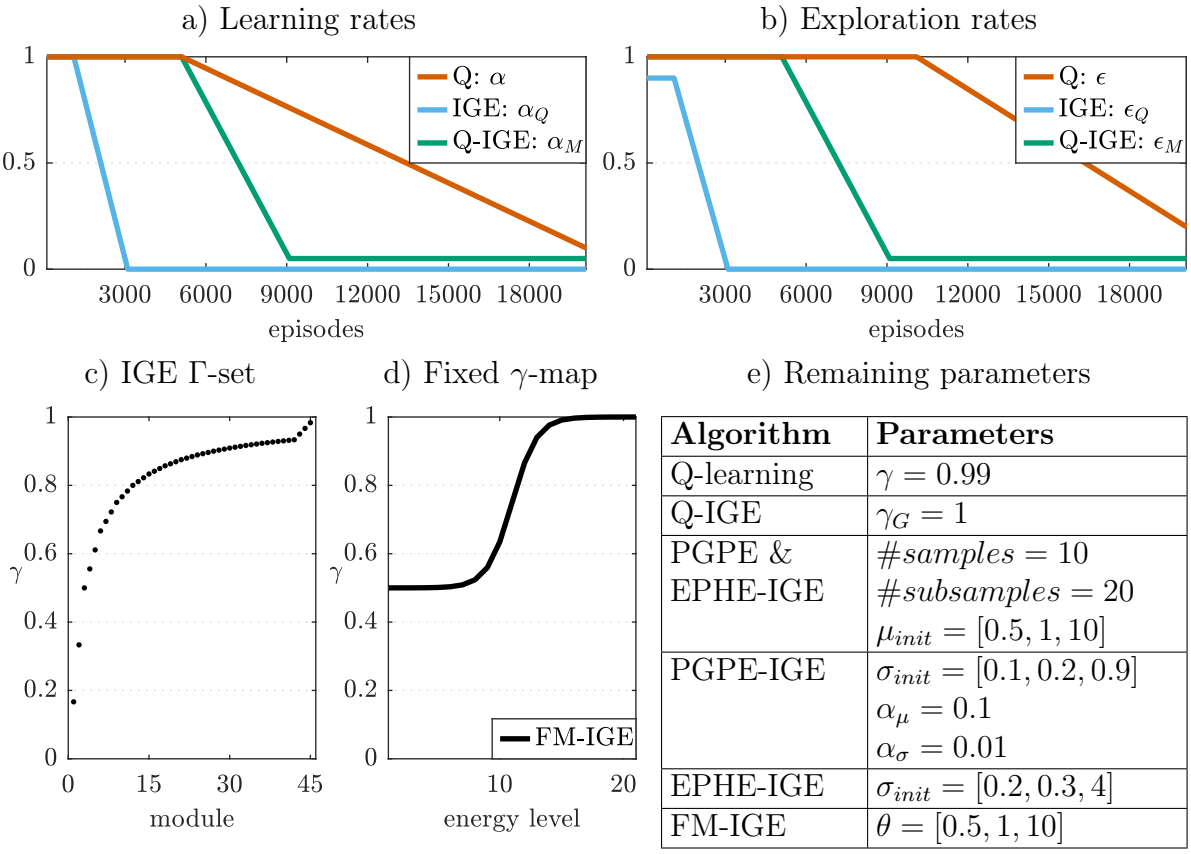
All IGE algorithms used the same set of 45 γ parameters for their modules (Fig. 3.14, c). The chosen γ parameters follow the recommended set of modules according to Theorem 9 (page 47). First, 14 modules were chosen according to: $\gamma_{i=1:14} = \frac{i}{i+1}$. Then, between each module γ_i and its smaller module γ_{i-1} two extra modules were added with equal distances between all modules. Between the last module $\gamma_i = 14$ and 1, three modules were added with equal distances between all modules. This procedure resulted in a distribution of γ parameters that becomes more dense toward $\gamma = 0.933$.

The learning rates and the exploration parameters change over the episodes for all algorithms. All algorithms have a stronger learning rate and exploration in the beginning of the learning. This was needed to achieve a good learning performance.

The γ-modules of each IGE algorithm were given 3000 episodes to learn (Fig. 3.14, a, b). The learning rate α_Q was set to 1 for 1000 episodes and then linearily decreased to 0 until episode 3000. The exploration rate ϵ_Q was accordingly set to 0.9 for 1000 episodes and then linearily decreased to 0 until episode 3000.

The learning of the γ-map for the policy search algorithms (EPHE-IGE and PGPE-IGE) started after the modules finished their learning. They could not learn the map in parallel to the learning of the modules. The performance of the modules, i.e. the resulting reward if they are used as active modules, varies greatly during their learning because of the exploration. As a result, the distributions (μ, σ) that the PGPE and EPHE-IGE learn over the policy parameters θ tend to become very extreme. The algorithms could not recover from these extreme distributions, even if the modules had learned their policies and their performance stabilized. To avoid this problem, the learning of the γ-map for the EPHE-IGE and PGPE-IGE was started after 3000 episodes when the modules finished learning.

The Q-IGE algorithm did not have this problem and could learn the γ-map and the γ-modules in parallel. Its learning rate α_M and the exploration rate ϵ_M were set to 1 for 5000 episodes and then linearily reduced to 0.1 until episode 9000 (Fig. 3.14, a, b).



e) Remaining parameters

Algorithm	**Parameters**
Q-learning	$\gamma = 0.99$
Q-IGE	$\gamma_G = 1$
PGPE & EPHE-IGE	$\#samples = 10$ $\#subsamples = 20$ $\mu_{init} = [0.5, 1, 10]$
PGPE-IGE	$\sigma_{init} = [0.1, 0.2, 0.9]$ $\alpha_\mu = 0.1$ $\alpha_\sigma = 0.01$
EPHE-IGE	$\sigma_{init} = [0.2, 0.3, 4]$
FM-IGE	$\theta = [0.5, 1, 10]$

Figure 3.14: The learning parameters for each algorithm. The learning rates and exploration parameters change over time (a, b). EPHE-IGE and PGPE-IGE start to learn their γ-mapping after the modules finished learning at 3000 episodes. All CA-IGE algorithms use the same set of 45 γ-modules (c). The remaining parameters of each algorithm are listed in Table (e). Figure (d) shows the γ-map of the FM-IGE and for the initial mean μ_{init} of the policy parameter distribution of the EPHE and PGPE-IGE.

Classical Q-learning needed the longest time for exploration and before its learning rate parameter α could be decreased (Fig. 3.14, a, b). Its learning rate α was set to 1 for 6000 episodes and then linearily decreased to 0.2 until episode 20,000. The exploration rate ϵ was kept to 1 for 10,000 episodes. It was then linearly reduced to 0.3 until episode 20,000.

The mean of the policy parameter distribution for the EPHE-IGE and the PGPE-IGE were initialized to $\mu_{init} = [0.5, 1, 10]$. The PGPE-IGE used as initial variance for the policy parameter distribution $\sigma_{init} = [0.1, 0.2, 0.9]$. The EPHE-IGE needed a larger initial variance of $\sigma_{init} = [0.2, 0.3, 4]$. Both algorithms sampled 10 policy parameters per iteration and evaluated each on 20 episodes. The step size parameters of the PGPE-IGE were set to $\alpha_\mu = 0.1$ and $\alpha_\sigma = 0.01$. The discount factor for the γ-map of the Q-IGE was set to $\gamma_G = 1$.

Besides the IGE algorithms, which learned the γ-map, a fixed mapping IGE (FM-IGE) was also evaluated. Its γ-map was fixed, but it still needed to learn the values of its modules. The γ-map had the same sigmoidal form as the PGPE-IGE and the EPHE-IGE

(Eq. 3.13). The parameters $\theta = [0.5, 1, 10]$ of the map were set to the initial mean of the policy parameter distribution for the PGPE and EPHE-IGE. It used the same learning rate and exploration parameters for the modules as the other IGE variants.

The energy level $s^C \in \mathbb{R}$ of the agent is real-valued. All algorithms based on Q-function (Q-learning and Q-IGE) used a discretized energy level $\lfloor s^C \rfloor$ to represent the Q-function as a table.

Performance Measurement

The algorithms were evaluated by three performance measures. First, the performance of the final policy at the end of the learning. Second, the learning curve, i.e. the performance of the learned policy after a certain numbers of episodes. Third, the cumulative reward over the course of learning.

The first performance measure is the performance of the final policy. It was measured after the end of the learning at 20,000 episodes. During the measurement, the agents did not learn or explore. All agents used their greedy policies. For the policy search methods (PGPE and EPHE-IGE) the parameters of their γ-map $G(s^C, \theta)$ were set to the final mean of their policy parameter distribution: $\theta = \mu$. Each agent was evaluated for 1000 episodes. The agent's initial energy level for each episode was pseudo randomized between 1 and 20, as it was done during the learning. The final performance is the average reward $\bar{R}$ per episode over the 1000 episodes:

$$\bar{R} = \frac{1}{1000} \sum_{k=1}^{1000} \left[\sum_{t=0}^{T-1} r_{k,t} \right] ,$$

where $r_{k,t}$ is the reward in episode k from time step t.

The second performance measurement evaluated how fast the algorithms learn their policies. It measured the performance of the learned greedy policies after every 100 episodes. Similar to the measurement of the final policy, the agents did not learn or explore during the evaluation. They used their greedy policy. Each agent was evaluated for 200 episodes, each episode started with a pseudo random energy level between 1 and 20. The performance is the average reward per episode over the 200 episodes ($\frac{1}{200} \sum_{k=1}^{200} \left[\sum_{t=0}^{T-1} r_{k,t} \right]$). The resulting learn curve over episodes shows how fast the agents can learn their greedy policies.

The third performance measure is the cumulative reward collected by the agent during learning until a certain number of episodes K: $\sum_{k=1}^{K} \left[\sum_{t=0}^{T-1} r_{k,t} \right]$. The resulting curve over all episodes shows the performance of an agent during learning. This is an important measurement in the case that not only the final performance of an agent matters, but also the reward that it gathers during the learning process itself. This is in particular important for adaptive agents which need to adapt to changing environments and tasks during their life time.

3.4.3 Experimental Results

As discussed in Section 3.2.2, the reward function R^M for the modules should be invariant to changes in the context. This allows the modules to learn a consistent set policies over

all contexts. Two approaches to set the reward function R^M of the γ-modules were evaluated. The first approach is a general approach that can be used for any CMDP. The modules only learn under a subset of contexts $\Phi \subset S^C$ for which the reward function does not vary. The second approach makes use of the specific reward function of the energy foraging task (Eq. 3.15). The reward function is composed of two components. One of the components is independent of context and could be used for modules so that they can learn under all contexts. The performance of the IGE variants was better in complex environments compared to classical Q-learning for both approaches. This section introduces both approaches and discusses their experimental results.

Results for Agents Learning under Sub-Contexts

First, the results for agents that learned only under a subset $\Phi \subset S^C$ of contexts are discussed. Learning from a subset of contexts to define the reward function for the γ-modules of an CA-IGE is a general approach that can be applied to any CMDPs. The results show that the CA-IGEs outperformed classical Q-learning even with this general approach. The section first introduces the definition of the modules reward function before its results are discussed.

The γ-modules learned their Q-values only under a subset of contexts. As discussed in Section 3.2.2, the modules reward function R^M should not vary over contexts. Furthermore, long term strategies should be optimal for it because then the short term strategies will be learned by modules with stronger discounting in parallel. For the energy foraging task, the modules reward function R^M is defined by the tasks reward function R (Eq. 3.15) under the contexts that the agent has energy ($\Phi = (s^C)_{\forall s^C > 0}$):

$$R^M(s^E_{t+1}) = \begin{cases} R(s^E_{t+1}, s^C_{t+1}) & \mid \text{if } s^C_t > 0 \quad \text{(agent has energy)} \\ \emptyset & \mid \text{if } s^C_t = 0 \quad \text{(agent has no energy)} \end{cases},$$

where $\emptyset$ means that the reward function is not defined and that the γ-modules will not be updated. As a result, the γ-modules will not learn from transitions in which the agent has zero energy ($s^C = 0$). The reward function R^M allows the modules to learn a consistent set of policies, because the reward function R of the energy foraging task is consistent over all positive energy levels ($s^C > 0$). The reward function only changes if the agent reaches an energy level of zero by giving a strong negative punishment. Moreover, as long as the agent has energy, the optimal policy is the one that results in the highest energy payoff. This allows the modules to learn not only the long term strategy with the highest payoff, but also short term strategies by modules with strong discounting.

The performance of the final policy show that the CA-IGE variants that learned only under the sub contexts (Q-IGE(s), PGPE-IGE(s), EPHE-IGE(s), and FM-IGE(s)) outperformed classical Q-learning in tasks with higher complexity (Figs. 3.15 and 3.16). First, classical Q-learning (Q) is compared to the Q-IGE(s). Both had a similar performance for all five task settings in the simple MDP layout. Although Q-learning can in theory learn the optimal policy for the task, it could not find it in every experimental run. Moreover, the theoretical performance difference between the optimal policy compared to the best policy that IGE variants can find is relatively small. As a result, the performance of Q-learning and Q-IGE(s) was similar in simple task layouts.

The Q-IGE(s) outperformed Q-learning in the moderate and complex MDP layouts for the task settings 1, 2, 3, and 5. (Figs. 3.15 and 3.16). For the high movement punishment setting (Setting 4), both had a similar final performance. The results for the policy search variants (PGPE-IGE(s) and EPHE-IGE(s)) followed a similar trend. The IGE variants showed similar performance compared to Q-learning in the simple task layout for most task settings (2, 3, and 5). Whereas, in the moderate and complex task layout they could outperform Q-learning. They had a similar performance to Q-learning in the deterministic tasks (Setting 1), and a slightly worse performance in the high movement punishment tasks (Setting 3) for moderate and complex layouts. Comparing both algorithms with each other showed that the PGPE-IGE(s) slightly outperformed the EPHE-IGE(s). Compared to the Q-IGE(s), both had either a similar performance or they are outperformed by it. The worst performance had the FM-IGE(s). Its γ-modules learned the optimal policies to reach different goal-states, but the fixed mapping from energy level to the appropriate γ-module was not optimal.

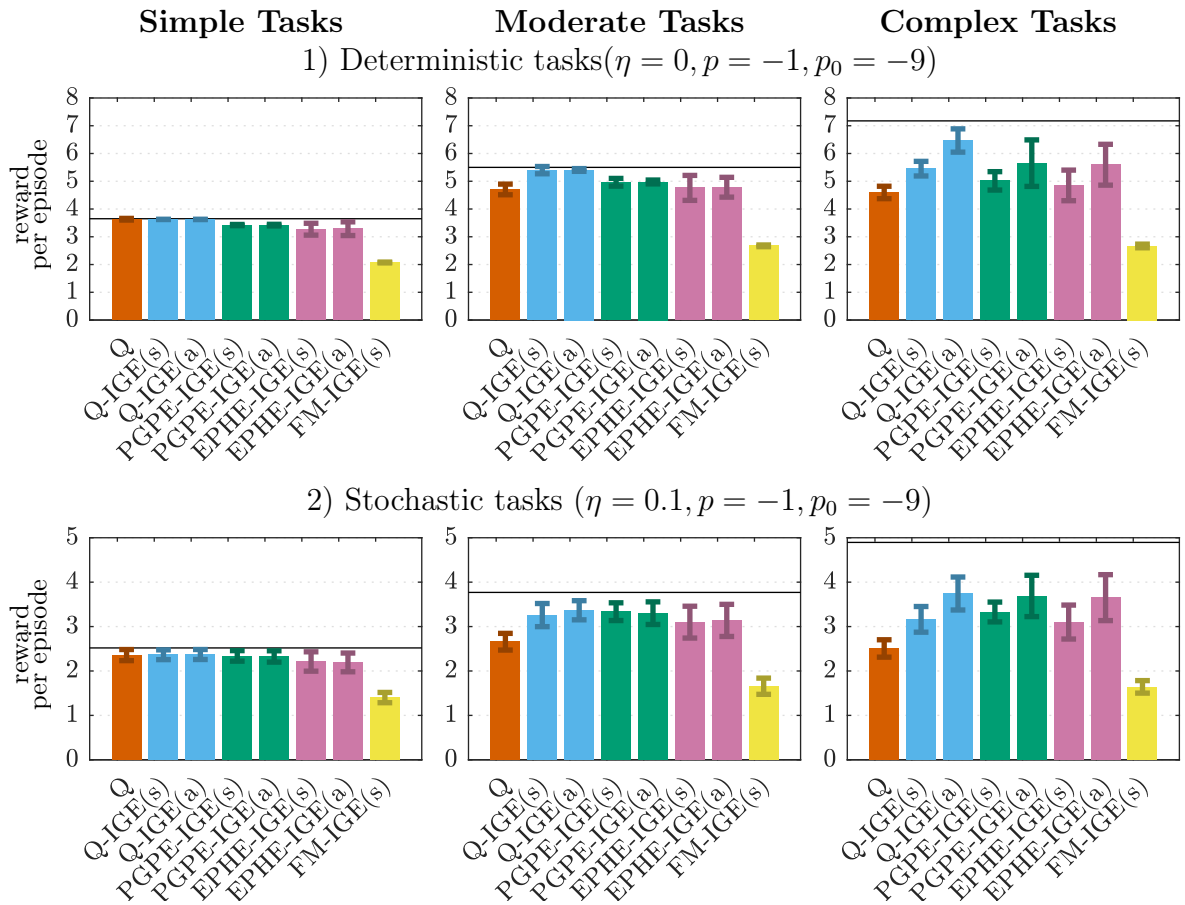


Figure 3.15: The CA-IGE variants outperform classical Q-learning (Q) in the performance of their final policies, in moderate and complex task layouts. The average reward per episode over 1000 episodes was measured. The bars show the mean of the average from over 100 independent runs. The error bar shows the standard deviation. The black line indicates the optimal possible performance in each task. IGE variants with (s) learned under sub-context, variants with (a) learned under all context.

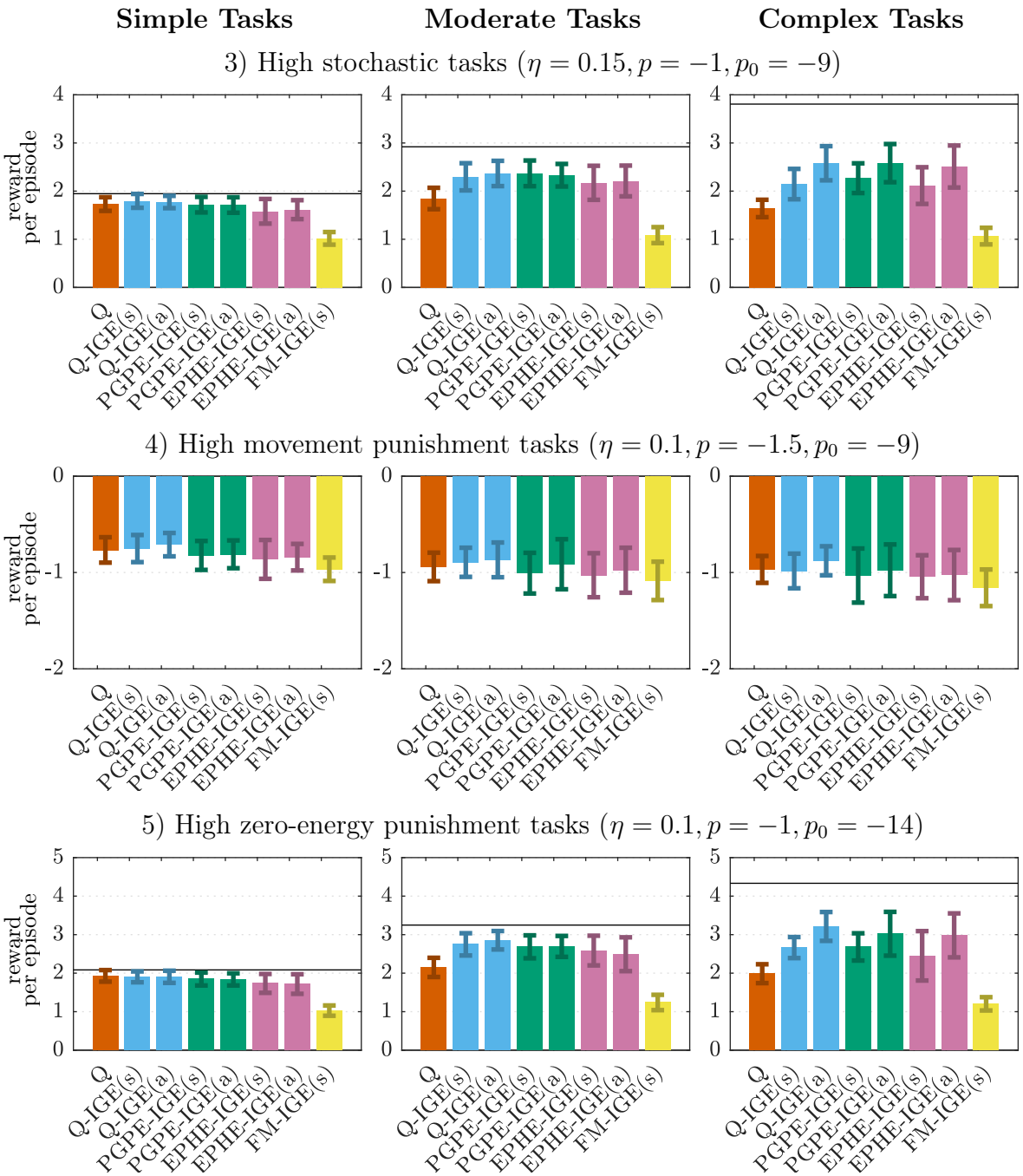


Figure 3.16: The CA-IGE variants outperform classical Q-learning (Q) in the performance of their final policies, in moderate and complex task layouts for most task settings. The performance is only similar in the high movement punishment tasks (Setting 4). The average reward per episode over 1000 episodes was measured. The bars show the mean of the average from over 100 independent runs. The error bar shows the standard deviation. The black line indicates the optimal possible performance in each task. IGE variants with (s) learned under sub-context, variants with (a) learned under all context.

In summary, in terms of performance of the final policy after 20,000 episodes, Q-learning had a similar performance to the IGE variants in the simple task layout. But, for the moderate and complex task layout, Q-learning was outperformed by the IGE-variants, in particular by Q-IGE(s).

The next performance measure is the learning curves of the learned policies over episodes (Fig. 3.17). The performance in the deterministic (Setting 1) and the stochastic (Setting 2) tasks exemplify the general trend of all results. Fig. A.1 under Appendix A shows the results for the other task settings.

First, Q-learning is compared to the Q-IGE(s). In the deterministic task (Setting 1), Q-learning initially resulted in a faster learning compared to Q-IGE(s). In the simple layout, Q-learning could retain its better learning rate compared to Q-IGE(s). In the moderate and complex layout, Q-IGE(s) overtook Q-learning after ∼3000 episodes. For stochastic tasks, Q-learning had only for the first ∼2500 episodes a better performance, before it was overtaken by the Q-IGE(s). The dominance of Q-IGE(s) over Q-learning

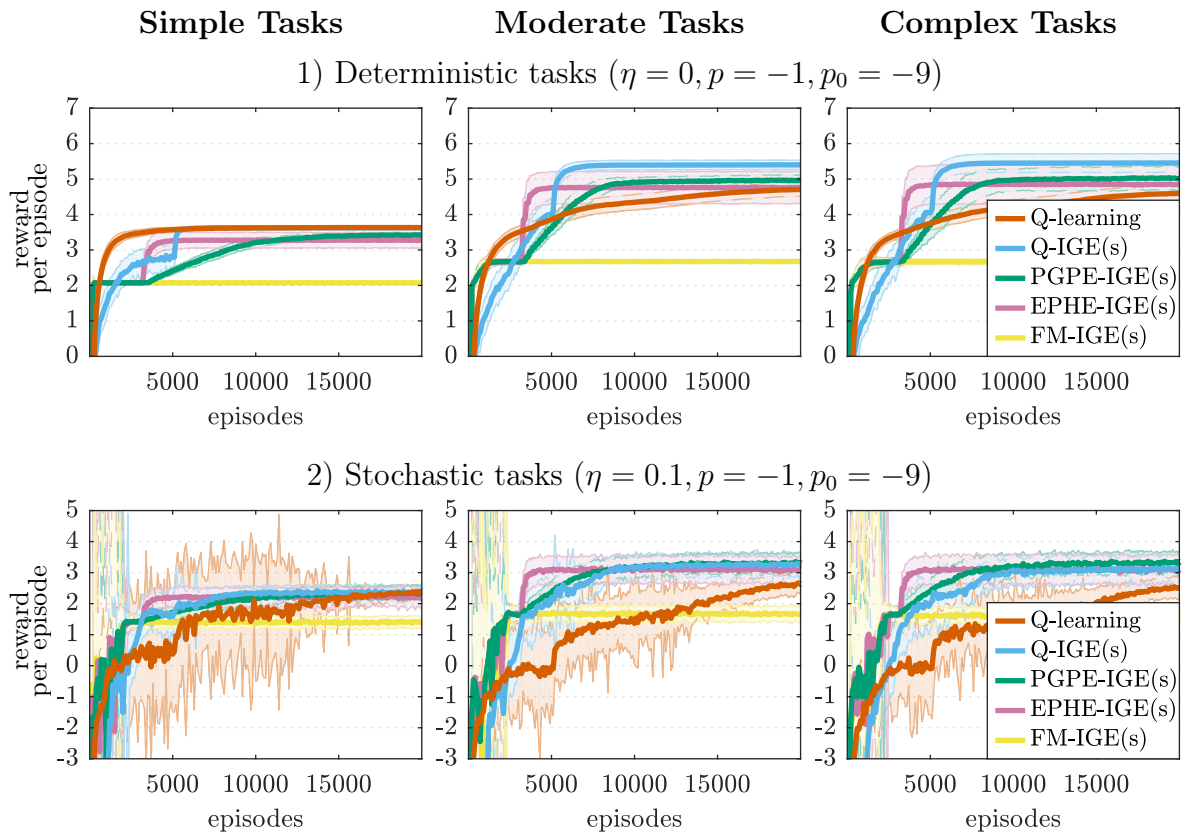


Figure 3.17: The CA-IGE variants had a better learning rate compared to classical Q-learning. Q-learning was only better in the simple, deterministic task. The average reward per episode was measured for the greedy policy after every 100 episodes of learning. The greedy policy was evaluated for 200 episodes. Learning was stopped during the evaluation. The lines show the mean of the average over 100 independent runs. The shaded area shows the standard deviation. Only the CA-IGE variants that learn their modules values under a sub-context are shown.

was stronger in the moderate and complex environment.

The PGPE-IGE(s) and the EPHE-IGE(s) initially had a better learning rate compared to Q-learning because they only learned values for their γ-modules for the first 3000 episodes. They had no learning or exploration over their initial γ-map. The fixed map is not optimal, but it initially had a good performance compared to the Q-learning algorithms that had to learn the best policy for each context from the beginning. In the deterministic tasks (Setting 1), Q-learning could overtake both policy search variants after $\sim$1000 episodes. In the moderate and complex setting Q-learning was then overtaken again after $\sim 3000 - 5000$ episodes. Comparing PGPE with EPHE-IGE(s), the EPHE-IGE(s) had initially a faster learning rate. EPHE-IGE(s) could reach its asymptotic performance after $\sim$4000 episodes. PGPE-IGE(s) needed up to $\sim$8000 episodes, but reached a better final performance. The inital learning rate of both is slightly better compared to the Q-IGE(s) in all stochastic environments.

The FM-IGE(s) had initially the same learning rate as the PGPE-IGE(s) and the EPHE-IGE(s), but it stopped improving after 3000 episodes because then the modules stopped learning.

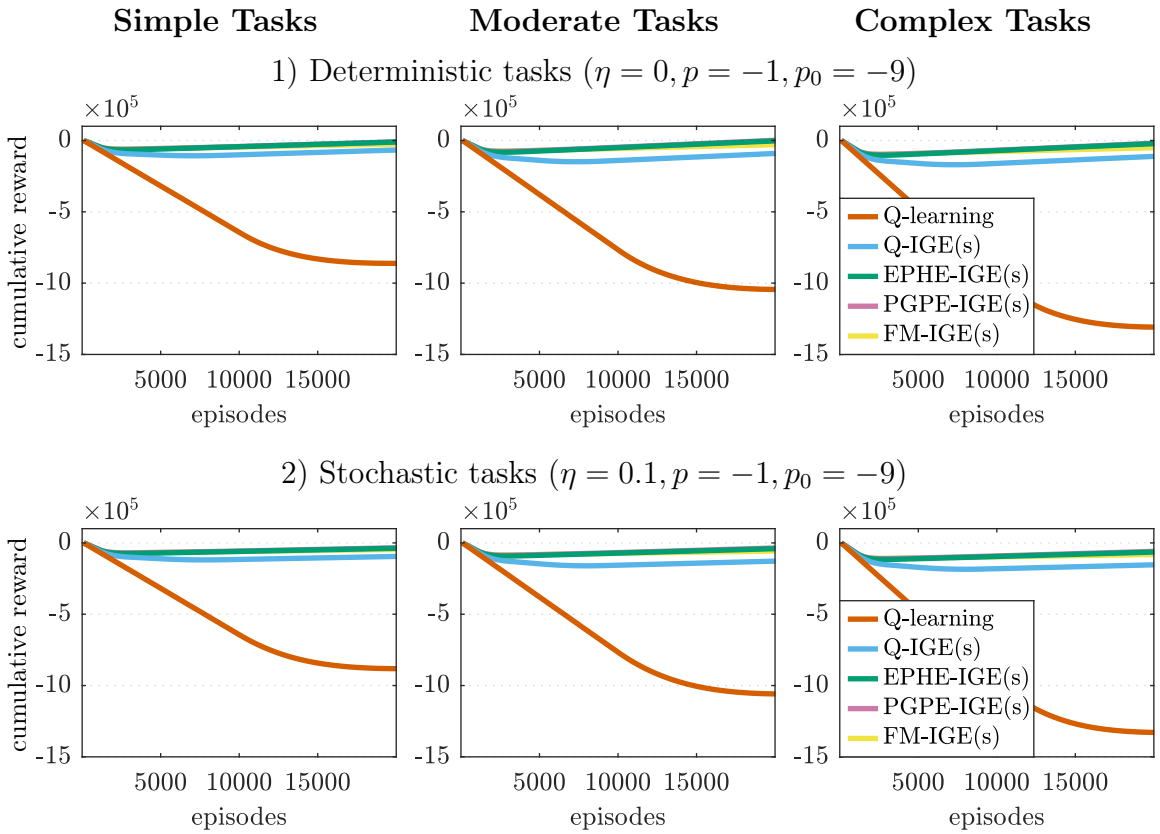


Figure 3.18: The CA-IGE variants had a better cumulative reward compared to classical Q-learning for all tasks. The cumulative reward was measured over the course of learning. The lines show the mean of the cumulative reward over 100 independent runs. The standard deviations are too small to be plotted. Only the CA-IGE variants that learn their modules values under a sub-context are shown.

In summary, Q-IGE(s) had generally a better asymptotic performance compared to PGPE-IGE(s) and EPHE-IGE(s), but their learning rate was better. Moreover, EPHE-IGE(s) had a faster learning rate compared to PGPE-IGE(s), but also a lower asymptotic performance. The IGE variants seem to have a trade-off between learning speed and asymptotic performance. Nonetheless, all IGE variants had generally a better learning rate compared to Q-learning, except when the task is simple and deterministic.

The third performance measure evaluates the cumulative reward over the course of learning. The results for the deterministic (Setting 1) and the stochastic (Setting 2) tasks show a large difference between the IGE variants and classical Q-learning (Fig. 3.18). The results are similar for all other task settings (Fig. A.4 in Appendix A). Q-learning had the poorest performance in all tasks. For example, in the deterministic task (Setting 1) with a simple layout, the cumulative reward after 20,000 episodes was approximately −860,000 for Q-learning. In contrast, the Q-IGE(s) had −66,000, the FM-IGE(s) had −25,000, the EPHE-IGE(s) had −10,500, and the PGPE-IGE(s) had −9,200. For all task settings the difference between the IGE variants and classical Q-learning was very strong. This is the result of Q-learning's need for a strong exploration of each task context, i.e. energy level, to learn a relatively good final policy. Due to the exploration, the agent often reached a energy level of zero and received strong punishment. The IGE-variants did not need so much exploration.

In summary, the experimental results show a clear difference between classical Q-learning and the CA-IGE variants. In theory, Q-learning is guaranteed to converge to the optimal policy for each task, whereas the IGE algorithms cannot guarantee this. However, the IGE algorithms outperformed Q-learning in nearly all tasks and for the others, they had a similar performance. In tasks with the simple layout, Q-learning and the CA-IGE variants learned policies with a similar final performance. If the task complexity increases, the CA-IGE variants generally outperformed Q-learning. Besides the performance of the final policy, the CA-IGE variants also showed a better learning rate in stochastic settings and especially moderate and complex task layouts. In particular, the cumulative reward showed a very large difference between Q-learning and the CA-IGE algorithms among the performance measures.

Comparing the CA-IGE algorithms with each other shows that they had a trade-off between learning rate and asymptotic performance. For many settings, the Q-IGE(s) had the best asymptotic performance, followed by the PGPE-IGE(s) and then the EPHE-IGE(s). Nonetheless, the learning rate was generally better for the EPHE-IGE(s), followed by the PGPE-IGE(s) and then the Q-IGE(s).

Results for Agents Learning under all Contexts

The previous section discussed the results for CA-IGE variants where the γ-modules are learned based on a subset $\Phi \subset S^C$ of contexts, i.e. non-zero energy levels ($s^C > 0$). A problem with this approach is that the γ-modules do not learn from transitions that are not under these contexts. This results in a higher demand for observations to learn an appropriate policy. Depending on the algorithm, and the task, the number of used observations over the course of the 20,000 learning episode lay between 80 to 95 percent (Fig. 3.19, and Fig. A.7 of Appendix A). For the energy foraging task it is possible to use all observations by defining the modules reward function R^M in a different way. The

experimental results of this approach showed that it outperformed agents that learned their γ-modules only from observations with non-zero energy levels.

The approach that uses observations of all contexts to learn γ-modules uses the property that the reward function of the energy foraging task is a linear combination of two components (Eq. 3.15). The first component (R^P) depends only on the task state s^E_{t+1} to which the agent transitions. It defines the punishment p that the agent receives for each transition and the reward r_G it receives for reaching an energy source s_G. It is independent from context. Therefore, it can be used as the reward function R^M for the γ-modules:

$$R^M(s^E_{t+1}) = R^P(s^E_{t+1}) \ .$$

This allows the agent to update its Q-values after each transition. The reward function allows the agent to also learn all important strategies in the task. It does not punish trajectories to energy sources that are far away, because it does not consider the energy level of the agent. Thus, the modules learn for large γ's long term policies that give a high payoff. The modules with smaller γ's can learn in parallel the short term strategies.

All CA-IGE variants (Q-IGE(a), PGPE-IGE(a), EPHE-IGE(a), and FM-IGE(a)) were evaluated with this approach. They were compared to the variants that update the

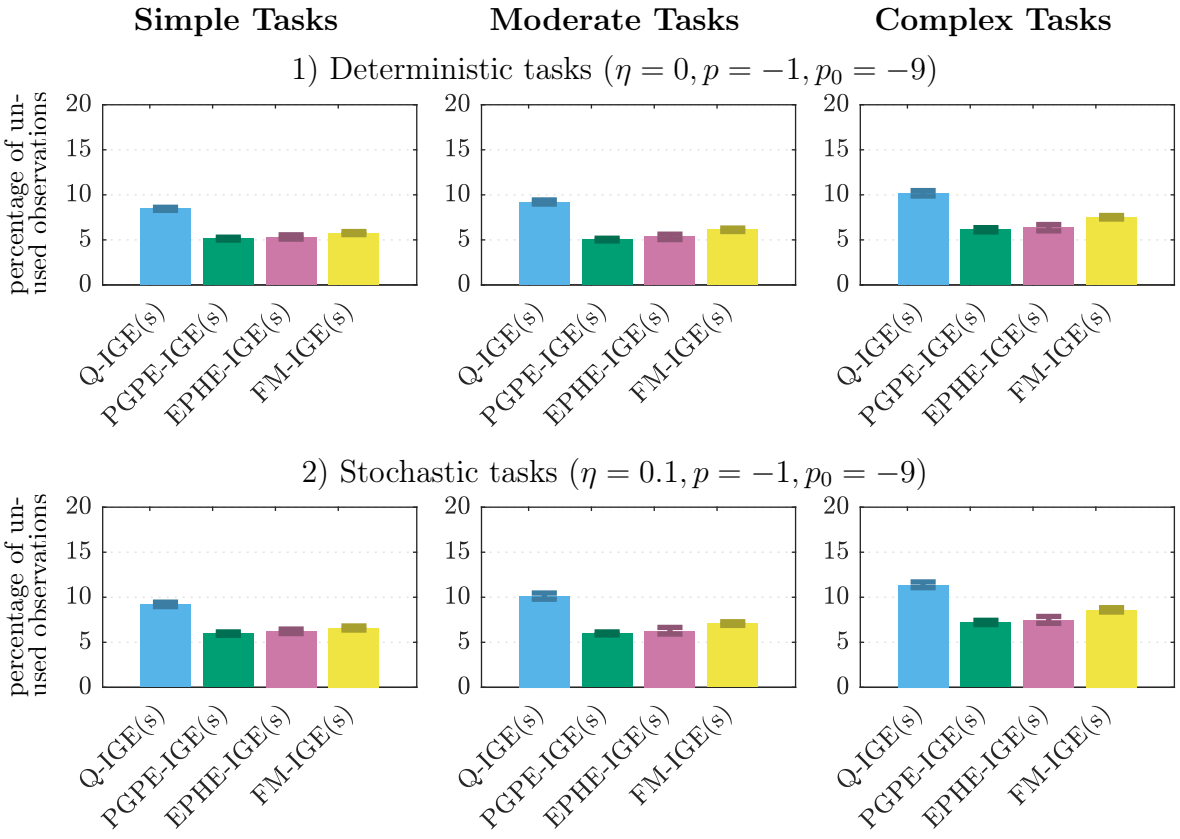


Figure 3.19: The approach of learning values of γ-modules only for non-zero energy levels ($s^C > 0$) can not use between 5 to 15 percent of the observations. The bars show the mean percentage of observations that could be used to update the modules over 100 independent runs. The error bar shows the standard deviation.

modules Q-values only under non-zero energy levels (Q-IGE(s), PGPE-IGE(s), EPHE-IGE(s), and FM-IGE(s)). The identifier (a) stands for "all contexts" and (s) for "sub-contexts". First, the algorithms are compared based on the performance of their final policy (Figs. 3.15 and 3.16). For the simple and the moderate task layouts, learning under all contexts gave a similar performance for all CA-IGE variants compared to learning under sub-contexts. For some CA-IGE variants and task settings a small improvement existed. Nonetheless, a significant improvement existed for each CA-IGE variant in the complex task layouts for the Settings 1, 2, 3, and 5. Only the high movement punishment tasks (Setting 4) showed no improvement. The learning curves and the cumulative rewards followed the same tendency (Appendix A).

In summary, the componentwise structure of the reward function for the energy foraging task can be exploited to allow the γ-modules to learn from each observation, regardless of the current context. This increased the performance of the CA-IGE algorithms, especially for the complex task layouts. As a result, the CA-IGE variants could further improve their advantage over classical Q-learning.

3.4.4 Discussion

The results for the foraging energy task showed that CA-IGE algorithms can outperform classical methods such as Q-learning. Although Q-learning is theoretically guaranteed to converge to the optimal policy, in practice, the non-optimal CA-IGE variants yielded either a similar or a better asymptotic performance and they had a faster learning rate. Only in simple and deterministic tasks could classical Q-learning outperform the CA-IGE variants. Although these are striking results, their are some limitations and critical points that have to be considered in regard to the application CA-IGE algorithms.

Missing Theoretical Background about Optimal Application Scenarios

One critical point is that the CA-IGE cannot guarantee to be better, compared to classical algorithms, for every possible task. The energy foraging task has some advantageous features that allow the CA-IGE to outperform classical algorithms. It has a trade-off between time and reward strength for different possible strategies. This allows the CA-IGE to learn the different strategies and to apply them under different contexts.

Moreover, the reward function of the task allowed the γ-modules to learn a consistent set of policies with few observations. A sub-context approach for the module's reward function R^M could be successfully used by learning only under non-zero energy levels. Nonetheless, the agents could still use 80 to 95 percent of all observations to update the γ-modules. The reward function could be decomposed into a component that is consistent over all contexts, making it possible to update the γ-modules from all observations.

Other tasks might not have such features. For example, the γ-modules might only be able to learn under a small set of sub-contexts. This would increase the number of total observations that the CA-IGE needs to learn its modules policies. As a result, the CA-IGE could loose its advantage in learning speed over classical algorithms. At the moment, the theoretical background does not exist to evaluate for which tasks the CA-IGE has an advantage over classical methods.

The Q-IGE

The Q-IGE had generally the best asymptotic performance in the energy foraging tasks. Q-learning had a similar performance only for simple task settings. The Q-IGE also outperformed Q-learning in the learning rate for more complex task settings. Nonetheless, the PGPE-IGE and the EPHE-IGE performed better in terms of the learning rate.

The Q-IGE learned its γ-map $G(s^E, s^C)$ over the task and the context state space. The policy search variants define the γ-map only over the context state space S^C. This could be the reason why the Q-IGE has a lower learning rate compared to the PGPE-IGE and the EPHE-IGE. Q-IGE has to learn the optimal strategy for each task and context state pair, instead of only for contexts. Experiments with a γ-mapping $G(s^C)$ only over the context states showed that the Q-IGE was not able to learn the task. Its performance was significantly worse compared to classical Q-learning and the other CA-IGE variants.

The problem of learning the mapping only over context states seems to be that the Q-values have a high variance. The values are the expected future reward sum if module γ is used under context s^C: $Q^G(s_t^C, \gamma_t) = \mathrm{E}_{S^E}\left[\sum_{t=0}^{T-1} \gamma_G^t r_t\right]$. The variance of this measurement can be very high, because the reward depends strongly on the agents position s^E in the grid world. For example, an agent could be far away from a goal state that would be reached by using module $\hat{\gamma}$ under context $\hat{s}^C$. Because of the punishment per movement, the reward sum might be low. Whereas, if the agent is nearby the goal state, then the reward sum is higher. As a result, the variance over the reward sum is high, making it difficult for the Q-values to converge. Q-values over the task and context space $Q^G(s_t^E, s_t^C, \gamma_t)$ do not have this problem and can converge.

Although Q-IGE has to learn the map over task and context states it still performed better compared to classical Q-learning. The reason seems to be that Q-IGE does not learn which individual actions are optimal for each context like classical Q-learning. Instead, it learns which module's policy is optimal and the policies of the modules already provide the optimal path to one of several goal states. Thus, the Q-IGE learns over macro actions, i.e. which goal state to reach under which context, and not over micro actions, i.e. in which direction to walk under which context. This reduces the number of strategies the Q-IGE has to explore to find a good mapping.

The PGPE-IGE and the EPHE-IGE

The PGPE-IGE and the EPHE-IGE had a lower asymptotic performance compared to the Q-IGE, but their learning rates were generally better. The PGPE-IGE had a lower learn rate compared to the EPHE-IGE, but also a higher asymptotic performance.

An important advantage of the EPHE over the PGPE is that it has fewer learning parameters. For the PGPE, the tuning of the step sizes α_μ and α_σ were crucial to reach a good performance. The EPHE does not have such learning parameters, making it easier to apply to a problem.

Although, the PGPE-IGE and the EPHE-IGE could outperform classical Q-learning in the complex task settings of the energy foraging task, they have some limitations. As many other policy search methods, the PGPE-IGE and the EPHE-IGE evaluate the performance of sampled parameters $\theta^i \sim \mathcal{N}(\mu, I\sigma^2)$ for their γ-map $G(s^C, \theta)$ (Eq. 3.13) by the cumulative reward over whole trajectories. The performance is the reward sum

that a γ-map generates during a trajectory. In CMDPs, the reward of trajectories depend strongly on the context under which they are executed. For the energy foraging task, the parameters θ of the γ-map should be evaluated for different start energy levels of the agent. The experimental design allows this, because for every block of 20 episodes each start energy level between 1 and 20 is used once. This guarantees that the γ-map can be evaluated under all possible contexts. For other tasks, this might not be possible. For example, the agent could start for many episodes with a high energy level. Under this circumstance, the PGPE-IGE and EPHE-IGE learn good parameters θ_A for this context. If afterward, the initial energy level is very low for many episodes, then the algorithms might be able to adapt their γ-map parameters to them, but the new parameters θ_B will only be optimized for low energy levels. Thus, if the context switches back to higher energy levels, the parameter θ_B yields again a low performance. The Q-IGE does not have this problem. It learns separate Q-values for every context. Values learned for one context are not discarded if the agent is in a different context for a longer time. In conclusion, the usage of the PGPE and EPHE to learn the γ-map is only useful in tasks where agents can evaluate γ-maps over all possible contexts in short time windows.

Another critical point if comparing the policy search methods (PGPE-IGE and EPHE-IGE) to the value-based methods (Q-learning and Q-IGE) is that the form of their γ-map includes already a priori information about the task. Their γ-map is a sigmoidal function (Eq. 3.13) over the contexts, i.e. the energy levels. The sigmoidal function already entails the information that similar discounting parameters γ's are good for similar energy levels. This is true for the energy foraging task. Moreover, the form of a sigmoid fits well with the assumption that for lower energy levels short term strategies are better and, vice versa, that for higher energy levels long term strategies are better. For other tasks, such assumptions might not be possible. The γ-map could be changed to include no a priori information about the task. For example, for each context s^C it could have a parameter $\theta_{S^C} \in [0, 1]$ which defines directly the optimal discount factor with $G(s^C, \theta) = \theta_{s^C}$. Experiments with such γ-maps resulted in a low learning performance below the level of classical Q-learning. The potential reason is that the number of γ-map parameters increased for these maps from 3 (Eq. 3.13) to 20 which made their optimization harder. In summary, the γ-map of the PGPE-IGE and EPHE-IGE needs a reasonable form to allow a good learning performance.

3.5 Conclusion

The IGE provides a framework to adapt quickly to changes in the context of tasks. The γ-modules of an IGE learn different policies for a task that represent different solutions for a trade-off between gained reward and invested time. Modules with weak discounting (large γ's) learn policies resulting in large reward sums, but that might take longer to reach. Strong discounting (small γ's) result in policies that prefer short term strategies with less reward. The IGE learns these different policies in parallel from the same observations.

This ability of the IGE is useful for tasks where the context defines the optimal trade-off between reward and time. In such tasks, the IGE allows rapid adaptation to different contexts. For example, an agent might have the task of foraging for food. The task has a trade-off between food quality and the time needed to reach some food dependent on the

hunger level the agent. If the agent is well feed, it can use a long-term strategy that result in food of high quality, i.e. a high reward value. Whereas, if the agent is very hungry, it will prefer more short-term strategies that result in any food, even of lower quality. The IGE learns long-term and short-term strategies at the same time. If the agent learns the task while being well-fed and later is has to do the task while being hungry, the agent can immediately switch to a short-term strategy without any additional learning.

Agents adapt to different contexts by selecting a γ-module for a certain context and then following the module's policy. The selection is based on a γ-map from context to γ-modules. The IGE can only adapt well to different contexts if the γ-map is appropriate, i.e. when it provides the most appropriate module for a certain context. Two possibilities exists to find an appropriate map. First, for certain tasks a map can be analytically derived. Three task scenarios exist for which a map can be derived: 1) tasks which have constant risk of interruption, where the probability of interruption is the context, 2) tasks with a constant risk that all future reward becomes zero, where the probability of this risk is the context, and 3) tasks in which a constant punishment is given for each step, where the punishment level is the context. The other approach to find an appropriate γ-map is to learn it. The learning problem of a map is itself an MDP. Therefore, standard reinforcement methods can be applied such as Q-learning or policy search methods.

An open question is for which tasks the IGE provides an advantage over classical algorithms such as Q-learning. In terms of optimality, the IGE can guarantee to converge to the optimal policy only for the three tasks for which a γ-map can be derived. In these tasks, the IGE has, in contrast to classical methods, the ability to adapt immediately to changes in context and therefore is better compared to them. For tasks, where the γ-map has to be learned, the IGE cannot guarantee to be optimal. Nonetheless, because of its adaptive abilities it can outperform classical methods in terms of learning rate. In practice, this can lead to a better final performance of the IGE's policies compared to classical methods. The theoretical background is currently missing to determine for which tasks this advantage exists.

Chapter 4

Adaptation to Changing Objectives

The IGE has the ability to immediately adapt to changing objectives of a task that define different trade-offs between the amount of reward that should be gained and the invested time. One of our daily routines, going out for lunch, is is an example of such tasks. Several restaurants exist in our neighborhood. Each requires a different amount of time to reach and provides food of different quality. The general goal is to learn the shortest way to a good restaurant, but our specific objective might change from day to day. One day we might want to go to the best restaurant. Another day we are under stress and we have to judge between the restaurant quality and the time to get there. Another day we want to go to the best restaurant given a fixed time limit.

In principle, such tasks can be formalized as episodic MDPs (S, A, T, R). The state space S is our position. The actions A are the directions in which we can go. A transition function $T(s, a, s')$ defines what position s' can be reached from position s with action a in a fixed time, e.g. $\Delta t = 30$ seconds. Each restaurant is a goal state $s \in S^G$. Reaching and eating in a restaurant provides a reward $R(s)$ reflecting its food quality. The problematic point is how different objectives regarding the trade-off between food quality and invested time can be implemented.

A standard approach would be to use a different reward function R for each objective. For example, in the case of learning the way to the best restaurant, the reward function of reaching a restaurant returns simply its food quality. An agent will learn the shortest way to the best restaurant with this reward function. If the agent is under stress and has to judge between food quality an invested time, the reward function could give a punishment, a negative reward, for each performed step until a restaurant is reached. Thus the agent might not go to the best restaurant, but to one that is closer. In the case of a fixed time limit, the reward function could return a punishment if a restaurant is reached with more steps than the limit. With this approach, each objective represents a different task for which a new policy has to be learned. But learning takes time, and the objectives might change from one episode to another. Also, the number of possible objectives can be infinite. Thus, a classical agent might not have time to learn an appropriate policy for each objective.

The IGE can overcome this problem. It has the ability to learn policies that represent different trade-offs between rewards and time invested to obtain them (Section 2.2). Moreover, the IGE can decode information about the expected reward and the needed time of its policies (Section 2.3). Based on this information, the IGE can immediately

select an appropriate policy for a new objective. It does not need to learn the task for each objective from scratch, like classical model-free methods.

Model-based methods also provide a solution to the problem of adapting rapidly to new objectives. They learn a model of the external environment that can be used to find the best policy for a new objective. Nonetheless, they can be computationally demanding. Furthermore, learning a correct model can be very difficult for stochastic tasks with high state dimensions.

This chapter has the following structure. Section 4.1 introduces the Objective Adaptive MDP (OA-MDP) used to formalize tasks with the problem of adapting to different objectives. Afterward, the IGE framework used to solve OA-MDPs, the Objective Adaptive IGE (OA-IGE), is described in Section 4.2. Two variants of the OA-IGE exist that differ in the way they compute information necessary to select the most appropriate γ-policy for an objective. The first variant, introduced in Section 4.3, decodes the expected future reward trajectory from the Q-values of the γ-modules. The second variant, introduced in Section 4.4, learns estimations about the outcomes of the different γ-policies.

4.1 Objective Adaptive Markov Decision Processes

Tasks that should be solved under different objectives are formulated as Objective Adaptive MDPs.

Definition 9. An Objective Adaptive Markov Decision Process (OA-MDP) is an MDP with a finite or infinite set of objective functions $J = (f_1, f_2, \ldots)$:

$$\mathrm{OA-MDP}\,(A, S, T, R, J) \ ,$$

where A is a finite set of actions, S is a finite set of states, $T : S \times A \times S \mapsto [0, 1]$ is a transition probability function and $R : S \times A \times S \mapsto \mathbb{R}$ is a reward function. Objective functions evaluate reward trajectories: $h = (r_1, r_2, \ldots, r_T) \in H$. Objectives either take into account the full trajectory ($f_k : H \mapsto \mathbb{R}$) or the sum over the rewards and the number of steps ($f_k : \mathbb{R} \times \mathbb{N} \mapsto \mathbb{R}$) of a trajectory with $f_k(R = \sum_{t=0}^{T-1} h_t, T)$. At each episode, one of the objectives is active. The goal is to maximize the expectation over the active objective function k, while using a minimum expected number of steps:

$$\min_{\mathrm{E}[T]} \max_{\pi_k} \mathrm{E}_{\pi_k}\left[f_k(H)\right] \ , \tag{4.1}$$

where π_k is the policy used for objective f_k, and H is a random variable over trajectories generated by π_k.

The restaurant example from the introduction can be formalized as an OA-MDP. The agent's possible positions are the state space S. Its actions A are the different movement directions and the transition function $T(s, a, s')$ defines their outcomes. Restaurants are goal states $s \in S^G$. The reward function $R(s)$ indicates the food quality of restaurants after reaching them. The three objectives that the agent should fulfill are represented as objective functions: $J = (f_1, f_2, f_3, \ldots)$. The first objective is to go to the restaurant with highest food quality. Its objective function is $f_1(h) = \sum_{t=0}^{T-1} r_t$. The second objective makes a judgment between the time to reach a restaurant and its food quality. A possible

formulation is a linear combination of both pieces of information: $f_2(h) = \sum_{t=0}^{T-1} r_t - \frac{1}{2}T$. The third objective sets a strict time limit of five steps:

$$f_3(h) = \begin{cases} \sum_{t=0}^{T-1} r_t & | \; T \leq 5 \\ -10 & | \; T > 5 \end{cases},$$

where a violation is strongly punished with a reward of -10. Depending on the active objective, the agent has to change its policy, i.e. to go to a different restaurant, to maximize it.

Objective functions can be defined in two forms. The first form $f(h)$ uses the full reward trajectory h as input. It can evaluate a trajectory based on the exact time when a reward was received. This makes this formulation very flexible. For example, the objective could be to obtain in the first 5 steps at least a reward sum of 10 and then to maximize reward until the end of the episode. Theoretically, the IGE can be used to solve objectives with this form, but practically it is restricted to the subset of deterministic, goal-only-reward tasks. However, many practical objectives can be represented by the second form for objectives $f(R, T)$ that takes into account only the reward sum $R = \sum_{t=0}^{T-1} r_t$ and the number of steps T of a trajectory. For example, all three objectives of the lunch example (f_1, f_2, f_3) can be expressed in this form. The IGE can be applied to any OA-MDP that uses this form.

The major challenge in OA-MDPs is that the number of objective functions can be infinite such that each episode can have a different objective. Thus, an agent might experience a certain objective only a few times or even only once. As a result, it might not have enough time to learn a dedicated optimal policy for each objective function. Instead, given a new objective it has to adapt immediately to it.

4.2 The Objective Adaptive Independent γ-Ensemble

The Objective Adaptive Independent γ-Ensemble (OA-IGE) is an IGE framework that allows immediate adaptation to a new objective using information gained from previous experience under different objectives. It uses the ability of the IGE to learn a set of policies by its γ-modules. The policies represent different solutions for the trade-off between their resulting reward and invested time. The OA-IGE tries to select one of its γ-policies that maximizes the given objective function (Fig. 4.1). Because objectives may change from one episode to another and may not repeat, the OA-IGE tries to adapt immediately to a given objective function without actually experiencing how its policies perform for a new objective.

The goal of the OA-IGE is to identify the module $\tilde{\gamma}$ that has the policy $\pi_{\tilde{\gamma}}$ maximizing the objective function f_k:

$$\tilde{\gamma} = \underset{\gamma}{\operatorname{argmax}} \, \mathrm{E}_{\pi_\gamma, s_0} \left[f_k(H) \right] \quad \text{or} \quad \tilde{\gamma} = \underset{\gamma}{\operatorname{argmax}} \, \mathrm{E}_{\pi_\gamma, s_0} \left[f_k(R, T) \right] \; ,$$

where H is the random variable over the trajectories, and R and T over the reward sum and number of steps for policy π_γ starting in state s_0. If several modules optimize the objective, then a module resulting in the shortest number of steps T among these is selected. Using the policy of module $\tilde{\gamma}$ would maximize the active objective (Eq. 4.1).

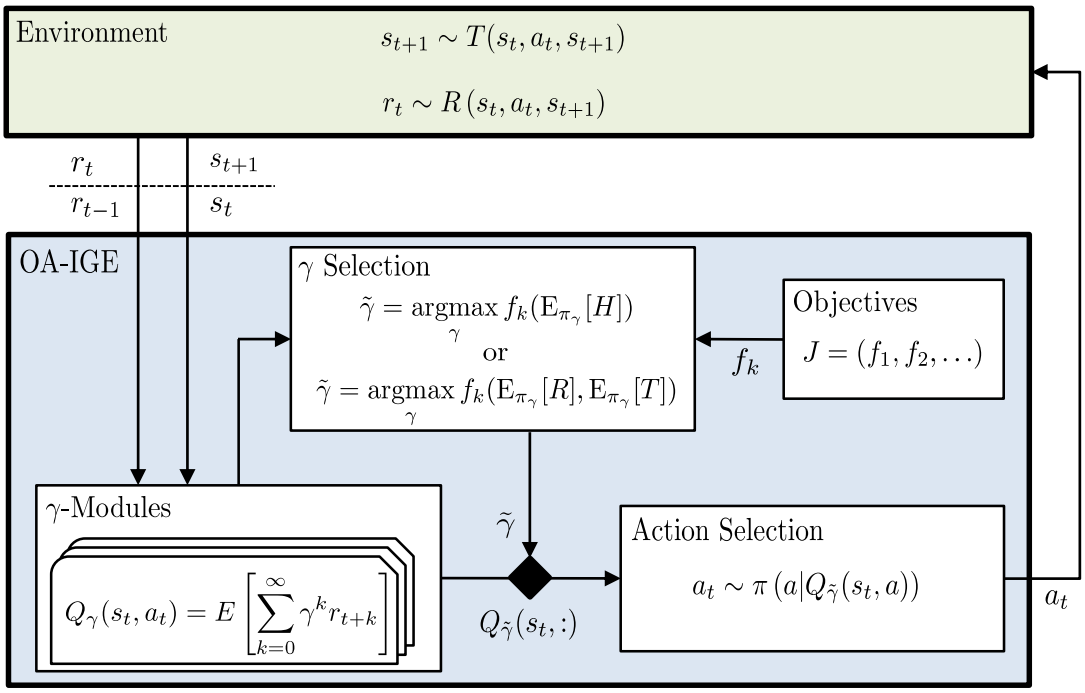


Figure 4.1: The general framework of the Objective Adaptive IGE (OA-IGE). The goal of the agent is to maximize the active objective f_k from the set of infinite objectives J. Objective functions evaluate either full trajectories $f_k(h)$ or the reward sum r and length t of a trajectory $f(r, t)$. The agent selects the active γ-module $\tilde{\gamma}$ based on the approximated outcome of each modules policy for the active objective. Approximations are computed based on either a decoded expected trajectory (Section 4.3) or a learned prediction of the expected reward sum and length (Section 4.4).

However, the OA-IGE has no access to the true expectation over an unseen objective function. Instead, the OA-IGE approximates it using the expectation over the trajectories or their reward sum and number of steps:

$$\mathrm{E}_{\pi_\gamma, s_0}[f_k(H)] \approx f_k(\mathrm{E}_{\pi_\gamma, s_0}[H]) \quad \text{or} \quad \mathrm{E}_{\pi_\gamma, s_0}[f_k(R, T)] \approx f_k(\mathrm{E}_{\pi_\gamma, s_0}[R], \mathrm{E}_{\pi_\gamma, s_0}[T]) \ . \tag{4.2}$$

The selection of the active γ-module using the approximation becomes:

$$\tilde{\gamma} = \underset{\gamma}{\text{argmax}}\, f_k(\mathrm{E}_{\pi_\gamma, s_0}[H]) \quad \text{or} \quad \tilde{\gamma} = \underset{\gamma}{\text{argmax}}\, f_k(\mathrm{E}_{\pi_\gamma, s_0}[R], \mathrm{E}_{\pi_\gamma, s_0}[T]) \ .$$

If several modules maximize the objective, on with the shortest expected steps is selected.

If the approximation of the outcome for the objective function is correct, i.e. $\mathrm{E}[f_k(H)] = f_k(\mathrm{E}[H])$, then the OA-IGE selects the most appropriate policy for the objective function. However, correctness cannot be guaranteed for every task or objective function. Nonetheless, the use of the approximation allows the OA-IGE to adapt immediately to a new objective function without experiencing how its policies perform. Experimental results in Sections 4.3 and 4.4 show that the approximation yields a practical advantage over classical algorithms.

The next part describes some cases in which the correctness of the approximation can

be guaranteed. It is followed by a description of how the expectation over a trajectory or its reward sum and length are computed by the OA-IGE.

Correctness of the Approximation

The correctness of the approximation (Eq. 4.2), i.e. that $\mathrm{E}[f_k(H)] = f_k(\mathrm{E}[H])$ can be guaranteed for a subset of tasks and objective functions. The following three cases exemplify tasks and objective functions for which the approximation is correct. They do not constitute a complete list of all cases. Correctness is shown for objectives $f(h)$ that use the full reward trajectory as input. Nonetheless, the following cases apply also for objectives $f(R, N)$ that use the reward sum and the number of steps as input, because the discussed objectives use only the reward sum and number of steps to evaluate a trajectory.

In the case of tasks with a deterministic transition and reward function the approximation is correct for any objective function.

Theorem 13. *The approximation* $\mathrm{E}_{\pi_\gamma,s_0}[f(H)] \approx f(\mathrm{E}_{\pi_\gamma,s_0}[H])$ *is correct for any objective function in OA-MDPs that have deterministic state transitions and a deterministic reward function.*

Proof. In deterministic tasks the trajectory h_{π,s_0} resulting from a policy π and starting in state s_0 is the same for each episode, i.e. the random variable over trajectories H has only h_{π,s_0} as outcome. Thus, the expectation over the objective function is:

$$\mathrm{E}_{\pi,s_0}[f(H)] = f(h_{\pi,s_0}) \ .$$

This is equal to its approximation, i.e. the result of the objective function over the expected trajectory:

$$\mathrm{E}_{\pi,s_0}[H] = h_{\pi,s_0} \quad \Leftrightarrow \quad f(\mathrm{E}_{\pi,s_0}[H]) = f(h_{\pi,s_0}) \ ,$$

proving that the approximation is correct for deterministic OA-MDPs. □

In the case of general OA-MDPs that include stochasticity, the approximation for some objective functions can be proven. The objective of maximizing the reward sum is such a case.

Theorem 14. *The approximation* $\mathrm{E}_{\pi,s_0}[f(H)] \approx f(\mathrm{E}_{\pi,s_0}[H])$ *is correct for the objective function*

$$f(h) = \sum_{t=0}^{T-1} h_t \ , \tag{4.3}$$

in all OA-MDPs.

Proof. In general, the transitions and rewards in OA-MDPs can be stochastic. Therefore, a specific policy π starting in state s_0 can result in different possible trajectories $h_i \in H$, each having a different probability $\Pr(h_i)$ to occur. Therefore the expectation over the objective function is given by:

$$\begin{aligned} \mathrm{E}_{\pi,s_0}[f(H)] &= \textstyle\int_i \Pr(h^i) f(h^i) di \\ &= \textstyle\int_i \Pr(h^i) \sum_{t=0}^{T^i-1} (h_t^i) di \ , \end{aligned}$$

where T^i is the length of trajectory h^i. The approximation of this expectation is equal. It can be shown by using the sum rule in integration:

$$\begin{aligned} f(\mathrm{E}_{\pi,s_0}[H]) &= \textstyle\sum_{t=0}^{T-1} \mathrm{E}_{\pi,s_0}[H_t] \\ &= \textstyle\sum_{t=0}^{T-1} \int_i \Pr(h^i) h_t^i di \\ &= \textstyle\int_i \Pr(h^i) \sum_{t=0}^{T^i-1} (h_t^i) di \ , \end{aligned}$$

where $T = \max_i T^i$ is the maximum length of all possible trajectories and the expectation of the reward for time step t is $\mathrm{E}_\pi[H_t] = \int_i Pr(h^i) h_t^i di$. The reward for time steps that are longer than its trajectory are assumed to be zero: $\forall t \geq T^i : \ h_t^i = 0$. This shows that the approximation of the objective function to maximize the reward sum (Eq. 4.3) is correct for any OA-MDP. □

In other cases, approximations are correct for certain combinations of tasks and objective functions. For example, the objective to get the highest reward sum, but to use only a maximum number of steps can be correctly approximated in tasks that have a deterministic transition function.

Theorem 15. *The approximation* $\mathrm{E}_{\pi,s_0}[f(H)] \approx f(\mathrm{E}_{\pi,s_0}[H])$ *is correct in all OA-MDPs with a discrete transition function* T *for the objective function*

$$f(h) = \begin{cases} \sum_{t=0}^{T-1} r_t & | \ T \leq n \\ p & | \ T > n \ , \end{cases}$$

where $n \in \mathbb{N}$ *is the maximum number of steps the agent should perform and* $p \in \mathbb{R}^-$ *is a negative punishment if the agent used more steps.*

Proof. Each trajectory $h^i \in H$ for policy π starting in state s_0 has the same length: $\forall i : T^i = T$, because the task has a deterministic transition function. Therefore, the expectation over the objective function is given by:

$$\begin{aligned} \mathrm{E}_{\pi,s_0}[f(H)] &= \textstyle\int_i \Pr(h^i) f(h^i) di \\ &= \begin{cases} \int_i \Pr(h^i) \sum_{t=0}^{T-1} (h_t^i) di & | \ T \leq n \\ p & | \ T > n \ . \end{cases} \end{aligned}$$

Using the sum rule in integration, its approximation is shown to equal:

$$\begin{aligned} f(\mathrm{E}_{\pi,s_0}[H]) &= \begin{cases} \sum_{t=0}^{T-1} \mathrm{E}_{\pi,s_0}[H_t] & | \ T \leq n \\ p & | \ T > n \end{cases} = \begin{cases} \sum_{t=0}^{T-1} \int_i \Pr(h^i) h_t^i di & | \ T \leq n \\ p & | \ T > n \end{cases} \\ &= \begin{cases} \int_i \Pr(h^i) \sum_{t=0}^{T^i-1} (h_t^i) di & | \ T \leq n \\ p & | \ T > n \ , \end{cases} \end{aligned}$$

where $\mathrm{E}_{\pi,s_0}[H_t]$ is the expectation of the reward for time step t. □

In summary, the OA-IGE's approximation of the expected outcome for an objective function is correct for some task variants and objective functions. In other cases, it represent only an approximation which might result in using an inappropriate policy. Nonetheless, the approximation allows the OA-IGE to adapt immediately to a new and unseen objective function.

Identification of the Expected History

The OA-IGE computes the approximation of the expected outcome for an objective (Eq. 4.2), using either the expectation over the reward trajectory $\mathrm{E}_{\pi_\gamma,s_0}[H]$ or the reward sum $\mathrm{E}_{\pi_\gamma,s_0}[R]$ and the number of steps $\mathrm{E}_{\pi_\gamma,s_0}[T]$. Which expectation is used depends upon the form of the objective functions, either $f(h)$ or $f(r,n)$. Two variants for the OA-IGE exist, one for each of the two forms of the objective functions. The first, the Decoding OA-IGE, can theoretically approximate the expectation over the trajectory $\mathrm{E}_{\pi_\gamma,s_0}[H]$. It decodes the expected history based on Q-values from several modules. However, in practice it can only be applied to the subset of deterministic, goal-only-reward environments.

The second variant, the Prediction OA-IGE, can be applied for objective functions that use the reward sum and number of steps $f(r,n)$. It learns a prediction for the expected reward sum $\mathrm{E}_{\pi_\gamma,s_0}[R]$ and the expected number of steps $\mathrm{E}_{\pi_\gamma,s_0}[N]$ for the policy of each γ-module. In contrast to the Decoding OA-IGE, it can be applied to any OA-MDP. The next two sections describe the two OA-IGE variants and compare each experimentally to Q-learning in grid-world tasks.

4.3 Decoding of Trajectories

The first OA-IGE variant, the Decoding OA-IGE (DOA-IGE), is used for OA-MDPs with objectives $f(h)$ that use the full reward trajectory h as input. The algorithm decodes the expected reward trajectory $\mathrm{E}_{\pi_\gamma,s_0}[H]$ from its γ-modules to select the most appropriate γ-policy for the active objective. First, the algorithm is introduced, followed by an comparison of it to a classical Q-learning approach in a deterministic, goal-only-reward OA-MDP. The DOA-IGE outperformed Q-learning. It could immediately adapt to changing objective functions, whereas Q-learning needed several hundred episodes to learn a new policy for each.

4.3.1 The Decoding OA-IGE

The DOA-IGE (Algorithm 4.1) exploits the ability of the IGE to decode the expected reward trajectory $\mathrm{E}_{\pi_\gamma,s_0}[H]$ from several γ-modules that follow the same policy (Section 2.3). Based on this, it selects the γ-policy that maximizes the active objective f_k in the beginning of an episode by:

$$\tilde{\gamma} = \underset{\gamma}{\operatorname{argmax}}\, f_k(E_{\pi_\gamma,s_0}[H]) \ .$$

The selected module's policy $\pi_{\tilde{\gamma}}$ is then used during the whole episode to define the agent's actions.

Decoding of the expected trajectory is theoretically possible for any MPD (Section 2.3). However, in practice, decoding is difficult and numerically unstable. Nonetheless, it can be done for the subset of deterministic, goal-only-reward MDPs. In such MDPs the expected reward trajectory for a greedy policy π is given by:

$$\mathrm{E}_{\pi,s_0}[H] = (\underbrace{0,0,\ldots,0}_{\text{T-1 times}}, \mathrm{E}_\pi[r_{T-1}]) \ . \tag{4.4}$$

Algorithm 4.1: Decoding Objective Adaptive IGE

Input:

Learning rate: $\alpha \in [0, 1]$

Sorted list of discount factors: $\Gamma = (\gamma_1, \ldots, \gamma_m)$ with $\gamma_i \in (0, 1)$

initialize $Q_\gamma(s, a)$ to zero

repeat (for each episode)

  initialize state s, and goal f_k

  `// calculate expected rewards and number of steps for module pairs`

  for i **from** *1* **to** $m - 1$ **do**

    $R_i \leftarrow r(\gamma_i, \gamma_{i+1})$

    $T_i \leftarrow n_s(\gamma_i, \gamma_{i+1})$

  end

  `// remove invalid module pairs and calculate goal function` f_k

  for i **from** *2* **to** $m - 1$ **do**

    if $T_{i-1} = T_i$ *and* $T_i = T_{i+1}$ **then**

      $h \leftarrow (\underbrace{0, \ldots, 0}_{T_i - 1 \text{ times}}, R_i)$

      $J(i) \leftarrow f_k(h)$

    else

      $J(i) \leftarrow \emptyset$

    end

  end

  `// select as active module` $\tilde{\gamma}$ `a module that maximizes` f_k

  $\tilde{\gamma} \leftarrow \Gamma\left(\text{argmax}_i\, J(i)\right)$

  repeat (for each step in episode)

    $a \leftarrow$ choose an action for s derived from $Q_{\tilde{\gamma}}(s, a)$ (e.g. ϵ-greedy)

    r, $s' \leftarrow$ take action a, observe outcome

    forall the $\gamma \in \Gamma$ **do**

      $Q_\gamma(s, a) \leftarrow Q_\gamma(s, a) + \alpha\left(r + \gamma \max_{a'} Q_\gamma(s', a') - Q_\gamma(s, a)\right)$

    end

    $s \leftarrow s'$

  until s *is terminal-state*

until *termination*

The length T of the trajectory and the visited states are the same for each episode, because state transitions are deterministic and the policy is greedy. The only random variable is the final reward r_{T-1}. All previous rewards are 0. As a result, the expected trajectory $\mathrm{E}_{\pi,s_0}[H]$ can be constructed using the expected final reward $\mathrm{E}_{\pi,s_0}[r_{T-1}]$ and the trajectory length T. Given two modules (γ_a, γ_b) which have the same greedy policy ($\pi_{\gamma_a} = \pi_{\gamma_b}$), this information can be decoded from their values with:

$$T = \frac{\log(V_{\gamma_a}(s_0)) - \log(V_{\gamma_b}(s_0))}{\log(\gamma_a) - \log(\gamma_b)} \quad \text{and} \quad \mathrm{E}_{\pi_{\gamma_a},s_0}[r_{T-1}] = \frac{V_{\gamma_a}(s_0)}{\gamma_a^T} \ . \tag{4.5}$$

Theorem 8 (page 44) shows their derivation.

The following procedure decodes the expected trajectory $\mathrm{E}_{\pi,s_0}[H]$ for all modules and approximates their outcomes for the active objective. First, it iterates over all neighbor pairs (γ_i, γ_{i+1}) of its γ-modules $\Gamma = (\gamma_1, \ldots, \gamma_m)$. Neighboring pairs have usually similar values and therefore the same policy, which is the prerequisite to decode their expected trajectory by Eq. 4.5. Most pairs will have the same policy. However, for certain module pairs, the modules will have different policies and the resulting decoded length T and reward sum R are incorrect (Section 2.3). These cases are identified by checking whether the number of steps is similar to its neighboring pairs:

$$T(\gamma_{i-1}, \gamma_i) = T(\gamma_i, \gamma_{i+1}) = T(\gamma_{i+1}, \gamma_{i+2}) \ .$$

If this condition is violated, the module pair (γ_i, γ_{i+1}) is ignored. For all other pairs, the expected trajectory $\mathrm{E}_{\pi,s_0}[H]$ is constructed based on T and R (Eq. 4.4). Then, the outcome for the active objective function $f_k(\mathrm{E}_{\pi,s_0}[H])$ is calculated. One of the γ-modules from pairs with the highest expected outcome is selected and one of its module's policies is used for action selection during the episode.

4.3.2 Deterministic, Goal-Only-Reward Tasks

The DOA-IGE can be used for deterministic, goal-only-reward tasks. To evaluate its performance it has been compared to a time-dependent Q-learning approach. First, the task is described followed by an introduction of the time-dependent Q-learning algorithm.

Task Design

The DOA-IGE was evaluated on a discrete, two-dimensional grid-world task (Fig. 4.2). The grid world has 10x10 states. The agent starts each episode from a fixed position s_0. The episode ends if it reaches one of 7 goal states or after a maximum of 30 steps. Each goal state is reachable by a different minimum number of steps and it gives a different reward value (Fig. 4.2, b). The DOA-IGE has the ability to learn the optimal policy to reach 5 of the 7 goal states as illustrated by the γ-Curve (Fig. 4.2, c).

The location of the goal states and their rewards were chosen so that they represent different solutions for the nine objective functions ($J = (f_1, \ldots, f_9)$), the agents had to adapt to. Although, the objective functions $f_k(h)$ take the whole trajectory as input, they are are defined in terms of the final reward r_T and the number of steps T (Fig. 4.4). In goal-only-reward tasks, reward trajectories are zero for all rewards up to the final reward

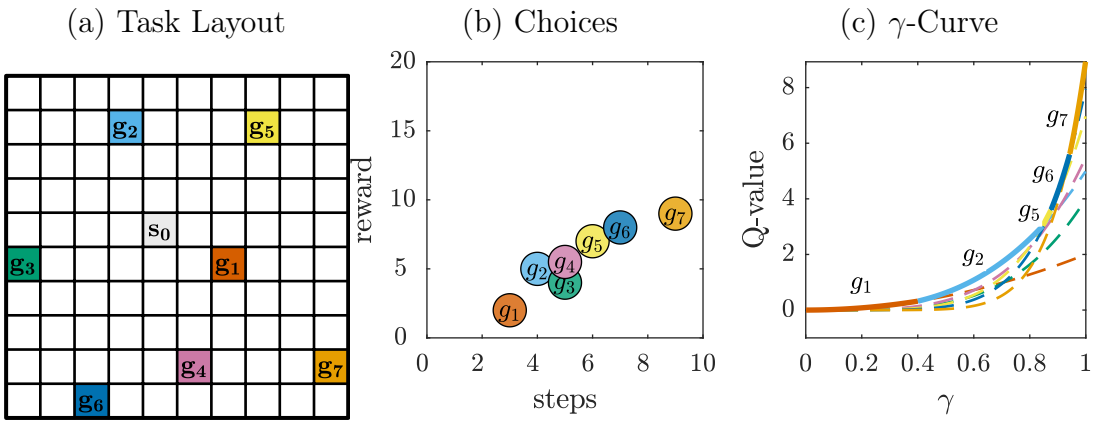


Figure 4.2: (a) Layout of the grid world used to evaluate the DOA-IGE in deterministic and stochastic, goal-only-reward tasks. The task has 7 goal states $g_1, \ldots, g_7$. The agent starts in state s_0. It can move in 4 directions (north, east, south, west). Each transition results in a reward of 0 until a goal state is reached. (b) If the agent starts in state s_0 it has several choices. Each choice represents the minimal number of steps n and the reward r of reaching one of the possible goal states. (c) The discounted values $V_\gamma(s_0, g) = \gamma^{n_g - 1} r_g$ for each choice for state s_0 (broken lines) show that the DOA-IGE will learn to reach 5 of the 7 goals. The solid line represents the values that the γ-Ensemble would learn $(\max_g V_\gamma(s_0, g))$.

r_{T-1}. Therefore, only information about the last reward and when it was reached is important.

The first objective f_1 is to receive the maximum reward in an episode. The second f_2 also maximizes reward, but a punishment of -1 for each step is given after 3 steps. f_3 gives exponentially increasing punishment for more than 3 steps. The goal of f_4 is to find the shortest path to the closest goal state. For f_5 the shortest path to a goal state that gives at least a reward of 6.5 is optimal. Reaching a goal state with less reward will result in a strong punishment. For f_6 the goal is to find the highest reward with a maximum of 7 steps. For f_7 the agent has only a maximum of 5 steps. In f_8 the goal is to maximize average reward. The final objective f_9 maximizes the average reward, but the agent has to reach at last a reward of 6.5.

The DOA-IGE was compared to a time-step dependent Q-learning that is described in the next section. For both algorithms, 100 runs were performed to measure their average learning performance. Each run consisted of 54,000 episodes that were divided in nine phases. In each phase, the agents had to adapt to a different objective function. The objectives did not change during the 6000 episodes of a phase to evaluate how long an agent needs to adapt to each objective. The performance was measured by the return for the objective function $f_k(h)$ that the agent received for each episode during the learning process.

Time-Dependent, Goal-Only-Reward Q-learning

The DOA-IGE was compared to a modified, time-dependent Q-learning variant (Algorithm 4.2). Q-learning had to be modified because the goal in OA-MDPs is to opti-

mize the objective function $f_k(h)$ and not the expected discounted future reward sum $\mathrm{E}_\pi\left[\sum_{t=0}^{T-1}\gamma^t r_t\right]$ as for standard Q-learning. Objective functions evaluate the whole reward trajectory of an agent during an episode. Q-learning can be adapted to optimize such objectives by adapting the rewards on which it is learning. Instead of using the OA-MDPs reward function R, the Q-values a learned on a different reward function ξ with

$$\xi(k, s_{t+1}) = \begin{cases} 0 & \mid s_{t+1} \notin S^G \\ f_k(h) & \mid s_{t+1} \in S^G \end{cases},$$

where k is the active objective and s_{t+1} is the state the agent transitions into. The function gives a reward of 0 for each step during an episode. The final reward, when a goal state is reached, is the outcome for the active objective based on the observed trajectory h. The Q-values are defined by:

$$Q(k, s, a) = \mathrm{E}\left[\sum_{t=0}^{T-1}\gamma^t \xi_t(k)\right],$$

where for each objective function f_k different values are learned.

Nonetheless, this approach has a problem. The final reward ξ_T depends on the full history of the agent and not only on its current state and action. This violates the Markov property that is needed to guarantee that Q-learning can solve the task optimally. This problem can by solved for goal-only-reward task by using the information of the current time step $i \in (0, \ldots, T-1)$ as an extra state dimension:

$$Q(k, i, s, a) = \mathrm{E}\left[\sum_{t=0}^{T-1}\gamma^t \xi_t\right], \text{with } \xi_t = \begin{cases} 0 & \mid t < T-1 \\ f_k(h^i) & \mid t = T-1 \end{cases}, \tag{4.6}$$

where h^i is the trajectory assuming that the agent is in the i'th time step:

$$h^i = [\underbrace{0, 0, \ldots, 0}_{\text{i-1 times}}, r_{T=i-1}]\ . \tag{4.7}$$

The final reward $\xi_{T-1} = f(h^i)$ used for learning the Q-function depends only on the final reward r_{T-1} from the reward function R and the number of steps i. As a result, the Markov property is fulfilled because the agent has the time step i as state information.

The update rule for the Q-function based on a observation $(k, t, s_t, a_t, s_{t+1})$ is:

$$\begin{aligned} Q(k, t, s_t, a_t) \leftarrow\ & Q(k, t, s_t, a_t) \\ & + \alpha\left(\xi(k, s_{t+1}) + \gamma \max_{a'} Q(k, t+1, s_{t+1}, a_{t+1}) - Q(k, t, s_t, a_t)\right), \end{aligned}$$

Q-values for all time steps $t = (0, \ldots, T-2)$ can be updated simultaneously for one observation, because the reward ξ_t depends only on the time step information and the current reward r_t. Moreover, the transition for time steps is fixed to $t \to t+1$. As a result, the update can be done for every time step after an observation, reducing the number of observations to learn the Q-function.

Algorithm 4.2: Time-dependent Q-learning for goal-only-reward OA-MDPs

Input:
Learning rate: $\alpha \in [0, 1]$
Discount factor: $\gamma \in [0, 1]$
Maximum number of steps: $T \in \mathbb{N}$

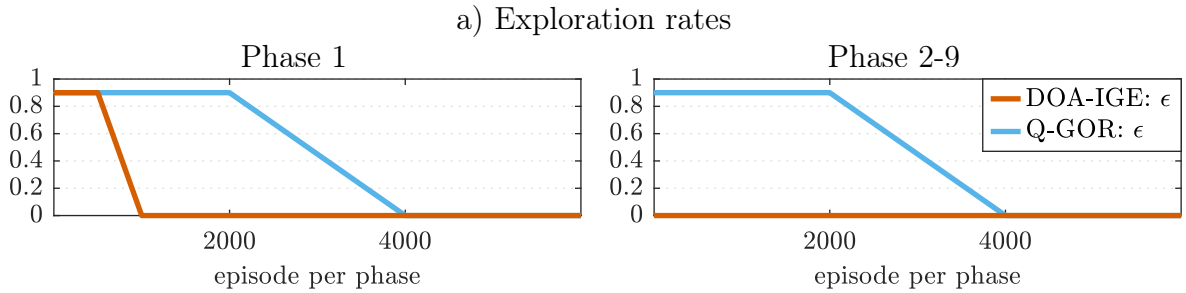
initialize $Q(k, t, s, a)$ to zero
repeat (for each episode)
  initialize state s, and goal f_k
  repeat (for each step t in episode)
    $a \leftarrow$ choose an action for s derived from $Q(k, t, s, a)$ (e.g. ϵ-greedy)
    `// take action and observe if it ends in a terminal state`
    r, s', $isTerminal \leftarrow$ take action a, observe outcome
    forall the $i \in (0, \ldots, T-2)$ **do**
      `// use the goal definition` f_k `as final reward of a task`
      if $isTerminal = true$ ***or*** $i + 1 = T$ **then**
        $h^i \leftarrow (\underbrace{0, \ldots, 0}_{i-1 \text{ times}}, r)$
        $\xi \leftarrow f_k(h^i)$ or $f_k(\sum_{j=0}^{i} h_j^i, i + 1)$
      else
        $\xi \leftarrow 0$
      end
      $Q(k, i, s, a) \leftarrow Q(k, i, s, a) + \alpha \left(\xi + \gamma \max_{a'} Q(k, i + 1, s', a') - Q(k, i, s, a)\right)$
    end
    $s \leftarrow s'$
  until s *is terminal-state*
until *termination*

Learning Parameters

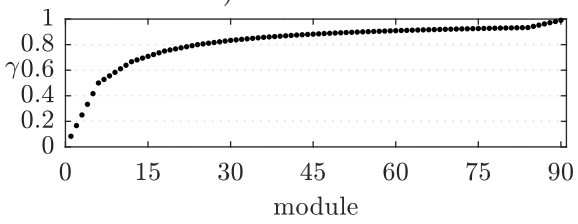
Learning parameters of the OA-IGE and time-dependent Q-learning were manually adjusted to allow learning of the optimal policy with few episodes. Both agents used the learning rate $\alpha = 1$ because the task is deterministic.

The ϵ-greedy action-selection strategy was used for both algorithms. However, their exploration rates differed (Fig. 4.3, a). The DOA-IGE explored only during the first phase. It kept an exploration rate of $\epsilon = 0.9$ for the first 500 episodes. Afterward, it reduced the exploration rate linearly to $\epsilon = 0$ until episode 1000. For the rest of Phase 1 and all successive phases, it had no exploration ($\epsilon = 0$). DOA-IGE, did not need further exploration because if a phase and the associated objective function changed, it could immediately select an appropriate policy from its γ-policies that it learned in the beginning of Phase 1.

In contrast, exploration was needed for Q-learning at the beginning of each phase. In



b) IGE Γ-set



c) Remaining parameters

Algorithm	Parameters
DOA-IGE	$\alpha = 1$
Q-GOR	$\alpha = 1$ $\gamma = 0.99$ $T = 30$

Figure 4.3: The learning parameters for the deterministic, goal-only-reward task. Both algorithms had a strong exploration at the beginning of learning that was reduced over time (a). Q-learning needed more exploration and it had to explore in each phase. DOA-IGE needed to explore only in Phase 1. The DOA-IGE used a set of 90 γ-modules with γ parameters that are more densely distributed towards $\gamma = 0.933$ (b).

the first 2000 episodes of each phase its exploration rate was set to $\epsilon = 0.9$. Afterward, the exploration rate was reduced to $\epsilon = 0$ until episode 4000. Q-learning's exploration is longer compared to the first phase of the DOA-IGE. This was necessary to allow Q-learning to find the optimal policy in Phase 5. Shorter explorations could not identify the optimal policy in this phase. Other phases would only have required a similar exploration strategy as the DOA-IGE for Phase 1.

The DOA-IGE used 90 γ-modules (Fig. 4.3, b). Their γ parameters followed the recommended set of modules from Theorem 9 (page 47). First, 14 modules were chosen according to: $\gamma_{i=1:14} = \frac{i}{i+1}$. Then, between each module γ_i and its predecessor module γ_{i-1}, four extra modules were added with equal distances between all modules. Between the last module $\gamma_{i=14}$ and 1, five modules were added with equal distances between all modules. This procedure resulted in a distribution of γ parameters that is more dense toward $\gamma = 0.933$. The discount factor of the Q-learning approach was set to $\gamma = 0.99$.

Experimental Results

The results of the experiment followed the expected outcomes for the DOA-IGE and the time-dependent Q-learning approach (Fig. 4.4). In terms of asymptotic performance, the DOA-IGE could reach the optimal policy for each objective besides f_7. The DOA-IGE was not able to solve it optimally because the optimal behavior, to go to goal state g_4, is not in the policy set of the DOA-IGE (Fig. 4.2, c). In contrast, the time-dependent Q-learning approach could find the optimal policy for every objective.

However, Q-learning needed to learn the optimal policy in each phase, which took

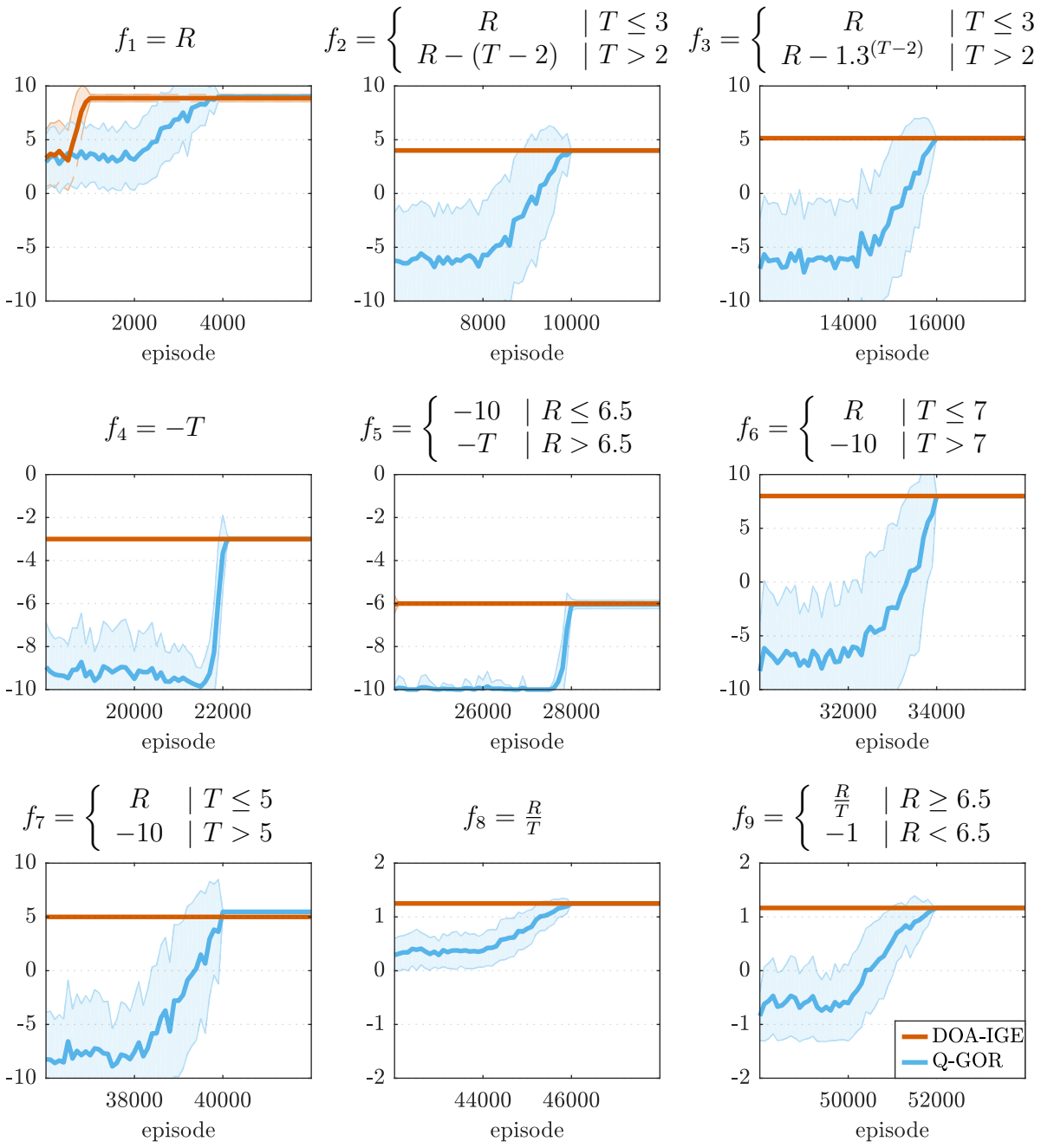


Figure 4.4: The DOA-IGE could immediately adapt to new objective functions compared to the time-dependent Q-learning approach in the deterministic, goal-only-reward task (Fig. 4.2). The performance was measured by the outcome of the objective function $f_k(h)$ per episode. Each of the 9 phases had a different objective function. R denotes the final reward r_{T-1}. The plots show the mean and the standard deviation over 100 runs per algorithm. The minimal reward per episode was limited to -10 to make the plots more readable, because some goal formulations can result in a large negative reward during explorations.

approximately 4000 episodes per phase. DOA-IGE could immediately adapt to any new objective after it learned its policy set in Phase 1. Moreover, Q-learning needed more exploration compared to the DOA-IGE in Phase 1. The longer exploration was needed to allow Q-learning to find the optimal policy in Phase 5.

4.3.3 Conclusion

The ability of the IGE to decode the expected trajectory can be used adapt immediately to a new objective function. Results from such a task with different goal formulations show that the DOA-IGE can outperform a Q-learning approach in terms of adaptation speed. The Q-learning approach needed several thousand episodes to adapt to a new objective. The DOA-IGE could immediately adapt; however, its decoding approach is currently restricted to deterministic, goal-only-reward OA-MDPs. The next section introduces a OA-IGE variant that can be used for any OA-MDP.

4.4 Prediction of Trajectory Statistics

The second OA-IGE variant, the Prediction OA-IGE (POA-IGE), can be applied for objectives $f(R, T)$ with the reward sum $R = \sum_{t=0}^{T-1} h_t$ and the trajectories length T as input. The algorithm learns the expectation over the reward sum and the trajectory length for each of its γ-modules to select the most appropriate for a given objective. First, the algorithm is introduced. Afterward, it is compared to Q-learning in two different task environments. The first is a stochastic, goal-only-reward task. The second is a general stochastic task. The results show that the POA-IGE can adapt immediately to new objectives compared to Q-learning.

4.4.1 The Trajectory Prediction OA-IGE

The POA-IGE (Algorithm 4.3) selects the most appropriate of its γ-policies for an objective f_k using the expectation over the reward sum $\mathrm{E}_{\pi_\gamma, s_0}[R]$ and the number of steps $\mathrm{E}_{\pi_\gamma, s_0}[T]$ for each policy:

$$\pi_k = \underset{\pi_\gamma}{\operatorname{argmax}} f_k(E_{\pi_\gamma, s_0}[R_\gamma], E_{\pi_\gamma, s_0}[T_\gamma]) \ .$$

Expectations are learned for every γ-module in form of state-dependent value functions, $R_\gamma(s)$ and $T_\gamma(s)$, because expectations can be formulated in a recursive manner according to the Bellman equation. This follows the idea behind the Horde architecture (Section 1.3) that is also composed of different value functions. The functions differ in their reward functions, which make it possible to predict different future events with them, such as the expected reward sum and number of steps until a goal is reached.

The expected reward sum $E_{\pi_\gamma, s_t}\left[\sum_{k=0}^{T-1-t} r_{t+k}\right]$ of using policy π_γ starting from state s_t is given by:

$$\begin{aligned} R_\gamma(s_t) &= E_{\pi_\gamma, s_t}\left[\sum_{k=0}^{T-1-t} r_{t+k}\right] \\ &= E_{\pi_\gamma, s_t}[r_t] + E_{\pi_\gamma, s_t}[r_{t+1}] + \ldots + E_{\pi_\gamma, s_t}[r_{T-1}] \\ &= E_{\pi_\gamma, s_t}[r_t] + E_{\pi_\gamma, s_t}[R_\gamma(s_{t+1})] \ . \end{aligned}$$

Algorithm 4.3: Prediction Objective Adaptive IGE

Input:
Learning rate: $\alpha \in [0, 1]$
Sorted list of discount factors: $\Gamma = (\gamma_1, \ldots, \gamma_m)$ with $\gamma_i \in (0, 1)$

initialize $Q_\gamma(s, a)$, $R_\gamma(s)$, and $T_\gamma(s)$ to zero
repeat (for each episode)
  initialize state s, and goal f_k
  `// select as active module` $\tilde{\gamma}$ `a module that maximizes` f_k
  $\tilde{\gamma} \leftarrow \text{argmax}_\gamma \, f_k(R_\gamma(s), T_\gamma(s))$
  repeat (for each step in episode)
    $a \leftarrow$ choose an action for s derived from $Q_{\tilde{\gamma}}(s, a)$ (e.g. ϵ-greedy)
    r, $s' \leftarrow$ take action a, observe outcome
    forall the $\gamma \in \Gamma$ **do**
      $Q_\gamma(s, a) \leftarrow Q_\gamma(s, a) + \alpha \left(r + \gamma \max_{a'} Q_\gamma(s', a') - Q_\gamma(s, a)\right)$
      `// update` R `and` N `only if the greedy action was used`
      if $a = \text{argmax}_{\bar{a}} \, Q_\gamma(s, \bar{a})$ **then**
        $R_\gamma(s) \leftarrow R_\gamma(s) + \alpha \left(r + R_\gamma(s') - R_\gamma(s)\right)$
        $T_\gamma(s) \leftarrow T_\gamma(s) + \alpha \left(1 + T_\gamma(s') - T_\gamma(s)\right)$
      end
    end
    $s \leftarrow s'$
  until *s is terminal-state*
until *termination*

The number of steps can be defined in the same recursive manner:

$$\begin{aligned} T_\gamma(s_t) &= E_{\pi_\gamma, s_t}[T] \\ &= E_{\pi_\gamma, s_t}\left[\sum_{t=0}^{T-1} 1\right] \\ &= 1 + \Pr(t+2|\pi_\gamma) + \Pr(t+3|\pi_\gamma) + \cdots + \Pr(T|\pi_\gamma) \\ &= 1 + E_{\pi_\gamma, s_t}[T(s_{t+1})] \ , \end{aligned}$$

where $\Pr(t|\pi_\gamma)$ is the probability that the trajectory has t steps, i.e. that it did not end in a terminal state until then. Expectations are formulated for episodic environments that end if a goal state or a maximum number of steps per episode is reached. In continuous tasks, steps $T_\gamma(s_t)$ would diverge toward infinity and the reward sum might diverge. The idea of state-dependent discount factors from the Horde framework (Sutton, Modayil, et al. 2011) could be used to stop the learning after a goal state was reached.

The POA-IGE learns predictions for each module in parallel to the learning of their Q-functions. Based on an observation (s_t, a_t, r_t, s_{t+1}), the expectation of the γ-modules, which have action a_t as greedy action $(\forall \gamma : \ a_t = \text{argmax}_a \, Q_\gamma(s_t, a))$, are updated by:

$$R_\gamma(s_t) \leftarrow R_\gamma(s_t) + \alpha \left(r_t + R_\gamma(s_{t+1}) - R_\gamma(s_t)\right) \ ,$$

and

$$T_\gamma(s_t) \leftarrow T_\gamma(s_t) + \alpha \left(1 + T_\gamma(s_{t+1}) - T_\gamma(s_t)\right) \ ,$$

where α is the learning rate parameter also used for the update of the Q-values.

At the beginning of an episode the active module $\tilde{\gamma}$ is selected based on the expectations for the initial state s_0:

$$\tilde{\gamma} = \underset{\gamma}{\operatorname{argmax}} f_k \left(R_\gamma(s_0), T_\gamma(s_0)\right) \ .$$

The policy of the selected module $\tilde{\gamma}$ is then followed during the episode.

In contrast to the Decoding OA-IGE which can only be applied to deterministic, goal-only-reward environments, the Prediction OA-IGE can be applied to any OA-MDP. For deterministic, goal-only-reward MDPs, preliminary experiments showed that the DOA-IGE and the POA-IGE perform similarly. The next two sections evaluate the POA-IGE in a stochastic, goal-only-reward task and a general stochastic task without restrictions on the reward function.

4.4.2 Stochastic, Goal-Only-Reward Tasks

The POA-IGE is first compared to the time-dependent Q-learning approach for goal-only-reward tasks (Algorithm 4.2) in a stochastic, goal-only-reward task. Q-learning outperformed POA-IGE in asymptotic performance for most objective functions, but the POA-IGE adapted immediately to a new objective. Q-learning needed several thousand episodes to do so. The next part introduces the experimental design, followed by the learning parameters used for the algorithms and the results of the experiment.

Task Design

The task uses the same grid world environment as the deterministic, goal-only-reward task (Fig. 4.2). In contrast, a stochasticity of $\eta = 0.1$ for state transitions was introduced, where for each transition a probability of η exist that the agent moves in a random direction, instead of its intended direction. The task has the same nine objective functions ($J = (f_1, \ldots, f_9)$) as the deterministic task, but their formulations changed (Fig. 4.6). Instead of using the full trajectory h as input, the objectives use the reward sum $R = \sum_{t=0}^{T-1} r_t$ and the length T of the trajectory as input. The task was separated in nine phases. Each phase had a different objective function and the agents had 6000 episodes to adapt to it.

Learning Parameters

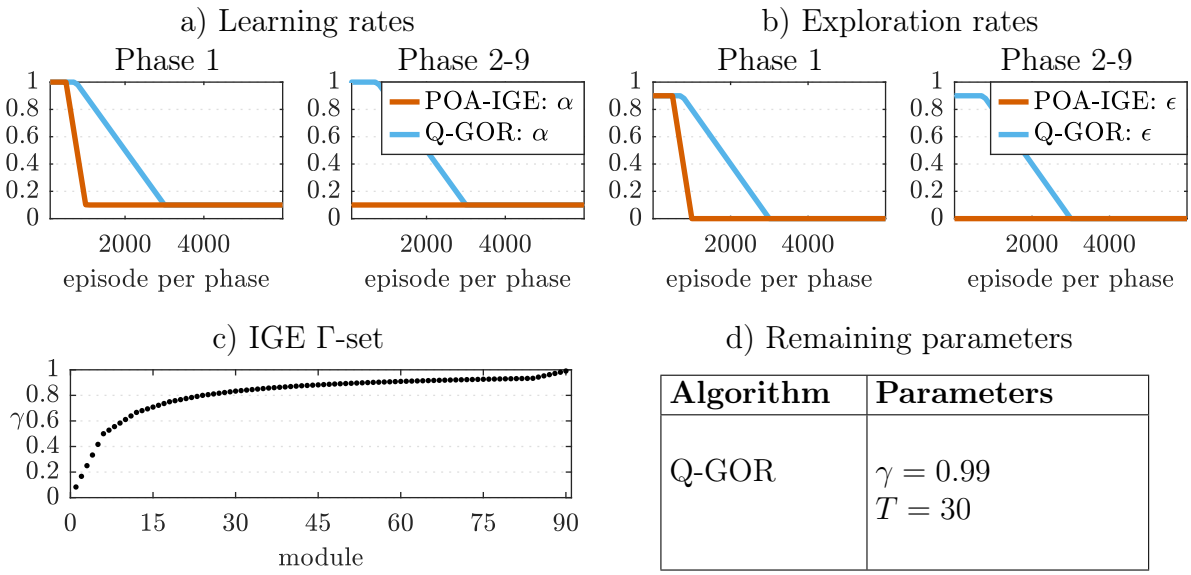
The POA-IGE was compared to the time-dependent Q-learning approach for goal-only-reward tasks (Algorithm 4.2). Learning parameters of both algorithms were manually optimized to yield a high asymptotic performance while having a high learning rate.

The learning rate parameter α of both algorithms was set to $\alpha = 1$ in the beginning of learning to allow a faster convergence of the values (Fig. 4.5, a). Over the course of learning it was reduced to $\alpha = 0.1$. The POA-IGE kept $\alpha = 1$ for 500 episodes and reduced it linearly to $\alpha = 0.1$ until episode 1000. The learning rate stayed at $\alpha = 0.1$ for

the rest of Phase 1 and for all following phases. The Q-learning approach needed a longer learning time to reach its asymptotic performance in each phase. Moreover, it needed to learn a new policy for each phase. Its learning was kept to $\alpha = 1$ for 750 episodes and linearly reduced to $\alpha = 0.1$ until episode 3000 for each phase.

Both algorithms used the ϵ-greedy action selection. Similar to the learning rate, the exploration rate ϵ was high at the start of learning and then reduced (Fig. 4.5, b). The POA-IGE's exploration rate was $\epsilon = 0.9$ for the first 500 episodes of Phase 1 and then reduced to $\epsilon = 0$ until episode 1000. It stayed at $\epsilon = 0$ for the rest of Phase 1 and all successive phases. Q-learning used a learning rate of $\epsilon = 0.9$ for the first 750 episodes. Afterward, it was linearly reduced to $\epsilon = 0$ until episode 3000 for each phase.

The POA-IGE used a set of 45 γ-modules (Fig. 4.5, c). Their discount factors were chosen using the same process as for the DOA-IGE for the deterministic, goal-only-reward experiment (Section 4.3.2). First, 14 modules were chosen according to $\gamma_{i=1:14} = \frac{i}{i+1}$, but instead of adding 4 modules with equal distance to each other between each of the chosen 14 modules, only 2 were added. The POA-IGE needed fewer modules compared to the DOA-IGE, because the DOA-IGE needs three modules with the same policy to decode their expected trajectory. As a result, additional modules were necessary to ensure that each possible policy in the task was learned by at least two modules. In contrast, the POA-IGE learns the expected return R_γ and number of steps T_γ for each module's policy by the module itself. Thus, only one module per policy is needed. The discount factor of Q-learning was set to $\gamma = 0.99$.



Algorithm	Parameters
Q-GOR	$\gamma = 0.99$ $T = 30$

Figure 4.5: The learning parameters for the stochastic, goal-only-reward task and the general task. Both algorithms had a high learning rate and exploration rate at the beginning of the learning that were reduced over time (a,b). Q-learning needed more exploration and it had to learn a new policy in each phase. POA-IGE needed to explore and learn its γ-policies only in Phase 1. The POA-IGE used a set of 45 γ-modules with γ parameters that are more densely distributed towards $\gamma = 0.933$ (c).

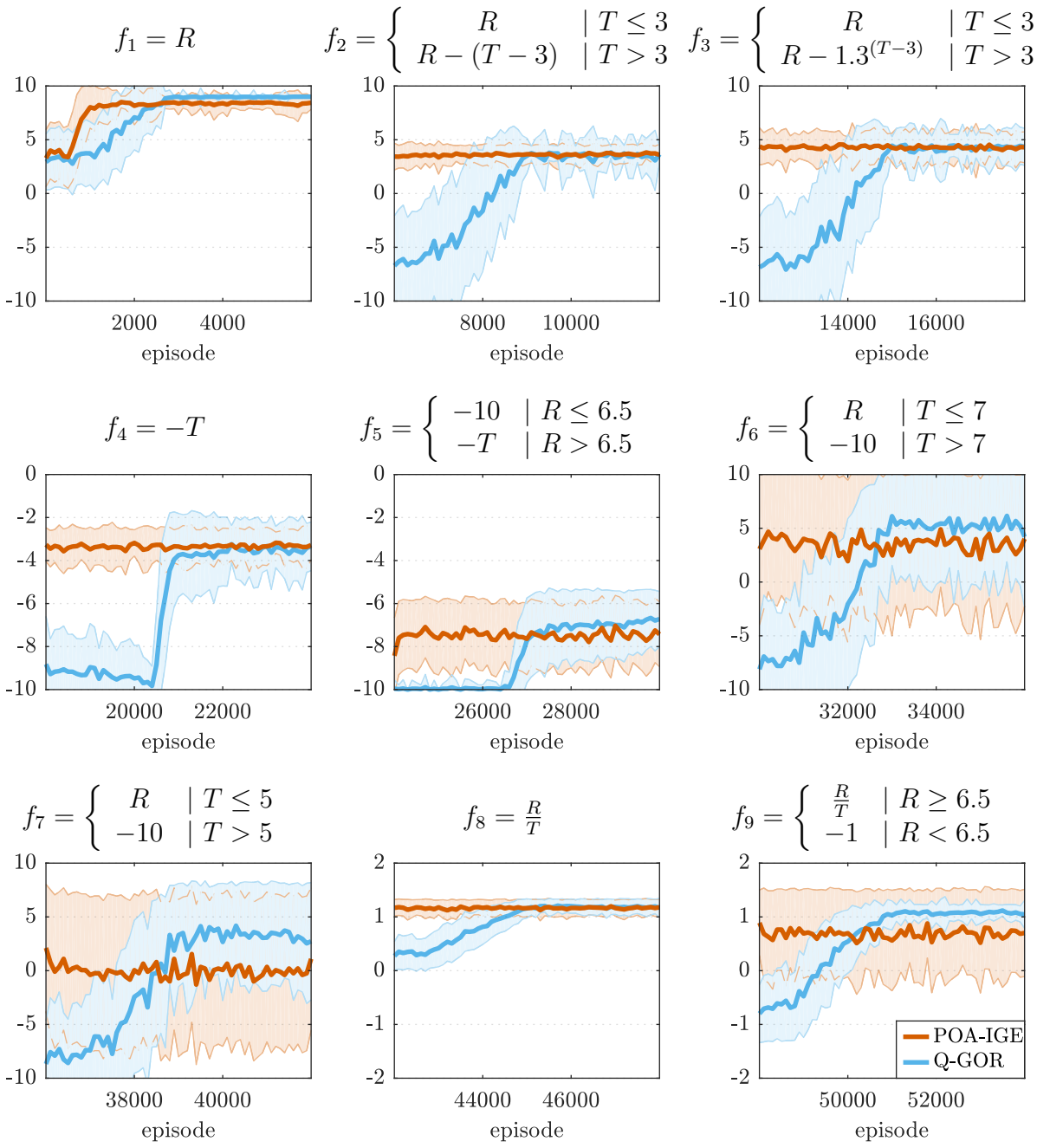


Figure 4.6: The POA-IGE could immediately adapt to new objective functions compared to the time-dependent Q-learning approach in the stochastic, goal-only-reward task (Fig. 4.2). Performance was measured by the outcome of the objective function $f_k(R, T)$ per episode, where $R = \sum_{t=0}^{T-1} r_t$ is the reward sum of the agent's trajectory and T is its length. Each of the 9 phases has a different objective function. The plots show the mean and the standard deviation over 100 runs per algorithm. The minimal reward per episode was limited to -10 to make the plots more readable, because some goal formulations can result in a large negative reward during explorations.

Experimental Results

The experimental results (Fig. 4.6) show that Q-learning reached a better asymptotic performance for 5 of the 9 objectives (f_1, f_5, f_6, f_7, f_9). The POA-IGE could reach a similar asymptotic performance for the other 4 objectives (f_2, f_3, f_4, f_8). The asymptotic performance of the OA-IGE approach was worse compared to the deterministic, goal-only-reward task (Fig. 4.4), because in deterministic environments the OA-IGE is guaranteed to approximate the correct outcome of its policies for each objective (Theorem 13). This is not guaranteed for the stochastic task. This was especially a problem for objectives with hard constraints (f_5, f_6, f_7, f_9).

Nonetheless, the POA-IGE could immediately find a relatively appropriate policy for each new objective. Q-Learning needed approximately 2000 to 3000 episodes to reach the same performance as the POA-IGE.

4.4.3 General Tasks

The POA-IGE can be applied to general MDPs where non-zero rewards can be given for any transition, because the POA-IGE learns the expectation over the full reward sum of a trajectory. This section compares the POA-IGE to Q-learning in such a task. Because of the non-zero rewards, the introduced time-dependent Q-learning approach (Algorithm 4.2) cannot guarantee convergence upon the optimal policy. It violates the Markov property in such tasks. An adapted variant of the Q-learning algorithm can be formulated that is better suited. However, this variant also cannot guarantee optimality. The results show that the POA-IGE outperformed it in terms of asymptotic performance and adaption speed.

This section introduces first the task used to evaluate POA-IGE and the Q-learning approach. Afterward, the changed version of the time-dependent Q-learning approach is introduced, followed by the experimental results.

Task Design

The task environment is a stochastic ($\eta = 0.1$) two-dimensional grid world (Fig. 4.7). It is similar to the environment used for the goal-only reward tasks (Fig. 4.2). It has the same basic layout with 7 goal states and a fixed start position. In contrast to the previous task, a punishment is given as reward for each state transition that does not end in a goal state (Fig. 4.7, a). For each goal state there exists an optimal path for which the agent receives a punishment of -1 per step. If the agent leaves the optimal path, it receives a punishment of -2. The rewards for reaching a goal state (Fig. 4.7, b) are chosen so that following the optimal path results in the same reward sum as for the optimal path to the same goal in the previous tasks.

The OA-MDP has the same nine objectives from the previous tasks (Fig. 4.8). Each objective is active in a different phase of the experiment. Each phase has 6000 episodes.

Time-Dependent Q-learning for General Tasks

The POA-IGE was compared to an adapted version of the time-dependent Q-learning approach. The Q-learning approach was adapted to work with environments that can have

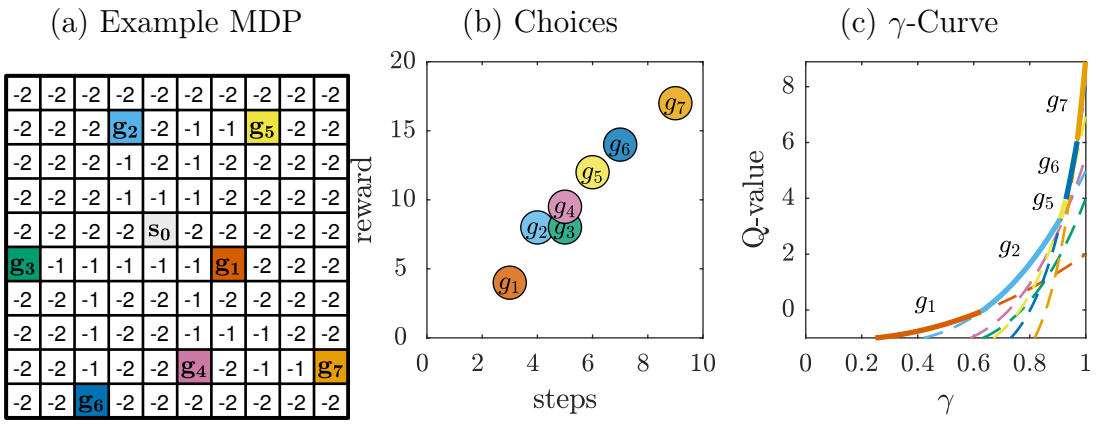


Figure 4.7: (a) A grid-world MDP with 7 goal states $g_1, \ldots, g_7$. The agent starts in state s_0. It can move in 4 directions (north, east, south, west). Transitions result in zero reward until a goal state is reached. (b) If the agent starts in state s_0 it has several choices to choose from. Each choice represents the minimal number of steps n and the reward r of reaching one of the possible goal states. (c) Discounted values $V_\gamma(s_0, g) = \gamma^{n_g - 1} r_g$ for each choice for state s_0 (broken lines). The solid line represents the values the γ-Ensemble would learn ($\max_g V_\gamma(s_0, g)$). For state s_0 the ensemble will learn policies to go to goal states g_1, g_2, g_5, g_6, g_7. It will not learn policies to go to g_3, g_4.

non-zero rewards for all time steps. The previous Q-learning approach (Algorithm 4.2) assumes that all rewards are zero until a goal-state is reached. Based on this assumption it recreates the past reward trajectory h. All rewards for the last $t - 1$ steps are set 0 and only if a goal state is reached is the current reward set to $R(s)$ (Eq. 4.7). This is not possible for general non-zero reward environments.

The adapted version of the Q-learning approach stores (Algorithm 4.4) at every time step the reward r_t in a history h. As in the previous version it learns its Q-function based on a changed reward function ξ (Eq. 4.6). It updates its Q-values for every step during an episode by zero ($\xi_t = 0$). If a goal state, i.e. a terminal state, is reached, it uses as reward for its Q-values the outcome for the active objective f_k. The outcome is either computed based on the full trajectory h or its reward sum and length:

$$\xi_{T-1} = f_k(h) \quad \text{or} \quad \xi_{T-1} = f_k\left(\sum_{i=0}^{|h|-1} h_i, |h|\right) .$$

The adapted, time-dependent Q-learning approach cannot guarantee convergence upon the optimal policy because the Markov property is violated for it. The final reward ξ_T in an episode depends on the full reward history and not only on the previous state and action. Optimality can only be guaranteed if the full reward trajectory of all previous steps is part of the state information ($Q : S \times H \times A \mapsto \mathbb{R}$). Unfortunately, this creates a huge state space for which learning is impractical. Nonetheless, with time information, Q-learning can still converge in many cases to an appropriate policy, as shown by the experimental results.

Algorithm 4.4: Time-dependent Q-learning for general OA-MDPs

Input:
Learning rate: $\alpha \in [0, 1]$
Discount factor: $\gamma \in [0, 1]$

initialize $Q(k, t, s, a)$ to zero
repeat (for each episode)
  initialize state s, and goal f_k
  `// start with an empty reward history`
  $h \leftarrow \emptyset$
  repeat (for each step t in episode)
    $a \leftarrow$ choose an action for s derived from $Q(k, t, s, a)$ (e.g. ϵ-greedy)
    `// take action and save reward in history`
    r, s', $isTerminal \leftarrow$ take action a, observe outcome
    $h_t \leftarrow r$
    `// if a goal state is reached, use the outcome for the`
    `// objective` f_k `as basis for the Q-function`
    if $isTerminal = true$ **then**
      $\xi \leftarrow f_k(h)$ or $f_k(\sum_{i=0}^{t-1} h_i, t)$
    else
      $\xi \leftarrow 0$
    end
    $Q(k, t, s, a) \leftarrow Q(k, t, s, a) + \alpha \left(\xi + \gamma \max_{a'} Q(k, t+1, s', a') - Q(k, t, s, a)\right)$
    $s \leftarrow s'$
  until s *is terminal-state*
until *termination*

Experimental Results

The POA-IGE and the adapted time-dependent Q-learning approach were evaluated using the same learning parameters as for the stochastic, goal-only-reward task (Fig. 4.5). Compared to the results of the stochastic, goal-only-reward task (Fig. 4.6), the POA-IGE performed better compared to Q-learning in terms of asymptotic performance and learning speed in the general task.

The POA-IGE outperformed Q-learning in its asymptotic performance in 5 of the 9 objectives (f_1, f_2, f_3, f_4, f_9). Both had a similar final performance for 3 objectives (f_5, f_6, f_8). Q-learning could slightly outperform POA-IGE for objective f_7. The reduced performance of the Q-learning approach, compared to the stochastic, goal-only-reward task, is the result of the violated Markov property. It hindered the algorithm to converge upon the optimal policy for some objectives.

Its low performance for objectives f_4 and f_5 was the result of the negative outcome values that these objectives give. Q-values were initialized to 0. Because the outcome for all objectives are negative, the agent is doing an optimistic exploration (Osband &

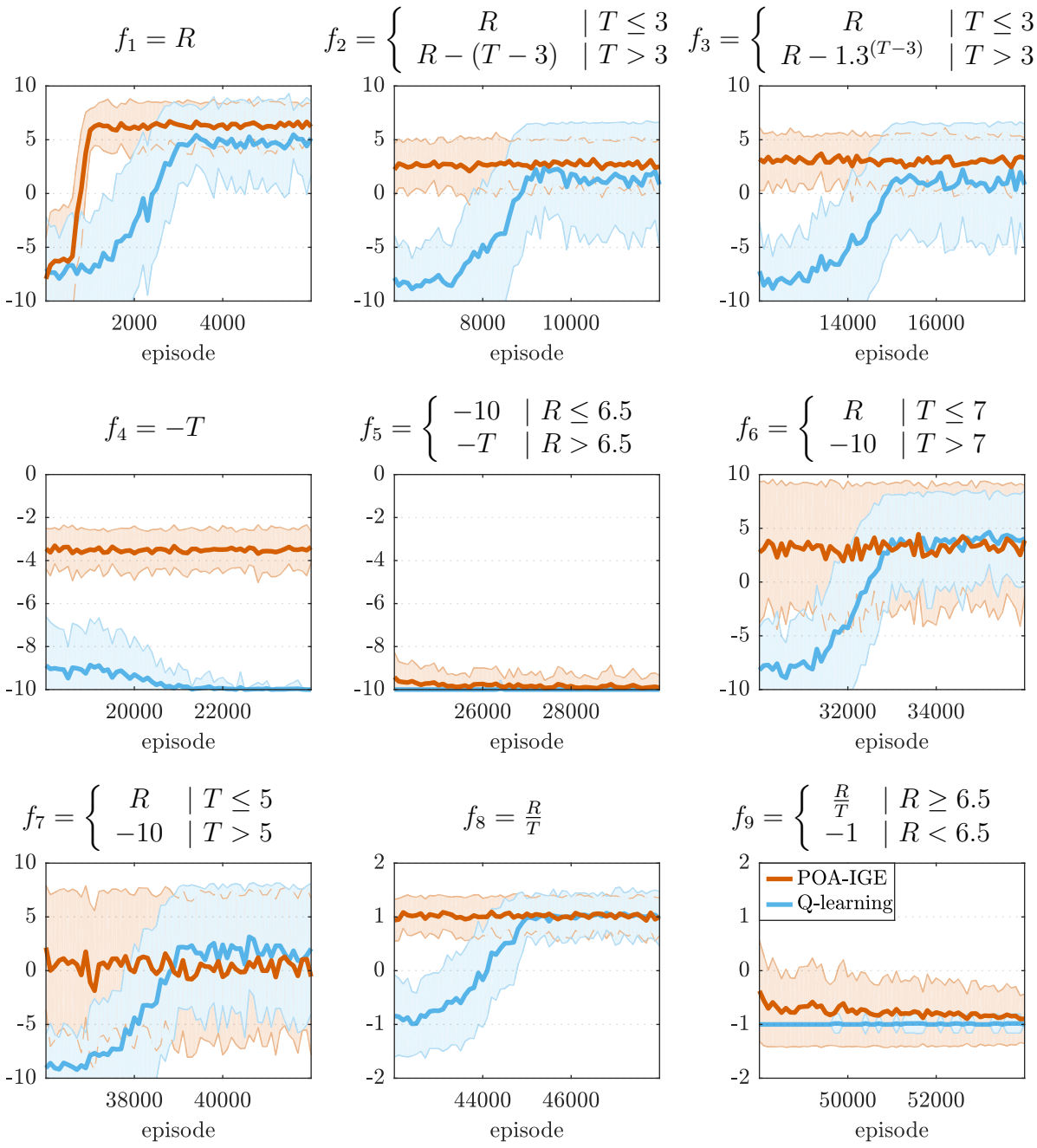


Figure 4.8: The POA-IGE immediately adapted to new objective functions compared to the time-dependent Q-learning approach in the stochastic task (Fig. 4.7). Performance was measured by the outcome of the objective function $f_k(R, T)$ per episode, where $R = \sum_{t=0}^{T-1} r_t$ is the reward sum of the agent's trajectory and T is its length. Each of the 9 phases has a different objective function. The plots show the mean and standard deviation over 100 runs per algorithm. The minimal reward per episode was limited to -10 to make the plots more readable, because some goal formulations can result in a large negative reward during explorations.

Van Roy 2017). It explores every possible state action pair for every possible time step, because it has to learn that their initial Q-values of 0 are not optimal. As a result, Q-learning needed more episodes to learn a good policy for these objectives. In the previous goal-only-reward experiments, the leaning rate was faster because the Q-learning approach could update its Q-values for all possible time steps of an state-action pair simultaneously for one observation.

As for the previous tasks, the POA-IGE could immediately adapt to a new objective function. Q-learning needed at least 3000 episodes to reach its final asymptotic performance.

4.4.4 Conclusion

The POA-IGE learns for each of its module's policies the expected reward sum and the expected number of steps. It uses this information to find its most appropriate policy for objective functions that use the reward sum and the length of the agent's trajectory as input. In contrast to the DOA-IGE, which decodes the expected trajectory from its modules Q-values, the POA-IGE can be applied to any task, regardless of being deterministic or stochastic or depending on the time points when non-zero reward is given. The experimental results show that the POA-IGE could immediately adapt to an unseen objective function. In contrast, time-dependent Q-learning approaches needed several thousand episodes to learn policies that reach the same performance. In tasks where non-zero reward is given for any state transition, the POA-IGE could also outperform Q-learning in its asymptotic performance.

4.5 Discussion

The following sections discuss similar approaches to the OA-IGE framework, the limitations to its ability to guarantee convergence upon the optimal policy and its memory demand in contrast to classical Q-learning.

Similar Zero-shot Learning Approaches

The ability of the OA-IGE to adapt immediately to an unknown objective makes it a zero-shot learning algorithm. The goal in zero-shot learning is to solve an unseen task on the first attempt, using a description of the task. In the case of OA-MPDs, the task is to optimize a new objective function in an MDP. The description is the objective function itself. Zero-shot learning is part of transfer learning (Section 1.2). Different zero-shot learning approaches have been explored in reinforcement learning, but none is connected to the problem of adapting an agent to different objectives based on the trade-off between the amount of gained reward and the invested time. The next part describes some approaches and compares the OA-IGE to them.

Some approaches make use of policies π_θ parameterized by a vector $\theta \in \mathbb{R}^N$. In the approach by Da Silva, Konidaris, & Barto (2012), tasks are also parameterized with a vector. After observing some tasks and learning their optimal policy parameters, a map from task parameters to policy parameters is learned. Given the task parameters of a

new task, the mapping is used to extrapolate its policy parameters. In the framework by Isele, Rostami, & Eaton (2016), each task t has a descriptor vector m^t. The approach learns for each task a coefficient vector s^t to describe the policy parameters $\theta^t = Ls^t$ and the task description $m^t = Ds^t$. L is a shared basis do define policies and D is a dictionary. Both are learned over several tasks. Given a new task t' and its descriptor $m^{t'}$, the approach identifies the coefficient $s^{t'}$ that describes best the new description $m^{t'} = Ds^{t'}$. The new $s^{t'}$ is then used to define the new policy by $\theta'^t = Ls^{t'}$. In difference, the OA-IGE learns several value-based policies by its γ-modules. It selects the most appropriate given a new objective. However, the discount parameter γ associated with each policy can be interpreted as a policy parameter ($\theta = \gamma$) that is similar to the discussed approaches. Nonetheless, learning a map from objective functions to the appropriate γ parameters is difficult, because the objectives in OA-MDPs can be expressed as any complex mathematical formulation and not as a relatively simple task vector. Moreover, the OA-IGE does not need to know the formulation of the objective functions. It only needs to compute their outcome for expected trajectories to adapt to them.

A different class of approaches has domain adaptation as goal where a specific task should be handled in different domains or environments. The approach of Devin et al. (2017) generalizes over different tasks such as opening a drawer and over different robots such as robot arms with 3 or 4 degrees of freedom. Different robots can be interpreted as different domains in which tasks should be performed. Each task and each robot have distinct neural network modules. The output of a task module is used as input for the robot module that generates the actions. Modules are trained over different task-robot combinations with the goal that each robot module can work in combination with every task module. Given an unseen task-robot combination, the corresponding modules that were learned in different combinations are combined to solve the new combination. The approach by Higgins et al. (2017) uses unsupervised learning for a deep neural network to identify the important state features for a task from a video-camera input. The policy of a task, such as grasping an object with a robot arm, is learned based on these features instead of the direct state information. If the task has to be performed in a different environment, e.g. picking up an object in a red room instead of the blue room where it was previously encountered, but where the important state features can be still extracted, e.g. the position of the object, then the previously learned policy can be immediately applied. The goal of the OA-IGE is different to the domain adaptation approaches. Its goal is to adapt its policy to different objectives in a single domain or environment.

Other approaches do zero-shot learning by dividing tasks in a sequence of sub tasks, e.g. go to room A, get object X, go to room B. The policies for sub tasks can be recombined for a new task, e.g. go to room B, get object X, go to room A. In (Andreas, Klein, & Levine 2016), each task is described by policy sketches, that are a list of symbols where each symbol is associated with a sub task. The approach learns for each symbol a sub policy to solve it, without direct information about the sub goals for this symbol. Given the policy sketch for a new task, the approach can immediately recombine its sub-policies accordingly. Oh et al. (2017) use for the sub tasks parameterized policies, which can generalize from from one sub-goal to another, e.g. from learning to pick up object X to picking up object Y. Given a sequential list of sub-goals for a task, the agent learns their corresponding sub-policies and how to combine them. In a new task the agent can immediately recombine its sub-policies and generalize to unseen sub-goals. In principle,

the tasks in OA-MDPs can also be divided in sub-tasks, such as go first to position A, then go to position B. However, the objective functions do not provide information about which of these sub-tasks should be performed in which sequence. Nonetheless, the OA-IGE could be used as a part of such sub-task frameworks as a way to generalize sub-tasks over different objectives based on the reward and time trade-off.

Schaul et al. (2015) proposed an approach based on the Horde architecture (Section 1.3). The tasks are to reach different goal states in grid-world environments. The approach learns value functions a function approximator that takes also the goal position as input. After learning the value functions to reach different goal states, the approach generalizes new value functions for new goal states. In OA-MDPs the optimal goal states change also from one objective to another, but which goal state is optimal for a certain objective and its position is not given by the formulation.

A general alternative approach to solve OA-MDPs is model-based reinforcement learning. Model-based approaches either have or learn a model for an environment, i.e. its transition probability and the reward function. Given a new objective, the framework can learn a policy to optimize the objective by using simulations of the task by the model. If the model is correct, the approach can guarantee convergence upon the optimal policy for any MDP and objective. In contrast, the OA-IGE cannot guarantee to find the optimal policy for all MDPs and objectives.

Nonetheless, the model-free approach of the OA-IGE can provide advantages in complex tasks. First, the learning of the correct model is problematic for high-dimensional state spaces and may not succeed (Tangkaratt, Morimoto, & Sugiyama 2016). Without a correct model, the model-based approach may not find a good policy. Second, even if a correct model exists, finding the optimal policy for a new objective can still be computational demanding. For example, in the recent breakthrough by Silver, Schrittwieser, et al. (2017), where the AlphaGo framework can beat the human world champion, the agent has a perfect model of the game. However, it takes the agent 29 millions of played games against itself to learn its final policy. The task of playing Go is different from the tasks used to evaluate the OA-IGE, but it shows that even if an optimal model is given, finding a good policy for a new task can have a high computational demand for a model-based framework. In contrast, the computational demand of the OA-IGE to adapt to a new task is relatively small. Given a new objective, it only has to approximate the outcome for the new objective for each of its modules policies. The computation of the outcome either based on the decoded expected trajectory (DOA-IGE) or on the learned expectation over their reward sum and length (POA-IGE) and has a low computational cost.

Multi-Objective Reinforcement Learning

The IGE, especially the OA-IGE, has strong connections to the field of multi-objective reinforcement learning (MORL) (Roijers et al. 2013; Liu, Xu, & Hu 2015). MORL is an extension to standard reinforcement learning where the reward function returns a vector instead of a scalar. Each element of the reward vector represents a different objective that the agent has to take into account. For example, a mining robot could mine not only a single mineral, but several different minerals at once. It can learn the way to different mining sites, which provide a different amount of minerals for each type. The reward function returns how much minerals could be mined at a certain site for each mineral

type.

Multi-objective MDPs (MOMDPs) are defined as a standard MDP:

$$\text{MOMDP}(A, S, T, \vec{R}) \ ,$$

with the distinction that the reward function $\vec{R} : S \times A \times S \mapsto \mathbb{R}^N$ returns a vector. As a result, the value function changes from a scalar function to a vector function:

$$\vec{V}^{\pi}(s_t) = \text{E}_{\pi}\left[\sum_{k=0}^{\infty} \gamma^k \vec{r}_{t+k}\right] \ ,$$

where $\vec{V}$ and $\vec{r}$ denote the vectorized version of the value and reward in MOMDPs. As in single-objective reinforcement learning the value function can be recursively formulated in form of a Bellman equation, and a Q-function can be defined in the same way. In difference to single-objective reinforcement learning, the optimal value cannot be easily defined, because the maximum over vectors does not exist. For example, three mining sites could exist in a mining task with two mineral types. Each site gives a return of $\vec{r}_1 = [0, 3]^{\intercal}$, $\vec{r}_2 = [1, 1]^{\intercal}$, and $\vec{r}_3 = [3, 0]^{\intercal}$. Which of the three sites is optimal is not immediately clear, because none has the maximum reward for both mineral types. However, a function that maps the vectors to a scalar value can be used to define the optimal solution. It could, for example, compute how much profit can be made by selling both mined mineral types. In MORL such functions are called scalarization functions. They map a multi-objective value $\vec{V}$ to a scalar value:

$$V_{\omega}^{\pi} = f(\vec{V}^{\pi}(s), \omega) \ ,$$

where ω is a weigh vector parameterizing f. The goal of the agent is maximize the scalarized value function: $\pi^*(s) = \text{argmax}_{\pi} V_{\omega}^{\pi}$. In the mining example the scalarization function could be a weighted sum over the vector components, where the weights for each component defines its selling price: $V_{\omega}^{\pi}(s) = \sum_i \omega_i \vec{V}_i(s)$.

Roijers et al. (2013) provide a taxonomy of MORL problems and their associated algorithms. Problems and algorithms can be differentiated between single-policy and multi-policy approaches. In single-policy approaches, the weights of the scalarization function are known during the learning phase. Therefore, a single optimal policy exists which optimizes the MOMDP. In principle, such scenarios could be solved by single-objective reinforcement learning that solves the problem for the scalarized value function. Nonetheless, the multi-objective approach of learning a vectorized value function can be helpful in scenarios where it is easier to learn the individual elements of the values, for example if their state representation can be simplified. In multi-policy approaches, the agent does not know the weights ω during the learning phase. It should therefore learn several policies, that can then be used during a selection phase to determine the optimal, or most appropriate, policy for a given weight vector ω.

A second property used to differentiate MORL approaches is the form of the scalarization function. The functions can be either linear or monotonically increasing. Linear scalarization functions are a weighted sum over the value dimensions: $V_{\omega}^{\pi}(s) = \sum_i \omega_i \vec{V}_i(s)$. As a result, the potential set of policies that result in the optimal policy for each possible weight vector ω is restricted to a convex hull set (Roijers et al. 2013). Therefore, multi-policy algorithms need only to learn such convex hull policies to guarantee optimality. In

difference, monotonically increasing scalarization functions can be non-linear. Their only restriction is that if all other value dimensions are fixed, then increasing any specific value dimension results in a higher scalar value. As a result, the set of policies to guarantee optimality have to be a Pareto optimal set, which includes the convex hull set (Roijers et al. 2013).

Please note, that approaches are also differentiated between the types of policies, that the algorithms learn. They can be either be deterministic or stochastic. Depending on the type and the other properties of an MOMDP, the set of optimal policies changes, i.e. if they are stationary or non-stationary. Please refer to Roijers et al. (2013) for more information.

The IGE can be understood as a specialized MORL algorithm having two objective dimensions. The first is the regular scalar reward function R. The second is the number of steps until a goal-state is reached. Both could be included in a vectorized reward function using $\vec{R}(s_t, a_t, s_{t+1}) = [R(s_t, a_t, s_{t+1}), -t]^\intercal$. The number of steps are multiplied by -1 because the general goal is to minimize and not to maximize them. In the case of OA-MDPs the objectives $J = (f_1, f_2, \ldots)$ represent different scalarization functions and weights ω. The OA-IGE is in this form a multi-policy algorithm. It learns several policies with its γ-modules. Depending on the current scalarization function f_k from J, it selects the most appropriate policy to maximize the function.

In terms of the scalarization functions, it can be shown that the OA-IGE learns in deterministic, goal-only-reward environments the convex hull set of policies. This insight can be derived from the definition of the IGE choice set in these environments (Theorem 7). It is therefore guaranteed to learn all optimal policies for linear scalarization functions. If the objective functions J are monotonically increasing scalarization functions, then the OA-IGE can not guarantee to learn all optimal policies, because the IGE policy set is not guaranteed to be the Pareto optimal set of all policies. For example, in the deterministic, goal-only-reward tasks the OA-IGE cannot learn to reach goal state g_4 (Fig. 4.2, c), which is part of the Pareto optimal set.

Existing multi-policy algorithms in MORL (Liu, Xu, & Hu 2015) use either a convex hull approach or a varying parameter approach. The convex hull approach by Barrett & Narayanan (2008) identifies the convex hull set of policies in an MOMDP. This allows the algorithm to guarantee optimality for linear scalarization functions similar to the OA-IGE. In difference, the OA-IGE also learns policies outside the convex hull set and can therefore also adapt to non-linear scalarization functions, although without guaranteeing optimality. Furthermore, the convex hull approach is a dynamic programming approach requiring a model of the MOMDP, whereas the OA-IGE is model-free.

Varying parameter approaches such as by Shelton (2001) learn different policies by varying the parameters ω of the scalarization function and learning the optimal policies for them. Then if a new target set of parameters ω is given, it is for example possible to select the learned policy with the closest parameters. Such algorithms need the knowledge of the set of scalarization functions J, and they are not guaranteed to be optimal. In the case of the OA-IGE, the set of scalarization functions can be unknown.

In summary, the IGE can be interpreted as a special case of an MORL approach, where the first objective is the reward and the second the number of steps. The approach to learn a set of policies based on different discountings has not been explored in MORL and represents an interesting new approach for MOMDPs that include time as a objective.

Optimality

A problematic point of the OA-IGE is that it can guarantee optimality only for a subset of tasks and objectives. This problem is based on two points. First, it can only adapt to new objectives using policies from the IGE policy set, but these might not have the optimal policy. For example, in the deterministic, goal-only-reward task the OA-IGE cannot learn to reach goal state g_4 (Fig. 4.2, c). As a result, it cannot find the optimal policy for objective f_7 that has g_4 as optimal solution (Fig. 4.4). Nonetheless, optimality can be guaranteed for a subset of goal formulations. The OA-IGE converges to the optimal policy for objectives that maximize the exponentially discounted reward sum $\mathrm{E}\left[\sum_{t=0}^{T-1} \gamma^t r_t\right]$. Depending on the discount factor γ, the Q-values of the corresponding γ module will converge to the optimal value function for it (Watkins & Dayan 1992; Tsitsiklis 1994). This includes also the case of maximizing the expected total reward sum $\mathrm{E}\left[\sum_{t=0}^{T-1} r_t\right]$, because this is the same objective as for $\gamma = 1$. Most interestingly, it is possible to prove its convergence upon the optimal policy for the average reward $\frac{r}{n}$ in deterministic, goal-only-reward MDPs as discussed in Chapter 5. For other objectives, the OA-IGE can be viewed as a heuristic that does not guarantee optimality, but that produces often good results with the ability to adapt immediately to a new objective.

The second point why optimality generally cannot be guaranteed is that OA-IGE uses the approximation $\mathrm{E}_{\pi_\gamma,s_0}[f(h)] \approx f(\mathrm{E}\pi_\gamma, s_0[h])$ or $\mathrm{E}_{\pi_\gamma,s_0}[f(R,T)] \approx f(\mathrm{E}_{\pi_\gamma,s_0}[R], \mathrm{E}_{\pi_\gamma,s_0}[T])$ to select its most appropriate policy for an objective. The approximation is only correct for a subset of tasks and objectives (Section 4.2). For other conditions, a wrong approximation might result in selecting an underperforming γ-policy. A solution to this problem is to use a similar approach as for the Context Adaptive IGE (Section 3.4.1) that learns a mapping from a context to the most appropriate γ-module. A map $(G : J \times S \mapsto \Gamma)$ from the objective function $f_k \in J$ and the agent's states $s \in S$ to the γ-module with the most appropriate policy could be learned for the OA-IGE. A value-based approach similar to the Q-IGE (Algorithm 3.4) could be used. The expected outcome F for using a module starting in state s for objective f_k could be learned:

$$F(k, s, \gamma) = \mathrm{E}_{\pi_\gamma}[f_k(R, N)] \ .$$

This would allow the OA-IGE to find the most appropriate module for a certain objective, because the value function F could learn the true expectation over outcomes. Zero-shot learning would still be possible by initializing F for a new objective f_k by using the approximation:

$$F_{init}(k, s, \gamma) = f_k(\mathrm{E}_{\pi_\gamma}[R], \mathrm{E}_{\pi_\gamma}[N]) \ .$$

Memory Demand

Comparing the memory demand of the OA-IGE to that of the classical time-dependent Q-learning approach shows that it needs less memory, depending on the number of objectives. Both approaches used a tabular representation of their value functions. The DOA-IGE needed to represent $|\Gamma| \times |S| \times |A| = 90 \times 100 \times 4 = 36{,}000$ values. The POA-IGE has two extra value functions for the expected reward sum and number of steps, but it also had less γ-modules. It needed $Q_\gamma : (|\Gamma| \times |S| \times |A|) + R_\gamma : (|\Gamma| \times |S|) + T_\gamma : (|\Gamma| \times |S|)$

$= 45 \times 100 \times 4 + 45 \times 100 + 45 \times 100 = 27{,}000$ values. The time-dependent Q-learning approaches needed to store values over the objectives, time steps, states and actions: $|J| \times T \times |S| \times |A|$, which resulted in $9 \times 30 \times 100 \times 4 = 108{,}000$ values. This is a factor of 3 larger than the memory demand for the DOA-IGE. Furthermore, the list of objectives J is in principle infinite, making it impossible for the Q-learning approach to store the values for each possible objective function. The OA-IGE algorithms do not store values for certain objectives and can generalize therefore to an infinite number of objectives.

4.6 Conclusion

The OA-IGE has the ability to adapt immediately to different and unseen objectives in an environment. Objectives define different requirements by the amount of reward that should be acquired and the time that should be invested. For example, the objective could be to maximize the total amount of reward, but to use only a certain maximum number of time steps. The ability of the IGE to learn different policies for an environment, each representing a different solution for a trade-off between the reward amount and invested time, allows it to adapt to such objectives. It can select its most appropriate policy for a given objective. The decision, which policy to use for an objective, is solved by approximating each policy's outcome for the objective. The Decoding OA-IGE uses the Q-values of its γ-modules to decode the expected future reward trajectory for each policy, but its application is limited to deterministic, goal-only-reward environments. The Prediction OA-IGE learns for each of its module's policies the expected future reward sum and the expected number of steps. It can be applied to any OA-MDP. Although, convergence upon the optimal policy can only be guaranteed for specific tasks and objectives, the OA-IGE can adapt immediately to new and unseen objectives by selecting an appropriate policy. In contrast, classical Q-learning approaches need several observations to adapt to a new objective.

Chapter 5

Average Reward Optimization

The tasks to which the IGE is applied are episodic in nature. The objective is usually to gain the highest reward during an episode, or under some time restriction. Nonetheless, in tasks that are repeated in a loop it would be more beneficial to optimize the average reward over the time steps. For example, a mining agent has to collect resources and bring them back to a storage facility. The agent has to repeat this task over and over in a loop. In this case, the optimal strategy would not be to collect the maximum amount of resources in each episode, because this could take the agent a very long time. Instead, the agent should optimize the average amount of the collected resources per time step. The average-reward objective results in the largest amount of reward for the invested time.

The objective is formulated as a function over the number of steps T that go towards infinity:

$$\lim_{T\to\infty} \frac{1}{T} \mathrm{E}\left[\sum_{t=0}^{T-1} r_t\right] . \tag{5.1}$$

The most widely utilized methods to optimize this objective are policy-based algorithms (Deisenroth, Neumann, & Peters 2011), but value-based methods, such as the IGE, have also been proposed (Mahadevan 1996). The value-based algorithms optimize the average reward by learning two entities. First, they learn an average-reward adjusted value function that is analogous to the value function for the discounted reward objective (Eq. 1.3, page 10). Second, the value function is based on an estimate of the average reward gained by the agents current policy. The estimation has to be learned in parallel to the value function. The agents policy is then defined over the value function by choosing the actions that maximize it.

This chapter explores the ability of the IGE to optimize average reward. The IGE allows optimizing the average reward per episode in deterministic, goal-only-reward MDPs. This objective is equal to optimizing the average reward over all time steps (Eq. 5.1) in tasks where the start state is the same for each episode. The IGE provides a new way to optimize average reward. In contrast to the average adjusted approaches, it does not need to learn an estimate of the average reward of the agents policy. Instead, it learns different policies to reach some of the goal states in an MDP (Section 2.2). It can identify which policy results in the highest average reward per step in an episode. Two algorithms are introduced that differ in the method used to identify the optimal policy. One is using the ability of the IGE to decode the expected reward and number of steps for its learned

policies. Based on the decoded information the average reward of each policy is calculated to detect the one that maximizes it. The other approach identifies the optimal policy by directly comparing the values of the modules with each other. For deterministic, goal-only-reward MDPs it can be shown that the IGE is guaranteed to learn the policy to the goal state that results in the optimal average reward. Moreover, it is possible to define the number of necessary modules and their γ factors for which the IGE is guaranteed to be optimal.

This chapter is organized in the following way. First, existing approaches using average-adjusted values are introduced in Section 5.1, followed by a description of the Average-Reward IGE (AR-IGE) in Section 5.2. The two AR-IGE variants and their optimality proofs are discussed in Sections 5.3 and 5.4. The results of an experimental comparison between the AR-IGE and the existing methods are given in Section 5.5. The experiments were performed in two task domains with randomly generated MDPs. The results show that the AR-IGE outperforms the existing approaches, especially in MDPs that change over time. The discussion in Section 5.6 provides details of the difference between the AR-IGE and the existing methods in terms of their objective, i.e. which type of average reward they optimize. In addition, it describes the relationship between the number of γ-modules and the performance of the AR-IGE.

5.1 Existing Average-Adjusted Approaches

Various approaches to optimize the average-reward objective exist. Policy-based approaches such as policy-gradient methods (Deisenroth, Neumann, & Peters 2011) are the most popular. Nonetheless, value-based approaches similar to Q-learning (Definition 3) also exist. This section introduces the value-based approaches.

Average reward optimization is usually formulated for continuous MDPs instead of episodic MDPs. In such MDPs the average reward for a policy π is defined by:

$$\rho^{\pi} = \lim_{T \to \infty} \mathrm{E}_{\pi} \left[\frac{1}{T} \sum_{t=0}^{T-1} r_t \right] , \tag{5.2}$$

where the expectation is over the policy π. The goal is to find the policy π^* that results in the highest average reward: $\rho^{\pi^*} \geq \rho^{\pi}$ for all π.

Value-based algorithms solve this problem for the class of unichain MDPs. It is necessary to first introduce some other concepts before unichain MDPs can be defined. These concepts describe structural aspects of MDPs in respect to possible transitions between states under some policy π. Policies are assumed to be stationary and deterministic, i.e. they do not change and map only one action to each state ($\pi : S \mapsto A$). Under this assumption, a state s_a is called to communicate with another state s_b if a policy exists that can reach s_b in one or more steps from s_a. A state is recurrent under a policy if the probability of visiting it again by starting from it is 1. Such states are visited again and again in a kind of a loop. In contrast, transient states might be visited for a certain number of times but then not anymore. Based on these definitions a recurrent or ergodic set of states can be defined. All states in such a set are recurrent and communicate with each other, but not with any other state outside the set.

Using the introduced concepts the class of unichain MDPs can be defined. An MDP is unichain if for each possible policy only one set of recurrent states exists and possibly some transient states. As a result, a policy might start in some states and then loop over the set of recurrent states. This allows the existence of a single average reward ρ^π that is associated with a policy in such MDPs. Multichain MDPs have for at least one possible policy more than one set of recurrent states. In such a case the average reward depends on the set of recurrent states over which the agent loops. Therefore, a single average reward ρ^π cannot be defined for such MDPs.

Analogues to the value function based on the expected discounted reward sum, a recursive value function for the average reward can be defined. In can be proven that for any unichain MDP a value function V^* and a scalar ρ^* exist that satisfy the following equation (Puterman 1994):

$$V^*(s_t) = \max_{a_t \in A} \left(R(s_t, a_t) - \rho^* + \sum_{s_{t+1} \in S} T(s_t, a_t, s_{t+1}) V^*(s_{t+1}) \right) . \tag{5.3}$$

The greedy policy π^*, i.e. the action that maximizes the right hand side, results in the optimal average reward ρ^*.

Several model-free and model-based algorithms exist that use the average-adjusted value function to optimize average reward (Mahadevan 1996). The IGE has been experimentally compared to some of the model-free approaches that use a TD approach similar to Q-learning for discounted value functions. All of the methods learn an average-adjusted Q-function. The Q-function follows the same principle as the state-value function (Eq. 5.3). All methods (Algorithm 5.1) start with an arbitrarily initialized Q-function. The Q-values are updated after a transition from s_t to s_{t+1} with action a_t, and the observation of reward r_t:

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha_Q \left(r_t - \rho + \max_{a_{t+1}} Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t) \right) ,$$

where $\alpha_Q \in [0, 1]$ is the learning rate, and ρ is a stochastic estimation of the average reward. The estimation ρ is also learned from the observations of transitions. It is updated after transitions that used the greedy action based on the current Q-function. The difference between the average-adjusted algorithms is their method to update the estimate ρ.

R-Learning (Schwartz 1993) was the first variant of these algorithms. Its average reward is updated by:

$$\rho \leftarrow \rho \; + \; \alpha_\rho \left(r_t - \rho + \max_{a_{t+1}} Q(s_{t+1}, a_{t+1}) - \max_{a_t} Q(s_t, a_t) \right) ,$$

where $\alpha_\rho \in [0, 1]$ is the learning rate for the average reward. The Semi-Markov Average Reward Technique (SMART) (Das et al. 1999; Mahadevan, Marchalleck, et al. 1997) calculates the average reward by summing over all perceived rewards during the experiment and dividing the sum by the total number of time steps:

$$\rho \leftarrow \frac{\sum_{t=0}^{T-1} r_t}{T} ,$$

Algorithm 5.1: Average-adjusted algorithms for continuous average-reward MDPs

Input:
Learning rate: $\alpha_Q \in [0, 1]$
Discount factor: $\alpha_\rho \in [0, 1]$

initialize $Q(s, a)$ arbitrarily
initialize ρ arbitrarily
initialize state s
repeat (for each step)
  $a \leftarrow$ choose an action for s derived from $Q(s, a)$ (e.g. ϵ-greedy)
  r, $s' \leftarrow$ take action a, observe outcome
  $Q(s, a) \leftarrow Q(s, a) + \alpha_Q\big(r - \rho + \max_{a'} Q(s', a') - Q(s, a)\big)$
  `// ρ is update for steps that follow the greedy policy`
  if $a = \text{argmax}_u\, Q(s, a)$ **then**
    $\rho \leftarrow$ update estimate of average reward (depends on the algorithm)
  $s \leftarrow s'$
until *termination*

where T is the total number of time steps up to the current time point. Relaxed SMART (rSMART) (Gosavi 2004) is variant of SMART for which a convergence proof of the algorithm was developed. It uses a relaxed update of the average reward:

$$\rho \leftarrow \rho + \alpha_\rho \left(\frac{\rho \cdot (T-1) + r_t}{T} - \rho \right) .$$

Robbins-Monro SMART (rmSMART) (Gosavi 2004) is a further variant of SMART which uses a Robbins-Monro update (Robbins & Monro 1951) of the average reward:

$$\rho \leftarrow \rho + \alpha_\rho (r_t - \rho) .$$

A more recent approach is CSV-Learning by Yang et al. (2016) where ρ is not updated during the learning but set to an initial guess. This algorithm can outperform existing approaches, but it is excluded from the analysis because of its required initial guess of the average reward.

5.1.1 Average-Reward Optimization in Episodic MDPs

The average-adjusted algorithms introduced above optimize the average reward in continuous tasks. In contrast, the objective of the AR-IGE is the optimization of the average reward per episode in episodic MDPs. The average reward per episode for a policy π is given by:

$$\rho_k^\pi = \mathrm{E}_\pi \left[\frac{1}{T_k} \sum_{t=0}^{T_k - 1} r_{k,t} \right] , \tag{5.4}$$

where T_k is the number of steps, and $r_{k,t}$ is the reward at step t of episode k.

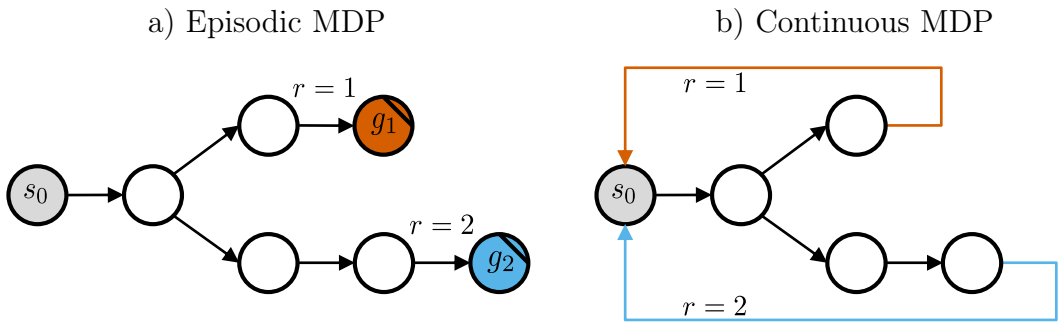


Figure 5.1: Episodic MDPs can be reformulated as continuous MDPs by introducing transitions back to a start state instead of transitions to the goal states. This allows the use of average-adjusted methods, such as R-Learning, for such tasks.

Average-adjusted methods can also be applied to episodic tasks that are continuously repeated. This can be done simply by adapting the task so that instead of a transition into a goal state the transition brings the agent to the initial state of the next episode (Fig. 5.1). As a result, the average-adjusted methods optimize the average reward over all steps (Eq. 5.2). In terms of the episodic representation of the MDPs they optimize the average reward over the steps of all episodes:

$$\rho^{\pi}_{exp} = \lim_{K\to\infty} \mathrm{E}_{\pi}\left[\frac{1}{\sum_{k=1}^{K} T_k} \sum_{k=1}^{K} \sum_{t=0}^{T_k-1} r_{k,t}\right] , \tag{5.5}$$

where K is the number of episodes, and T_k is the number of steps in episode k.

Generally, optimizing the average reward per episode is different from optimizing the average reward over the steps of all episodes. Nonetheless, both are the same for tasks where the initial state is the same for each episode (Theorem 16). Such tasks can therefore be optimized by either approach, the AR-IGE or the average-adjusted algorithms. The difference between average reward per episode and over all steps is further addressed in the discussion section.

Theorem 16. *The expected average reward over episodes (Eq. 5.5) for a greedy policy π in an episodic MDP with a deterministic transition function, and a single start state s_0 is equal to its expected average reward per episode (Eq. 5.4):*

$$\forall k : \rho^{\pi}_{k} = \rho^{\pi}_{exp} ,$$

where k is the index over episodes.

Proof. In MDPs with deterministic state transitions, the number of steps $T_k(\pi, s_0)$ resulting from a greedy policy π starting in the same initial state s_0 is the same for each episode:

$$\forall k : \; T_k(\pi, s_0) = T \; .$$

Based on this, the average reward per individual episode (Eq. 5.4) becomes:

$$\rho^{\pi}_{k} = \mathrm{E}_{\pi}\left[\frac{1}{T_k} \sum_{t=0}^{T_k-1} r_{k,t}\right] = \frac{1}{T} \sum_{t=0}^{T-1} \mathrm{E}\left[r_{k,t}\right] \; .$$

Furthermore, it shows that the expected average reward over the steps of all episodes (Eq. 5.5) is equal to the average reward per episode:

$$\rho^{\pi}_{exp} = \lim_{K\to\infty} \mathrm{E}_{\pi}\left[\frac{1}{\sum_{k=1}^{K} T_k}\sum_{k=1}^{K}\sum_{t=0}^{T_k-1} r_{k,t}\right] = \lim_{K\to\infty} \mathrm{E}_{\pi}\left[\frac{1}{\sum_{k=1}^{K} T}\sum_{k=1}^{K}\sum_{t=0}^{T-1} r_{k,t}\right]$$
$$= \lim_{K\to\infty} \frac{1}{KT} K \sum_{t=0}^{T-1} \mathrm{E}_{\pi}\left[r_{k,t}\right] = \frac{1}{T}\sum_{t=0}^{T-1} \mathrm{E}_{\pi}\left[r_{k,t}\right] = \rho^{\pi}_{k}.$$

□

5.1.2 Episodic Average-Adjusted Algorithms

In deterministic, goal-only-reward MDPs with a single start state, an alternative form of the average-adjusted algorithms can be defined that operates on the episodic formulation of the MDPs (Fig. 5.1, a). The alternative form uses the observation that the value of the start state is zero ($V^{\pi^*}(s_0) = 0$) for the optimal policy π^* in such MDPs. This results from the zero-rewards that are given in trajectories, besides the last transition into the goal state:

$$\begin{aligned} V^*(s_0) &= r_0 - \rho^* + r_1 - \rho^* + \ldots + r_{T-1} - \rho^* \\ &= r_{T-1} - (T)\rho^* \\ &= r_{T-1} - (T)\tfrac{r_{T-1}}{T} \\ &= 0\ , \end{aligned} \tag{5.6}$$

where $\rho^* = \frac{r^*}{T}$ is the average reward for the optimal trajectory, where T is its number of steps and r_{T-1} is the final reward. This insight can be used to formalize an episodic variant of the average-adjusted algorithms (Algorithm 5.2). For transitions into goal states ($isTerminal = true$), the continuous version of the algorithms would make a transition to the start state of the next episode and uses its value to update the last Q-value. The episodic variant uses the assumption that the optimal value of the start state is zero (Eq. 5.6) and updates the last Q-value by it, instead of using the Q-value of the next start state.

The performance of both versions, the continuous and the episodic form, of the average-adjusted algorithms were evaluated. The results show, that for some problems the continuous form is better and for others the episodic form (Section 5.5).

5.1.3 Conclusion

Model-free, value-based algorithms use the concept of an average-reward adjusted Q-function to optimize average reward. The value function is learned with a similar TD approach as classical Q-learning for the discounted value function. In addition the algorithms have to estimate the average reward ρ of its policy from observations. Although the methods were developed for continuous tasks, they can be applied to episodic tasks that are repeated. Their objective is the optimization of the average reward over the steps of all episodes. In contrast, the AR-IGE, which is introduced in the following section, optimizes the average reward per episode. The objectives of the AR-IGE and the average-adjusted algorithms are different for general MDPs. Nonetheless, the objectives

Algorithm 5.2: Average-adjusted algorithms for episodic average-reward MDPs

Input:
Learning rate for Q-values: $\alpha_Q \in [0, 1]$
Learning rate for average reward: $\alpha_\rho \in [0, 1]$

initialize $Q(s, a)$ arbitrarily
initialize ρ arbitrarily
repeat (for each episode)
  initialize state s
  repeat (for each step in episode)
    $a \leftarrow$ choose an action for s derived from $Q(s, a)$ (e.g. ϵ-greedy)
    r, $s', isTerminal \leftarrow$ take action a, observe outcome
    `// if the goal state is reached assume a value of 0 for the next state`
    if *isTerminal* = *false* **then**
      $Q(s, a) \leftarrow Q(s, a) + \alpha_Q\big(r - \rho + \max_{a'} Q(s', a') - Q(s, a)\big)$
    else
      $Q(s, a) \leftarrow Q(s, a) + \alpha_Q\,(r - Q(s, a))$
    end
    `// ρ is update for steps that follow the greedy policy`
    if $a = \text{argmax}_u\, Q(s, a)$ **then**
      $\rho \leftarrow$ update estimate of average reward (depends on the algorithm)
    $s \leftarrow s'$
  until s *is terminal-state*
until *termination*

are equal for deterministic MDPs where the initial state is the same for each episode. These MDPs are therefore used to compare the algorithms in two problem domains in Section 5.5.

5.2 The Average-Reward Independent γ-Ensemble

The Average-Reward Independent γ-Ensemble (AR-IGE) is a IGE framework to optimize the average reward per episode in episodic tasks (Fig. 5.2). It learns different policies by its γ-modules (Section 2.2). At the beginning of an episode, it selects the γ-policy that will optimize the average reward:

$$\tilde{\gamma} = \underset{\gamma}{\text{argmax}} \frac{\text{E}_{\pi_\gamma, s_0}[R]}{\text{E}_{\pi_\gamma, s_0}[T]},$$

where R is the reward sum of the episode and N is the number of steps. For exploration purposes, it uses an ϵ-greedy selection of the active module.

This chapter introduces two AR-IGE variants to identify the policy with the highest average reward. The first is the Decoding AR-IGE (DAR-IGE). It decodes the expected

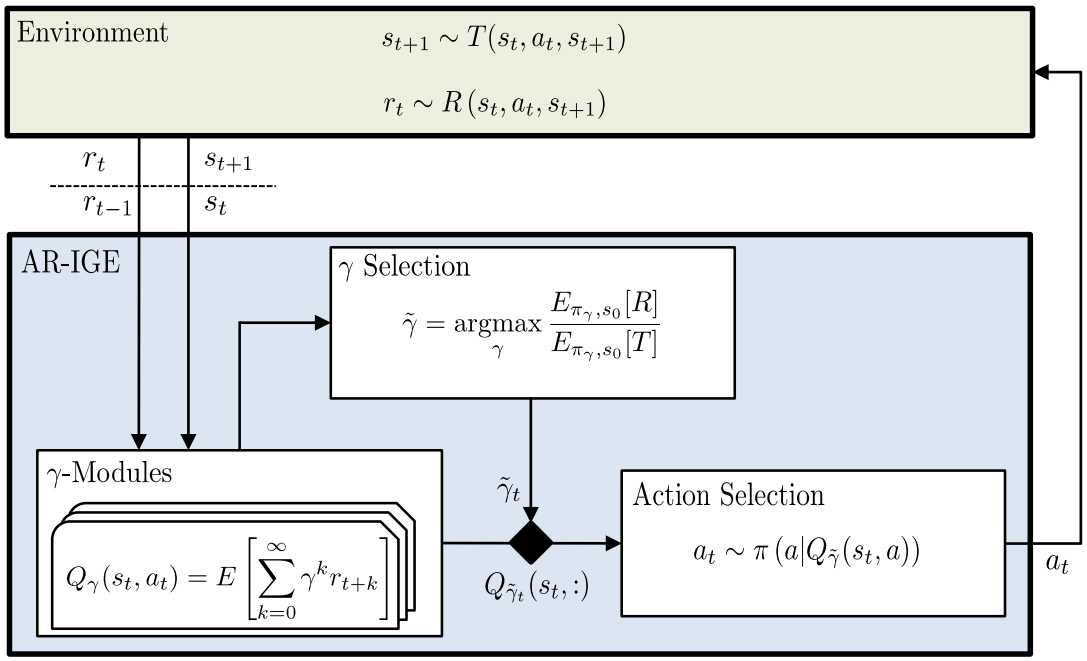


Figure 5.2: The general framework of the Average-Reward IGE (AR-IGE). The goal of the agent is to maximize the average reward per episode.

reward and the expected number of steps from the Q-values of the γ-modules to compute for each module the expected average reward. The decoded information can also be used to guide the exploration of the AR-IGE. The second approach is the Value Quotient AR-IGE (VQAR-IGE). It compares the Q-values of its modules directly with each other to identify the module with the best policy. It needs fewer modules compared to the Decoding AR-IGE. However, it cannot identify further information about its policies, besides identifying the best.

The AR-IGE framework differs in several ways from the average-adjusted approaches (Section 5.1). First, it does not learn an estimate of the average reward ρ^π. In addition, instead of learning only the optimal policy and its average reward, it learns several possible policies for a task and can compute their average reward. This can be useful, for example, if the algorithm has to adapt to changes in the MDP. The AR-IGE is able to quickly switch to another policy if the currently optimal policy becomes suboptimal.

Both AR-IGE variants are guaranteed to converge to the optimal average reward policy per episode. Two points have to be proven to show this. First, one of γ-modules has to converge to the optimal average-reward policy. Second, the γ-module with the optimal policy has to be identifiable. The first point is proven in the next section. Afterward, the DAR-IGE and the VQAR-IGE are introduced with the proofs that they can identify the module with the optimal policy.

The DAR-IGE was aleady introduced in (Reinke, Uchibe, & Doya 2017) with a proof of its optimality. However, the proof required the assumption that the IGE has an infinite number of γ-modules with uniformly distributed discount factors between 0 and 1. The proof presented in this chapter requires only a finite number of modules and it allows to define the required discount factors.

5.2.1 Existence of the Optimal Policy

To guarantee that the AR-IGE is optimal, some of its γ-modules have to learn the optimal average-reward policy per episode. Theorem 17 proves this point. Moreover, the theorem defines precisely which γ-modules, i.e. which discount factors Γ, are necessary to guarantee optimality. The proof makes use of the concept of choices (Definition 4, page 37) in deterministic, goal-only-reward MDP. It assumes that the Q-functions of all γ-modules converged to their optimal values Q^*_γ as proven by Watkins & Dayan (1992) and Tsitsiklis (1994).

Theorem 17. *For any deterministic, goal-only-reward MDP exists a γ-region Γ^* for which all its discount factors γ^* encode the optimal average reward choice c^*. The region is defined by:*

$$\Gamma^* = \left\{ \gamma^*_L = \frac{n^* - 1}{n^*} < \gamma^* < \gamma^*_U = \frac{n^*}{n^* + 1} \right\} ,$$

where γ^*_L and γ^*_U are the lower and upper border of the region, and n^* is the number of steps of the the optimal choice c^*.

Proof. The γ region for which the optimal choice c^* is the solution given all other possible choices in the MDP is defined by (Theorem 3, page 38):

$$\gamma^*_L = \max_{\forall c_i < c^*} \gamma_E(c_i, c^*) < \gamma^* < \gamma^*_U = \min_{\forall c_j > c^*} \gamma_E(c^*, c_j) ,$$

where $\gamma_E(c_x, c_y)$ is the discount factor for which the choices c_x and c_y have the same value (Eq. 2.6, page 38). The lower bound γ^*_L can be shown to be equal to $\frac{n^*-1}{n^*}$, starting with:

$$\begin{array}{rcll} \gamma^*_L(n^*) & = & \max_{\forall c_i < c^*} \gamma_E(c_i, c^*) & \\ & = & \max_{r^*, r_i, n_i} \left(\frac{r_i}{r^*}\right)^{\frac{1}{n^* - n_i}} & \text{s.t. } n_i < n^*,\ \frac{r^*}{n^*} \geq \frac{r_i}{n_i},\ n_i, n^* \in N^+ , \end{array} \tag{5.7}$$

with the choices $c_i = (r_i, n_i)$, and the optimal average reward choice $c^* = (r^*, n^*)$. The number of steps of the optimal choice n^* is assumed to be given, i.e. the goal is to find the lower region border γ^*_L given that the optimal choice has a certain number of steps n^*. The maximum can be solved by using the constraints that all choices c_i have fewer steps than the optimal choice ($n_i < n^*$), and that their average reward is smaller ($\frac{r^*}{n^*} \geq \frac{r_i}{n_i}$). Moreover, the number of steps can only be positive natural numbers ($n_i, n^* \in N^+$). First, the reward r_i is identified which maximizes Eq. 5.7:

$$\begin{array}{rcll} r_i^{max} & = & \text{argmax}_{r_i} \left(\frac{r_i}{r^*}\right)^{\frac{1}{n^* - n_i}} & \\ & = & \text{argmax}_{r_i} \left(\frac{r_i}{r^*}\right) & \\ & = & \text{argmax}_{r_i}\, r_i & \text{s.t. } n_i < n^*,\ \frac{r^*}{n^*} \geq \frac{r_i}{n_i},\ n_i, n^* \in N^+ . \end{array} \tag{5.8}$$

By reformulating the average reward constraint:

$$\frac{r^*}{n^*} \geq \frac{r_i}{n_i} \quad \Leftrightarrow \quad r_i \leq n_i \frac{r^*}{n^*} ,$$

the reward r_i^{max} (Eq. 5.8) is given by:

$$r_i^{max} = n_i \frac{r^*}{n^*} .$$

Using this result for the lower γ-region border γ_L^* (Eq. 5.7) reduces the reward variables r^* and r_i from its maximum formulation:

$$\begin{aligned} \gamma_L^*(n^*) &= \max_{r^*,r_i,n_i} \left(\tfrac{r_i}{r^*}\right)^{\frac{1}{n^*-n_i}} \\ &= \max_{r^*,n_i} \left(\frac{n_i \frac{r^*}{n^*}}{r^*}\right)^{\frac{1}{n^*-n_i}} \\ &= \max_{n_i} \left(\tfrac{n_i}{n^*}\right)^{\frac{1}{n^*-n_i}} \qquad \text{s.t. } n_i < n^*,\ n_i, n^* \in N^+ \ , \end{aligned} \tag{5.9}$$

Lemma 9.1 (page 47) shows that the maximum of Eq. 5.9 is given by:

$$\gamma_L^*(n^*) = \frac{n^*-1}{n^*} \ .$$

The upper bound γ_U^* of the γ-region is equal to $\frac{n^*}{n^*+1}$. It can be derived in a similar way as the lower bound γ_L^*, starting with its definition:

$$\begin{aligned} \gamma_U^*(n^*) &= \min_{\forall c_j > c^*} \gamma_E(c^*, c_j) \\ &= \min_{r^*,r_j,n_j} \left(\frac{r^*}{r_j}\right)^{\frac{1}{n_j-n^*}} \quad \text{s.t. } n_j > n^*,\ \tfrac{r^*}{n^*} \geq \tfrac{r_j}{n_j},\ n_j, n^* \in N^+ \ , \end{aligned} \tag{5.10}$$

where the larger choices $c_j = (r_j, n_j)$, and the optimal average reward choice $c^* = (r^*, n^*)$.

The minimum (Eq. 5.10) can be derived, by first deriving the reward r_j that yields the minimum:

$$\begin{aligned} r_j^{min} &= \operatorname{argmin}_{r_j} \left(\frac{r^*}{r_j}\right)^{\frac{1}{n_j-n^*}} \\ &= \operatorname{argmin}_{r_j} \left(\frac{r^*}{r_j}\right) \\ &= \operatorname{argmax}_{r_j} r_j \\ &= n_j \tfrac{r^*}{n^*} \qquad \text{s.t. } n_j > n^*,\ \tfrac{r^*}{n^*} \geq \tfrac{r_j}{n_j},\ n_j, n^* \in N^+ \ , \end{aligned}$$

Using this result for the upper γ-region border γ_U^* (Eq. 5.10) reduces the reward variables r^* and r_j from its minimum formulation:

$$\begin{aligned} \gamma_U^*(n^*) &= \min_{r^*,r_j,n_j} \left(\frac{r^*}{r_j}\right)^{\frac{1}{n_j-n^*}} \\ &= \min_{r^*,n_j} \left(\frac{r^*}{n_j \frac{r^*}{n^*}}\right)^{\frac{1}{n_j-n^*}} \\ &= \min_{n_j} \left(\tfrac{n^*}{n_j}\right)^{\frac{1}{n_j-n^*}} \qquad \text{s.t. } n_j > n^*,\ n_j, n^* \in N^+ \ , \end{aligned} \tag{5.11}$$

Lemma 9.2 (page 48) shows that the minimum of Eq. 5.11 is given by:

$$\gamma_U^*(n^*) = \frac{n^*}{n^*+1} \ .$$

Finally, after showing the borders of the γ region (γ_L^*, γ_U^*), it has to be shown that the region between these borders exists, i.e. that $\gamma_L^* < \gamma_U^*$:

$$\gamma_L^* < \gamma_U^* \quad \Leftrightarrow \quad \frac{n^*-1}{n^*} < \frac{n^*}{n^*+1} \quad \Leftrightarrow \quad (n^*-1)(n^*+1) < (n^*)^2 \quad \Leftrightarrow \quad (n^*)^2 - 1 < (n^*)^2 \ ,$$

which is true. This ends the proof of Theorem 17. □

Theorem 17 defines the region of discount factors which are guaranteed to learn an optimal average reward choice if it has n^* steps. Theorem 18 shows that these regions do not overlap. Therefore, an IGE needs at least N modules, one for each possible number of steps $n = (1, \ldots, N)$, to guarantee that any optimal average reward choice with a maximum of N steps can be learned by an IGE.

Theorem 18. *The γ-regions (Theorem 17) to learn the optimal average reward choice c^* with different number of steps n^* do not overlap.*

Proof. The regions do not overlap, because the upper border γ_U^* for an optimal choice with n^* steps is the same as the lower border γ_L^* for an optimal choice with n^* steps:

$$\gamma_U^*(n^*) = \gamma_L^*(n^*+1) \quad \Leftrightarrow \quad \frac{n^*}{n^*+1} = \frac{(n^*+1)-1}{(n^*+1)} \quad \Leftrightarrow \quad \frac{n^*}{n^*+1} = \frac{n^*}{n^*+1} \ .$$

□

In summary, if the AR-IGE has at least one module in each of the γ-regions defined by $\frac{n-1}{n} < \gamma < \frac{n}{n+1}$ (Theorem 17) for each possible n up to the maximum number of steps in the MDP, then it is guaranteed that at least one of its modules will learn the optimal average reward policy. The most interesting point of these γ-regions is, that they do not depend on the reward values of the choices in an MDP.

5.2.2 Alternative Geometric Proof of Optimality

A more intuitive proof for Theorem 17 can be constructed to show that some γ-modules with $\gamma \in (0, 1)$ are guaranteed to learn the optimal average-reward policy. However, the proof does not allow the definition of the borders of the γ-region that is guaranteed to learn the optimal choice.

The alternative proof makes use of a geometrical connection between the average reward of a choice and its angle in the choice space spanned by the number of steps n and the reward r (Fig. 5.3). The average reward of each choice $c = (r, n)$ can be represented by the tangent function of the angle α_c of the line that goes through the origin of the choice space and the choice. The tangent is given by:

$$\tan \alpha_c = \frac{r}{n+1} \ .$$

The reward r can be interpreted as the opposite side, and the number of steps n as the adjacent side for the angle α_c of a rectangular triangle. The triangle is defined by the origin and the choice in the choice space. Because the tangent function is monotonic increasing, the optimal average reward choice c^* has the largest angle of all choices in an MDP:

$$\forall c: \ \tan \alpha_{c^*} \geq \tan \alpha_c \quad \Leftrightarrow \quad \alpha_{c^*} \geq \alpha_c \ .$$

Therefore, all other choices have a smaller angle and lie below the line defined by the origin and the optimal choice in the choice space (Fig. 5.3):

$$\tan \alpha_{c^*} \geq \tan \alpha_c \quad \Leftrightarrow \quad \tan \alpha_{c^*} \geq \frac{r}{n} \quad \Leftrightarrow \quad r \leq n \cdot \tan \alpha_{c^*} \ . \tag{5.12}$$

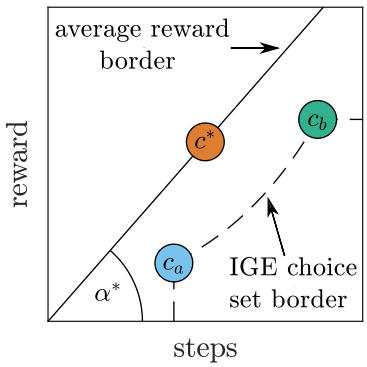


Figure 5.3: Geometric interpretation of the optimality proof.

As a result, the optimal choice c^* is guaranteed to be in the IGE choice set, i.e. it exists some discount factor γ for which it is the optimal solution. A choice is in the IGE choice set, given two other choices ($c_a = (r_a, n_a)$ and $c_b = (r_b, n_b)$) that bracket it, if its reward is above the border defined by Eq. 2.7 (page 40):

$$r > \frac{n \log \frac{r_b}{r_a} + n_b \log r_a - r_a \log r_b}{n_b - n_a} \; .$$

This reward border is a convex function between the two choices c_a and c_b in the choice set space (Fig. 5.3). Because it is convex, and the two choices lie below the border line defined by the optimal choice c^* (Eq. 5.12), the optimal choice c^* lies above the IGE choice set border. This shows that the optimal average-reward choice is learnable by certain γ-modules.

This proof only shows that γ parameters exist for which the optimal choice will be the solution. Based on this proof, to guarantee that the AR-IGE learns the optimal choice, it needs to be assumed that the number of modules is going to infinity and that the γ parameters are evenly spread between 0 and 1. In contrast, Theorem 17 shows also that only a finite number of modules is needed and it defines which γ parameters are needed. Nonetheless, the geometrical proof provides an intuitive explanation why the optimal average reward choice can be learned by an IGE for any deterministic, goal-only-reward MDP.

5.3 The Decoding Average-Reward IGE

The previous section introduced the AR-IGE framework and proved that some γ-modules of the IGE are guaranteed to learn the optimal average-reward policy. This section introduces the Decoding AR-IGE (DAR-IGE), the first of two AR-IGE variants to identify which of the γ-modules learned the best average-reward policy. The DAR-IGE uses the ability of the IGE to decode the expected reward and number of steps of each learned policy to identify the optimal one. The next section introduces the algorithm followed by the proof that the DAR-IGE can correctly identify the optimal average reward policy.

5.3.1 The DAR-IGE Algorithm

The DAR-IGE (Algorithm 5.3) is a IGE variant. It identifies at the beginning of an episode for each of its learned γ-policies the expected future reward and the number of steps to reach the goal state. Based on this information it calculates the expected average reward for each policy. One of the policies is then selected with an ϵ-greedy strategy based on their expected average reward. The policy is used during the whole episode until a terminal state is reached.

The DAR-IGE uses the ability of the IGE to decode the number of steps and the ex-

Algorithm 5.3: Decoding Average-Reward IGE

Input:
Learning rate: $\alpha \in [0, 1]$
Sorted list of discount factors: $\Gamma = (\gamma_1, \ldots, \gamma_m)$ with $\gamma_i \in (0\ 1)$

initialize $Q_\gamma(s, a)$ to zero
repeat (for each episode)
  initialize state s
  `// calculate expected rewards and number of steps for module pairs`
  for i **from** *1* **to** $m - 1$ **do**
    $R_i \leftarrow r(\gamma_i, \gamma_{i+1})$
    $T_i \leftarrow n_s(\gamma_i, \gamma_{i+1})$
  end
  `// remove invalid module pairs and calculate average reward`
  for i **from** *1* **to** m **do**
    if $T_{i-1} = T_i$ *or* $T_i = T_{i+1}$ **then**
      $\bar{R}(i) \leftarrow \frac{R_i}{T_i}$
    else
      $\bar{R}(i) \leftarrow \emptyset$
    end
  end
  `// select the module that maximizes the average reward`
  $\tilde{\gamma} \leftarrow$ choose module based on $\bar{R}(i)$ (e.g. ϵ-greedy)
  repeat (for each step in episode)
    $a \leftarrow$ choose an action for s derived from $Q_{\tilde{\gamma}}(s, a)$ (e.g. ϵ-greedy)
    r, $s' \leftarrow$ take action a, observe outcome
    forall the $\gamma \in \Gamma$ **do**
      $Q_\gamma(s, a) \leftarrow Q_\gamma(s, a) + \alpha\,(r + \gamma \max_{a'} Q_\gamma(s', a') - Q_\gamma(s, a))$
    end
    $s \leftarrow s'$
  until s *is terminal-state*
until *termination*

pected final reward of its γ-policies in deterministic, goal-only-reward MDPs (Theorem 8, page 44). Given two modules (γ_a, γ_b) with the same expected trajectory from state s_0, the number of steps to reach the goal state at the end of the trajectory is:

$$n_{s_0}(\gamma_a, \gamma_b) = \frac{\log V_{\gamma_a}(s_0) - \log V_{\gamma_b}(s_0)}{\log \gamma_a - \log \gamma_b} + 1 \ . \tag{5.13}$$

The expected reward for reaching the goal state can be decoded by:

$$r_{s_0}(\gamma_a, \gamma_b) = \frac{V_{\gamma_a}(s_0)}{\gamma_a^{n_{s_0}(\gamma_a,\gamma_b)-1}} \ .$$

Based on this information the average reward of the module pair is given by:

$$\bar{r}_{s_0}(\gamma_a, \gamma_b) = \frac{r_{s_0}(\gamma_a, \gamma_b)}{n_{s_0}(\gamma_a, \gamma_b)} \ . \tag{5.14}$$

The DAR-IGE computes the average reward for all neighbors in its set of modules Γ. Pairs which have different trajectories and therefore a wrongly computed average reward are detected by comparing the number of steps of neighboring pairs as discussed in Section 2.3. A module pair (γ_i, γ_{i+1}) is ignored if:

$$n_{s_0}(\gamma_{i-1}, \gamma_i) \quad \neq \quad n_{s_0}(\gamma_i, \gamma_{i+1}) \quad \neq \quad n_{s_0}(\gamma_{i+1}, \gamma_{i+2}) \ . \tag{5.15}$$

The active module $\tilde{\gamma}$ for an episode is selected with an ϵ-greedy approach controlled by the exploration rate ϵ_γ. With probability $1 - \epsilon_\gamma$ one of the modules with the policy that is expected to result in the highest average reward is selected:

$$\tilde{\gamma} = \underset{\gamma_i}{\operatorname{argmax}}\, \bar{r}_{s_0}(\gamma_i, \gamma_{i+1}) \ .$$

With probability ϵ_γ, a random policy is selected. The random selection is based on the different greedy policies that the agent can use starting from state s_0. The policies are identified based on the decoded number of steps $n_{s_0}(\gamma_a, \gamma_b)$ of the module pairs (Eq. 5.13). Each possible choice that an IGE can learn results in a different number of steps in a deterministic MDP (Definition 4, page 37). One of these policies is selected and one of the modules that encodes this policy is used as active module $\tilde{\gamma}$.

5.3.2 Proof of Optimality

This section provides the proof that the DAR-IGE is guaranteed to converge to the optimal average-reward policy per episode and discusses the number of modules that are needed. The proof is build on Theorem 17 that showed that some γ-modules converge to the optimal policy. The missing part to prove optimality is to show that the optimal policy can be identified within the set of all learned γ-policies . The proof assumes that no exploration over the modules is performed, i.e. $\epsilon_\gamma = 0$.

The goal of the DAR-IGE is to optimize the expected average reward per episode:

$$\mathrm{E}_{\pi,s_0}\left[\frac{1}{T}\sum_{t=0}^{T-1} r_t\right]$$

The DAR-IGE identifies the expected average reward for each γ-modules policy by decoding their policies expected reward r and the number of steps $n+1$. The decoded information is then used to calculate the average reward by Eq. 5.14. It can be shown that the average reward is correctly computed. First, Theorem 8 (page 44) proved that the decoded reward and number of steps are correct, i.e. that $n_{s_0}(\gamma_a, \gamma_b) = T_{\pi_{\gamma_{a,b}}, s_0}$ and $r_{s_0}(\gamma_a, \gamma_b) = \mathrm{E}_{\pi_{\gamma_{a,b}}, s_0}[r_{T-1}]$. In MDPs with a determinsitic transition function, the state transitions of trajectories of greedy policies are deterministic. Therefore, the number of steps for a policy T_{π,s_0} to reach a goal state is the same for each episode. Moreover, in goal-only-reward MDPs, only the final reward is non-zero. Thus, the expected average reward becomes:

$$\mathrm{E}_{\pi,s_0}\left[\frac{1}{T_{\pi,s_0}}\sum_{t=0}^{T_{\pi,s_0}-1} r_t\right] = \frac{1}{T_{\pi,s_0}}\sum_{t=0}^{T_{\pi,s_0}-1} \mathrm{E}_{\pi,s_0}[r_t] = \frac{\mathrm{E}_{\pi,s_0}[r_{T-1}]}{T_{\pi,s_0}} ,$$

which is the average reward that the DAR-IGE calculates for each module (Eq. 5.14) and proves that it can correctly select the optimal one.

The number of modules needed to guarantee that the optimal policy can be identified is $3 \cdot N$, where N is the maximum number of steps of the optimal trajectory in an MDP. This results from the fact that modules in N distinct γ-regions are needed (Theorem 17). Furthermore, the DAR-IGE needs to have at least three modules to correctly identify that the modules encode the same policy (Eq. 5.15).

5.4 The Value-Quotient Average-Reward IGE

This section introduces a different variant of the IGE to optimize average reward, the Value-Quotient Average-Reward Independent γ-Ensemble (VQAR-IGE). In contrast to the DAR-IGE, VQAR-IGE does not decode the average reward for each of its γ-policies. Instead, it compares the values of all modules directly with each other to identify the optimal policy. As a result, it only needs N modules to guarantee optimality, whereas the DAR-IGE needs $3N$ modules, where N is the maximum number of steps of the optimal trajectory in an MDP. However, the VQAR-IGE cannot decode extra information for its γ-policy, which can, for example, help to distinguish between the learned policies that can be helpful for exploration. First, the VQ-AR-IGE algorithm is introduced, followed by a discussion of its optimality.

5.4.1 The VQAR-IGE Algorithm

The VQAR-IGE (Algorithm 5.4) selects its active module $\tilde{\gamma}$ at the beginning of an episode and uses the module's values to define its policy. It's goal is to select the γ-module that maximizes the expected average reward per episode.

The selection of the active module is done with an ϵ-greedy approach controlled by the exploration parameter ϵ_γ. With a probability of $1-\epsilon_\gamma$, the module is chosen that tries to maximizes the average reward, otherwise a random module is selected for exploration. The identification of the γ-module that maximizes the average reward is directly based on a comparison of the values between all modules for start state s_0. The values are

Algorithm 5.4: Value-Quotient Average-Reward IGE

Input:
Learning rate: $\alpha \in [0, 1]$
Sorted list of discount factors: $\Gamma = (\gamma_1, \ldots, \gamma_m)$ with $\gamma_i \in (0\ 1)$

initialize $Q_\gamma(s, a)$ arbitrarily
repeat (for each episode)
  initialize s
  `// Selection of the active` γ`-module` γ^*
  initialize $nBiggerFactor$ to 0
  for i **from** *1* **to** m **do**
    for j **from** *1* **to** m **do**
      if $\frac{V_{\gamma_i}(s)}{V_{\gamma_j}(s)} \geq F_V(\gamma_i, \gamma_j)$ **then**
        $nBiggerFactor(i) \leftarrow nBiggerFactor(i) + 1$
      end
    end
  end
  `// select the module that maximizes the average reward`
  $\tilde{\gamma} \leftarrow$ choose module based on $nBiggerFactor$ (e.g. ϵ-greedy)
  repeat (for each step in episode)
    choose an action a for s derived from $Q_{\tilde{\gamma}}(s, a)$ (e.g. ϵ-greedy)
    take action a, observe r, s'
    forall the $\gamma \in \Gamma$ **do**
      $Q_\gamma(s, a) \leftarrow Q_\gamma(s, a) + \alpha \left(r + \gamma \max_{a'} Q_\gamma(s', a') - Q_\gamma(s, a)\right)$
    end
    $s \leftarrow s'$
  until s *is terminal-state*
until *termination*

compared by the quotient between each other ($\frac{V_\gamma(s_0)}{V_{\gamma_i}(s_0)}$). It can be shown that the quotient between the module that learned the optimal average reward policy and all other modules has to be greater than the following boundary:

$$F_V(\gamma_a, \gamma_b) = \frac{\gamma_a^{n(\gamma_a)-1} n(\gamma_a)}{\max\left(\gamma_b^{\lfloor n_\mathbb{R} \rfloor - 1} \lfloor n_\mathbb{R} \rfloor, \gamma_b^{\lceil n_\mathbb{R} \rceil - 1} \lceil n_\mathbb{R} \rceil\right)} \quad \text{with} \quad n_\mathbb{R} = -\frac{1}{\log \gamma_b}\ , \tag{5.16}$$

where $n(\gamma_a)$ is the number of steps n for which a module with discount factor γ_a is guaranteed to learn the optimal average reward choice with this number of steps ($\frac{n-1}{n} < \gamma_a < \frac{n}{n+1}$) as defined by Theorem 17. Only the quotient between the optimal module and all other modules is guaranteed to be bigger than this boundary. The module that fulfills

this condition is used as active module:

$$\tilde{\gamma} = \gamma \;\; s.t.\; \forall \gamma_i \neq \gamma : \;\; \frac{V_\gamma(s_0)}{V_{\gamma_i}(s_0)} \geq F_V(\gamma, \gamma_i) \; . \tag{5.17}$$

5.4.2 Proof of Optimality

The VQAR-IGE is guaranteed to converge to the optimal average reward policy per episode. The proof is built on Theorem 17 which shows that one of the γ-modules learns the optimal average-reward policy per episode. The following section shows that the selection condition of the VQAR-IGE (Eq. 5.17), based on boundary F_V, can identify the optimal module. It also discusses the requirement for the number of modules and their discount factor settings. The proof assumes that no exploration over the modules is performed, i.e. $\epsilon_\gamma = 0$.

The proof that Eq. 5.17 selects the optimal module is composed of two theorems. Theorem 19 shows that the quotient between the values of the module γ_*, that is guaranteed to learn the optimal choice (Theorem 17), and modules γ that are not is larger or equal than the boundary $F_V(\gamma_*, \gamma)$. This is the necessary condition that the quotient of the optimal module, compared to all other modules, has to fulfill. Theorem 20 shows that this condition is also sufficient, because the quotient of any module γ, that is not guaranteed to learn the optimal choice and the optimal module γ_*, is smaller than boundary $F_V(\gamma_*, \gamma)$. Both theorems make use of the concept of choices (Definition 4, page 37).

Theorem 19. *The quotient between the values of the module γ^* that is guaranteed to encode the optimal average choice c^* and the values of any module γ that is not according to Theorem 17 has the lower bound F_V:*

$$\forall c^*, c \in C, \;\; \gamma_* \in \Gamma^*, \gamma \notin \Gamma^* : \;\; \frac{V_{\gamma_*}(c^*)}{V_\gamma(c)} \geq F_V(\gamma_*, \gamma) \; , \tag{5.18}$$

where c is any possible choice, and $F_V(\gamma_*, \gamma)$ is defined by Eq. 5.16.

Proof. The inequality Eq. 5.18 is proven by showing that the minimum of the value quotient is equal to the boundary $F_V(\gamma_*, \gamma)$. First, the minimum is reformulated using the definition of values for choices (Theorem 2, page 37):

$$\min_{c,c^*} \frac{V_{\gamma_*}(c^*)}{V_\gamma(c)} = \min_{r^*,r,n} \frac{\gamma_*^{n(\gamma_*)-1} r^*}{\gamma^{n-1} r} \qquad \text{s.t.} \;\; \frac{r^*}{n(\gamma_*)} \geq \frac{r}{n} \; , \tag{5.19}$$

where $n(\gamma_*)$ is the number of steps of the optimal choice which γ_* is guaranteed to learn. The minimum can be solved by substitution, concentrating at first on the reward r. Eq. 5.19 is minimized by maximizing the reward:

$$\begin{aligned} r^{min} &= \text{argmin}_r \, \tfrac{\gamma_*^{n(\gamma_*)-1} r^*}{\gamma^{n-1} r} \\ &= \text{argmax}_r \, r \qquad\qquad \text{s.t.} \;\; \tfrac{r^*}{n(\gamma_*)} \geq \tfrac{r}{n} \end{aligned}$$

Given the average reward constraint ($\frac{r^*}{n(\gamma_*)} \geq \frac{r}{n} \;\Leftrightarrow\; r \leq n \frac{r^*}{n(\gamma_*)}$) the maximum of r is:

$$\begin{aligned} r^{min} &= \text{argmax}_r \, r \qquad\qquad \text{s.t.} \;\; \tfrac{r^*}{n(\gamma_*)} \geq \tfrac{r}{n} \\ &= n \tfrac{r^*}{n(\gamma_*)} \; . \end{aligned}$$

Using this result for the minimization of the value quotient (Eq. 5.19) eliminates the reward variables in the formulation:

$$\min_{r^*,r,n} \frac{\gamma_*^{n(\gamma_*)-1} r^*}{\gamma^{n-1} r} = \min_{r^*,n} \frac{\gamma_*^{n(\gamma_*)-1} r^*}{\gamma^{n-1} n \frac{r^*}{n(\gamma_*)}} = \min_{n} \frac{\gamma_*^{n(\gamma_*)-1} n(\gamma_*)}{\gamma^{n-1} n} = \frac{\gamma_*^{n(\gamma_*)-1} n(\gamma_*)}{\max_n \gamma^{n-1} n} . \quad (5.20)$$

The result for the maximum of Eq. 5.20's denominator is given by Lemma 19.1.

Lemma 19.1. *For $\gamma \in (0,1)$, $n \in \mathbb{N}$ it holds that*

$$\max_n \gamma^{n-1} n = \max\left(\gamma^{\lfloor n_\mathbb{R} \rfloor - 1} \lfloor n_\mathbb{R} \rfloor, \gamma^{\lceil n_\mathbb{R} \rceil - 1} \lceil n_\mathbb{R} \rceil\right) \text{ , with } n_\mathbb{R} = -\frac{1}{\log \gamma} . \quad (5.21)$$

Proof. Assuming that the number of steps is real valued ($n_\mathbb{R} \in \mathbb{R}$), the maximum of $f(n_\mathbb{R}) = \gamma^{n_\mathbb{R}-1} n_\mathbb{R}$ can be shown to be $n = -\frac{1}{\log \gamma}$ using the derivative test approach. Setting the first derivative of $f(n_\mathbb{R})$ to 0 results in:

$$\frac{\partial}{\partial n_\mathbb{R}} f(n_\mathbb{R}) \quad \Leftrightarrow \quad \frac{\partial}{\partial n_\mathbb{R}} \gamma^{n_\mathbb{R}-1} n_\mathbb{R} = \gamma^{n_\mathbb{R}-1}(n_\mathbb{R} \log \gamma + 1) = 0 \quad \Leftrightarrow \quad n_\mathbb{R} = -\frac{1}{\log \gamma} .$$

The second derivative $\frac{\partial^2 f}{\partial^2 n_\mathbb{R}} = \gamma^{n_\mathbb{R}-1} \log \gamma (n_\mathbb{R} \log \gamma + 2)$ at $n_\mathbb{R} = -\frac{1}{\log \gamma}$ is negative, showing that the extremum is a maximum:

$$\frac{\partial^2 f}{\partial^2 n_\mathbb{R}} \left(-\frac{1}{\log \gamma}\right) = \gamma^{-\frac{1}{\log \gamma}-1} \log \gamma \left(-\frac{1}{\log \gamma} \log \gamma + 2\right) = \gamma^{-\frac{1}{\log \gamma}-1} \log \gamma .$$

The second derivative is negative because $\log \gamma$ is negative and $\gamma^{-\frac{1}{\log \gamma}-1}$ is positive for $\gamma \in (0,1)$. Therefore, the maximum point for $\gamma^{n_\mathbb{R}-1} n_\mathbb{R}$ is at $n_\mathbb{R} = -\frac{1}{\log \gamma}$ for $n_\mathbb{R} \in \mathbb{R}$. However, this ignores that the steps in Eq. 5.21 are natural numbers ($n \in \mathbb{N}$). Thus, either the smaller $\lfloor n_\mathbb{R} \rfloor$ or the larger $\lceil n_\mathbb{R} \rceil$ natural number of $n_\mathbb{R}$ is the maximum point, finishing the proof of Lemma 19.1. □

Using Lemma 19.1 for the minimization of the value quotient in Eq. 5.20 results in the lower bound $F_V(\gamma_*, \gamma)$ defined by Eq. 5.16 and finishes the proof of Theorem 19. □

Theorem 19 shows that the optimal module will fulfill the necessary condition that its quotient with all other modules is larger than boundary F_V. The next theorem shows that this is also a sufficient condition, because no other module can fulfill it.

Theorem 20. *The quotient between the values of any module γ, that is not guaranteed to encode the optimal average choice c^*, and the value of the module γ_* that is according to Theorem 17 is smaller than the bound F_V:*

$$\forall c, c^* \in C,\ \gamma \notin \Gamma^*, \gamma_* \in \Gamma^* : \quad \frac{V_\gamma(c)}{V_{\gamma_*}(c^*)} \leq F_V(\gamma, \gamma_*) , \quad (5.22)$$

where c is any possible choice, and $F_V(\gamma_*, \gamma)$ is defined by Eq. 5.16.

Proof. The inequality Eq. 5.22 is proven by showing that the maximum of the value quotient is equal to the boundary F_V. First, the maximum is reformulated using the definition of values for choices (Theorem 2, page 37):

$$\max_{c,c^*} \frac{V_\gamma(c)}{V_{\gamma_*}(c^*)} = \max_{n,r,r^*} \frac{\gamma^{n-1} r}{\gamma_*^{n(\gamma_*)-1} r^*} \qquad \text{s.t. } \frac{r^*}{n(\gamma_*)} \geq \frac{r}{n} \ ,$$

where $n(\gamma_*)$ is the number of steps of the optimal choice for which γ_* is guaranteed to learn it. Similar to the proof of Theorem 19, the reward variables can be eliminated from the maximum by using the average reward condition ($\frac{r^*}{n(\gamma_*)} \geq \frac{r}{n} \ \Leftrightarrow \ r \leq n \frac{r^*}{n(\gamma_*)}$):

$$\max_{n,r,r^*} \frac{\gamma^{n-1} r}{\gamma_*^{n(\gamma_*)-1} r^*} \ = \ \max_{n,r^*} \frac{\gamma^{n-1} n \frac{r^*}{n(\gamma_*)}}{\gamma_*^{n(\gamma_*)-1} r^*} \ = \ \frac{\max_n \gamma^{n-1} n}{\gamma_*^{n(\gamma_*)-1} n(\gamma_*)} \ .$$

Lemma 19.1 can be used to reformulate the maximum:

$$\frac{\max_n \gamma_a^{n-1} n}{\gamma_b^{n(\gamma_b)-1} n(\gamma_b)} \ = \ \frac{\max\left(\gamma_a^{\lfloor n_\mathbb{R} \rfloor - 1} \lfloor n_\mathbb{R} \rfloor, \gamma_a^{\lceil n_\mathbb{R} \rceil - 1} \lceil n_\mathbb{R} \rceil\right)}{\gamma_b^{n(\gamma_b)-1} n(\gamma_b)} \ . \tag{5.23}$$

To show that the maximum of the value quotient (Eq. 5.23) is equal to the boundary F_V, Lemma 20.1 is used to identify the number of steps $n(\gamma)$ of the optimal choice for which γ is guaranteed to learn it.

Lemma 20.1. *The number of steps $n(\gamma) \in \mathbb{N}$ of the optimal choice $c^* = (r^*, n(\gamma))$ for which a module with the discount factor γ is guaranteed to learn it is given by:*

$$n(\gamma) = \begin{cases} \lfloor n_\mathbb{R} \rfloor & \mid \gamma^{\lfloor n_\mathbb{R} \rfloor - 1} \lfloor n_\mathbb{R} \rfloor > \gamma^{\lceil n_\mathbb{R} \rceil - 1} \lceil n_\mathbb{R} \rceil \\ \lceil n_\mathbb{R} \rceil & \mid \textit{otherwise} \end{cases} \ , \textit{ with } n_\mathbb{R} = -\frac{1}{\log \gamma} \ .$$

Proof. The choice c^* is guaranteed to be the outcome of module γ if its value is larger than any other choice:

$$\gamma^{n(\gamma)-1} r^* > \max_{r,n} \gamma^{n-1} r \quad \text{s.t. } \frac{r^*}{n(\gamma)} \geq \frac{r}{n}, \ n(\gamma) \neq n \ , \tag{5.24}$$

where the conditions are that c^* is the optimal average reward choice, and that in deterministic, goal-only-reward MDPs only one choice with a specific number of steps exists (Definition 4, page 37). Using the average reward condition ($\frac{r^*}{n(\gamma)} \geq \frac{r}{n} \ \Leftrightarrow \ r \leq n \frac{r^*}{n(\gamma)}$), the reward variables of Eq. 5.24 can be eliminated:

$$\begin{aligned} \gamma^{n(\gamma)-1} r^* > \max\nolimits_{r,n} \gamma^{n-1} r \quad &\Leftrightarrow \quad \gamma^{n(\gamma)-1} r^* > \max\nolimits_n \gamma^{n-1} n \tfrac{r^*}{n(\gamma)} \\ &\Leftrightarrow \quad \gamma^{n(\gamma)-1} n(\gamma) > \max\nolimits_n \gamma^{n-1} n \qquad \text{s.t. } n(\gamma) \neq n \ . \end{aligned} \tag{5.25}$$

Because the number of steps between the optimal choice and any other choice are different ($n(\gamma) \neq n$), Eq. 5.25 can only be fulfilled if $\gamma^{n(\gamma)-1} n(\gamma)$ is the maximum. Lemma 19.1 provides the solution of the maximum:

$$\max_{n(\gamma)} \gamma^{n(\gamma)-1} n(\gamma) \ = \ \max\left(\gamma^{\lfloor n_\mathbb{R} \rfloor - 1} \lfloor n_\mathbb{R} \rfloor, \gamma^{\lceil n_\mathbb{R} \rceil - 1} \lceil n_\mathbb{R} \rceil\right) \ \text{, with } \ n_\mathbb{R} = -\frac{1}{\log \gamma} \ .$$

As a result, $n(\gamma)$ is either $\lfloor n_\mathbb{R} \rfloor$ or $\lceil n_\mathbb{R} \rceil$ depending on if $\gamma^{\lfloor n_\mathbb{R} \rfloor - 1} \lfloor n_\mathbb{R} \rfloor$ is larger than $\gamma^{\lceil n_\mathbb{R} \rceil - 1} \lceil n_\mathbb{R} \rceil$ or not as stated by Lemma 20.1. □

Using Lemma 20.1 on Eq. 5.25, the maximum value quotient becomes:

$$\max_{c,c^*} \frac{V_{\gamma_a}(c)}{V_{\gamma_b}(c^*)} = \frac{\max\left(\gamma_a^{\lfloor n_{\mathbb{R}} \rfloor - 1} \lfloor n_{\mathbb{R}} \rfloor, \gamma_a^{\lceil n_{\mathbb{R}} \rceil - 1} \lceil n_{\mathbb{R}} \rceil\right)}{\max\left(\gamma_b^{\lfloor n_{\mathbb{R}} \rfloor - 1} \lfloor n_{\mathbb{R}} \rfloor, \gamma_b^{\lceil n_{\mathbb{R}} \rceil - 1} \lceil n_{\mathbb{R}} \rceil\right)} . \tag{5.26}$$

Using Lemma 20.1 on the boundary F_V results in:

$$F_V(\gamma_a, \gamma_b) = \frac{\max\left(\gamma_a^{\lfloor n_{\mathbb{R}} \rfloor - 1} \lfloor n_{\mathbb{R}} \rfloor, \gamma_a^{\lceil n_{\mathbb{R}} \rceil - 1} \lceil n_{\mathbb{R}} \rceil\right)}{\max\left(\gamma_b^{\lfloor n_{\mathbb{R}} \rfloor - 1} \lfloor n_{\mathbb{R}} \rfloor, \gamma_b^{\lceil n_{\mathbb{R}} \rceil - 1} \lceil n_{\mathbb{R}} \rceil\right)} , \tag{5.27}$$

showing that both (Eq. 5.26 and Eq. 5.27) are equal and that the inequality of Eq. 5.22 is correct, finishing the proof of Theorem 20. □

Theorems 19 and 20 show that the selection condition of the VQAR-IGE (Eq. 5.17) can guarantee to find the optimal module.

Both theorems compare the value quotient of modules that are guaranteed to learn the optimal choice and modules that do not. They do not consider the value quotient of two modules (γ_a, γ_b) that are guaranteed to learn the optimal choice with the same number of steps n ($\frac{n-1}{n} < \gamma_a < \gamma_b < \frac{n}{n+1}$). Therefore, to guarantee optimality, only one module per possible number of steps has to be used. As a result, in MDPs with optimal choices that have up to N steps, the VQAR-IGE needs N modules to guaranteed that one module converges to the optimal policy, and to identify it.

5.5 Experimental Evaluation

Both variants of the AR-IGE have been compared to the continuous and episodic variants of the average-adjusted algorithms (Section 5.1). The episodic variants are indicated by (e) and the continuous by (c) after their name. Two task domains were used for the comparison: decision-tree and grid-world tasks. For each domain, 100 tasks have been randomly generated for the evaluation. In decision-tree tasks, the AR-IGE had a slightly better performance than the average-adjusted algorithms. The AR-IGE's performance advantage was stronger in the more complex grid-world tasks. The optimal variant of the average-adjusted algorithms depended on the task domain. In decision-tree tasks, the continuous variants demonstrated a better performance. Whereas, in grid-world tasks, the episodic variants were better. The next section introduces both domains and their task designs, followed by a description of the used learning parameters. The final section discusses the experimental results.

5.5.1 Task Design

For each problem domain 100 tasks were randomly generated. First, the decision-tree domain and its generation of random tasks are described, followed by the description of the grid-world domain. Each task had three experimental conditions that are described in the final part of this section.

Decision-tree tasks have the form of directed, rooted tree graphs (Fig. 5.4, a). The agent starts in the root of the graph. Several paths with different lengths lead to goal states. Paths branch from each other at certain states where the agent has to choose between them. One hundred decision-tree tasks were randomly generated. The tasks were generated by first sampling the number of goal states $N^G \in \mathbb{N}$ from a uniform distribution between 3 and 6 ($N^G \sim \mathcal{U}(3, 6)$). Then, the number of steps $n_g \in \mathbb{N}$ to reach a goal state and its reward $r_g \in R$ were uniformly sampled between 1 and 25 ($n_g \sim \mathcal{U}(1, 25)$, $r_g \sim \mathcal{U}(1, 25)$). Afterward, the paths to the different goal states were iteratively created. At first, the states on the path to the goal with the smallest number of steps were created. Next, a loop over the other goals, sorted by their number of steps from smallest to largest, was used to create the other states. For the current goal, one of the previously created states was randomly chosen as a decision point for the path to the current goal. The missing states on its path from the decision point were created so that it has the correct number of steps n_g from the start state.

The grid-world tasks have a 9x9 state space (Fig. 5.4, b). The agent starts in a start state and has four actions to move either north, east, south or west. It can reach one of several goal states where it receives positive reward. Also for the grid-world domain, 100

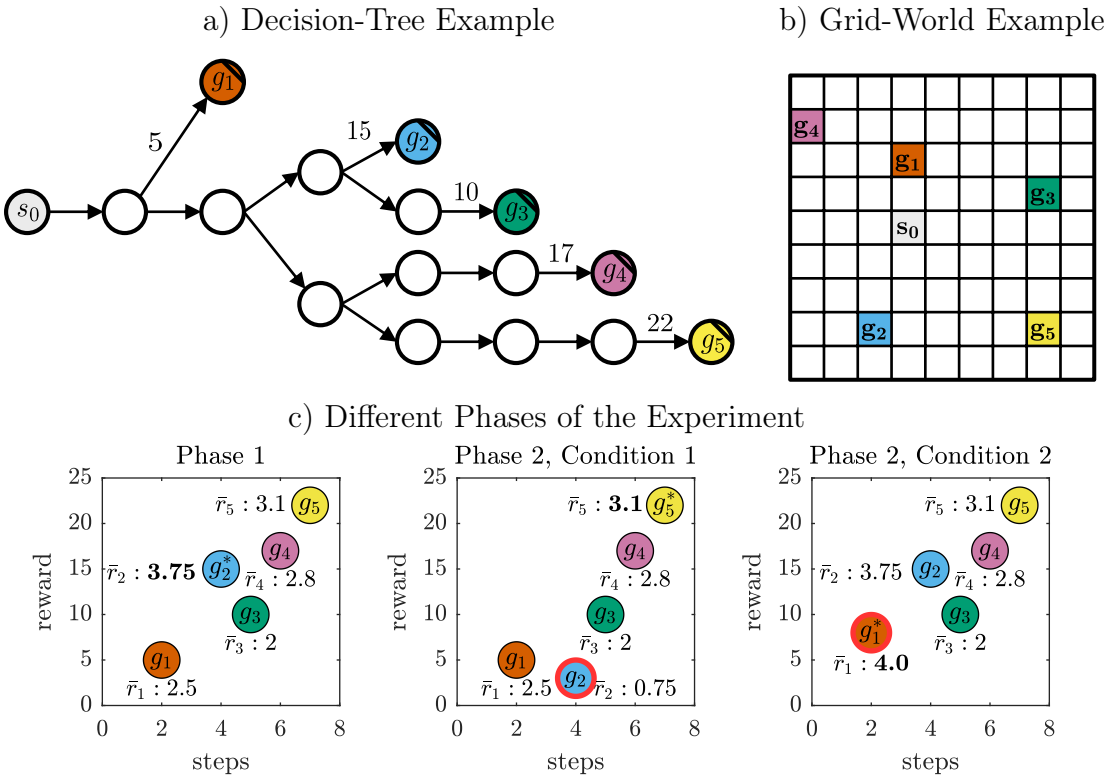


Figure 5.4: Example of a randomly generated task for the decision-tree (a), and the grid-world domain (b). Each task has two phases (c). In Phase 1, the agent learns the randomly generated task. In Phase 2, the rewards of the task are modified so that the optimal goal state changes. In Condition 1, the reward of the optimal goal from Phase 1 (g_2) is reduced, to make a different goal (g_5) optimal. In Condition 2, the reward of a previous non-optimal goal (g_1) is changed to make it optimal.

different tasks have been randomly generated. Each task has one randomly placed start state and a set of 3 to 6 randomly placed goal states ($N^G \sim \mathcal{U}(3,6)$). The rewards $r_g \in \mathbb{R}$ of every goal state were uniformly sampled between 0 and 25 ($r_g \sim \mathcal{U}(0,25)$).

Each algorithm was executed for 100 runs per task to measure their average performance. Each run was divided in two phase (Fig. 5.4, c). Phase 1 evaluated how the algorithms learn a new task. For decision-tree tasks the algorithms had 1000 episodes to learn, and for grid-world tasks they had 3000 episodes. For grid-world tasks more episodes were given, because they are more complex than decision-tree tasks. Phase 2 evaluated how the algorithms adjust to changes in the task. This was tested under two conditions. In the first condition, the reward for the optimal average-reward goal was reduced making it non-optimal. Whereas, in the second condition the reward for a non-optimal goal was increased to make it optimal. Phase 2 had a length of 500 episodes for decision-tree tasks and 1500 episodes for grid-world tasks. In both conditions of Phase 2, the agents started with the Q-values and average-reward estimates learned during Phase 1.

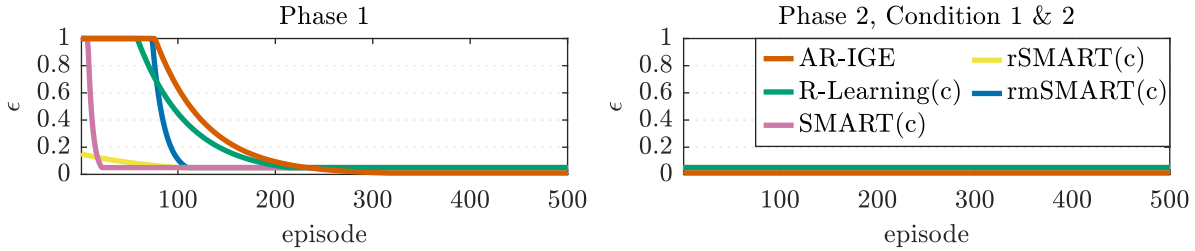
5.5.2 Learning Parameters

The learning parameters for each algorithm were selected to achieve a high average reward over all phases. For each task domain, a different set of learning parameters was identified (Figs. 5.5 and 5.6). Only the parameters for the continuous average-adjusted variants in the decision-tree, and the episodic ones for the grid-world tasks are discussed. The parameters of the other variants can be found in Appendix B.

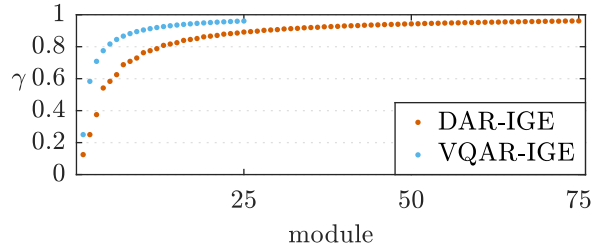
The learning rate for the Q-values of all algorithms and for each domain was set to $\alpha_Q = 1$. This was possible because all tasks were deterministic. The best learning rate

Learning Parameters for Decision-Tree Tasks

a) Exploration rates



b) AR-IGE Γ-sets



c) Remaining parameters

Algorithm	Parameters
All	$\alpha_Q = 1$
AR-IGE	$\epsilon_\gamma = 0.1$
R-Learning(c)	$\alpha_\rho = 0.005$
rSMART(c)	$\alpha_\rho = 1$
rmSMART(c)	$\alpha_\rho = 0.01$

Figure 5.5: The learning parameters for decision-tree tasks for the AR-IGE variants, and the continuous variants of the average-adjusted algorithms.

α_ρ for the expected average reward ρ of R-Learning, rSMART, and rmSMART differed between the algorithms (Figs. 5.5 and 5.6, c). For both domains, the best learning rate for the rSMART algorithm was $\alpha_\rho = 1$. As a result, its update rule is synonymous with the update rule of the SMART algorithm, and the performance for both algorithms was very similar.

All algorithms used an ϵ-greedy action selection. The exploration rate depended on the number of episodes k and was bounded between ϵ_{min} and 1:

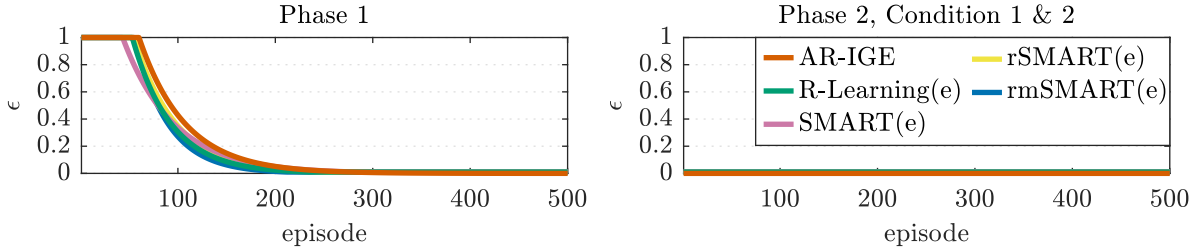
$$\epsilon(k) = \min\left(\epsilon_{min}, \max\left(1, \epsilon_{init} \cdot d^k\right)\right) ,$$

where $d \in [0, 1]$ is the decay factor, and $\epsilon_{init} \in \mathbb{R}^+$ is the initial ϵ. A ϵ_{init} larger than 1 allows the exploration rate to stay at 1 for several episodes at the beginning of learning. For most algorithms, a strong exploration in the beginning that decayed within 200 to 300 episodes to its minimum exploration rate was best for both domains (Figs. 5.5 and 5.6, a). Only, the SMART and rSMART algorithms reached their optimal performance with less exploration for the decision-tree domain.

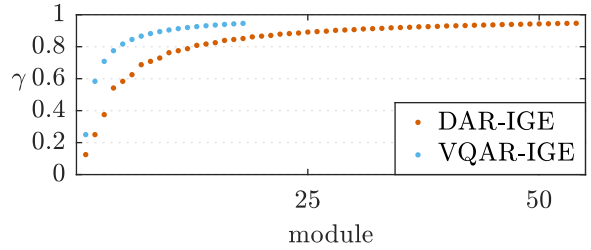
The γ-modules were selected to allow the AR-IGE to learn all possible optimal solutions based on Theorem 17. The decision-tree tasks had a maximum path length of 25 steps, resulting in a set of 75 modules for the DAR-IGE and 25 for the VQAR-IGE (Fig. 5.5, b). For the grid-world tasks, only 54 modules for the DAR-IGE and 18 modules for the VQAR-IGE were necessary (Fig. 5.6, b). The exploration rate of the AR-IGEs ϵ-greedy module selection was set to $\epsilon_\gamma = 0.1$ for decision trees, and $\epsilon_\gamma = 0.025$ for grid worlds.

Learning Parameters for Grid-World Tasks

a) Exploration rates



b) AR-IGE Γ-sets



c) Remaining parameters

Algorithm	Parameters
All	$\alpha_Q = 1$
AR-IGE	$\epsilon_\gamma = 0.025$
R-Learning(e)	$\alpha_\rho = 0.01$
rSMART(e)	$\alpha_\rho = 1$
rmSMART(e)	$\alpha_\rho = 0.005$

Figure 5.6: The learning parameters for grid-world tasks for the AR-IGE variants, and the episodic variants of the average-adjusted algorithms.

5.5.3 Experimental Results

The algorithms were compared based on their achieved average reward over the 100 randomly generated tasks per domain. Three measurements are used to compare the algorithms. The first is the total average reward gained over all steps of the experiment. It was averaged over the 100 tasks per domain, and calculated for each individual phase and condition. The second measurement evaluates the difference between the algorithms per task. The third compares the learning curve of the algorithms over all environments in terms of the average reward per episode. The next sections introduce each performance measure in detail and discusses their results for both task domains. Results for the episodic variants of the average-adjusted algorithms were partly presented in (Reinke, Uchibe, & Doya 2017).

Average Reward over all Steps

The main criteria to evaluate the algorithms and to select their learning parameters was their average reward per step gained over the steps of all phases. To make the performance comparable between tasks and phases, the average reward has been normalized to lie between 0 and 1, with 1 being the optimal average reward. The normalization uses the optimal average reward $\bar{r}^*_{m,p}$ for task m and phase p to adjust the reward of each step with:

$$\hat{r}_{m,p}(t) = \frac{r_{m,p}(t)}{\bar{r}^*_{m,p}} \ , \tag{5.28}$$

Algorithm	Decision-Tree Tasks		Grid-World Tasks	
	episodic	continuous	episodic	continuous
R-Learning	0.819	**0.845**	**0.784**	0.779
SMART	0.774	**0.832**	**0.772**	0.684
rSMART	0.823	**0.829**	**0.772**	0.683
rmSMART	0.818	**0.847**	**0.773**	0.709
DAR-IGE	**<u>0.849</u>**		**<u>0.800</u>**	
VQAR-IGE	**0.826**		**0.775**	

Table 5.1: Results for the normalized average reward over the steps of all phases. The results of the episodic and continuous variant of the average-adjusted algorithms are shown. The DAR-IGE had in both task domains the best performance. In decision-tree tasks, it was only slightly better than R-Learning(c), and rmSMART(c). In the more complex domain of grid-worlds, it had a stronger advantage over other algorithms. The continuous variant of the average-adjusted algorithms performed better in the decision-tree domain. Whereas, the episodic variant performed better in grid-world tasks. The VQAR-IGE had a reduced performance compared to the DAR-IGE in both tasks domains.

where $r_{m,p}(t)$ is the reward of step t. Based on the adjusted reward, the normalized average reward $\bar{r}_{m,p}$ per phase p and over all phases ($\bar{r}_m$) for task m are calculated by:

$$\bar{r}_{m,p} = \frac{1}{T_{m,p}} \sum_{t=0}^{T_{m,p}-1} \hat{r}_{m,p}(t) \ , \quad \bar{r}_m = \frac{\sum_{t=0}^{T_{m,1}-1} \hat{r}_{m,1}(t) + \sum_{t=0}^{T_{m,2}-1} \hat{r}_{m,2}(t) + \sum_{t=0}^{T_{m,3}-1} \hat{r}_{m,3}(t)}{T_{m,1} + T_{m,2} + T_{m,3}} \ , \tag{5.29}$$

where $T_{m,p}$ is the number of steps in phase p. The two conditions of Phase 2 are indexed by $p = 2$ and $p = 3$. Finally, the mean over the 100 tasks is taken and used as the basis to compare the algorithms:

$$\bar{r}_p = \frac{1}{100} \sum_{m=1}^{100} \mathrm{E}_{\mathrm{runs}} \left[\bar{r}_{m,p} \right] \ , \quad \bar{r} = \frac{1}{100} \sum_{m=1}^{100} \mathrm{E}_{\mathrm{runs}} \left[\bar{r}_m \right] \ ,$$

where $\mathrm{E}_{\mathrm{runs}} \left[\bar{r}_{m,p} \right]$ and $\mathrm{E}_{\mathrm{runs}} \left[\bar{r}_m \right]$ are the means over the 100 runs per task.

The results for the continuous and episodic variants of the average-adjusted algorithms showed a consistent trend for all algorithms (Table 5.1). In the decision-tree domain,

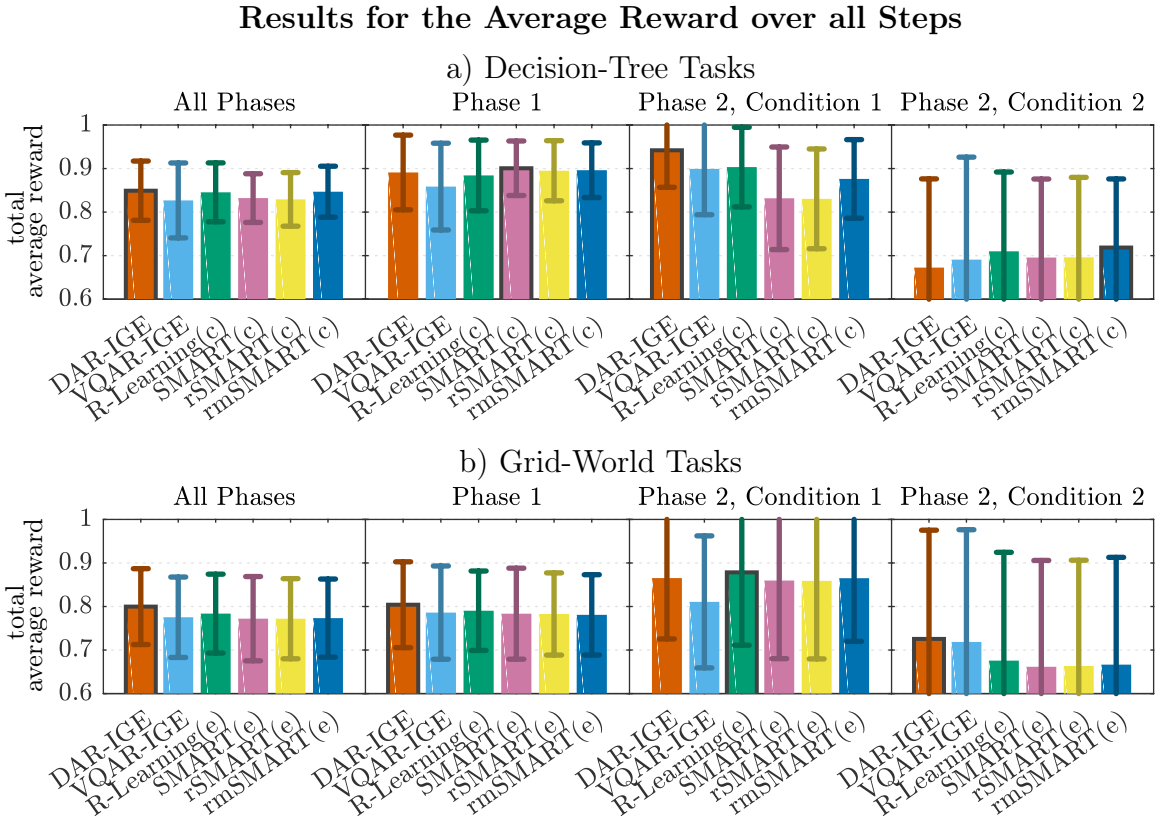


Figure 5.7: Results for the normalized average reward over the steps of all phases, and for the individual phases. The bars show the mean over the 100 tasks per domain, and the error bars the standard deviation. The DAR-IGE had the best performance in both domains. Although, it was not the optimal algorithm for each phase, as indicated by the black border. Different variants of the average-adjusted algorithms were compared for each domain. The continuous (c) performed better in decision-tree tasks, and the episodic (e) in grid-world tasks.

the continuous variants performed better for all algorithms. Whereas, in the grid-world domain, the episodic variants performed better. The following comparison to the AR-IGE concentrates only on the variant of the average-adjusted algorithms that had the best performance per domain. The results for the non-optimal variants can be found in Appendix B.

The DAR-IGE reached the highest total average reward over all phases for both domains (Fig. 5.7). For the decision-tree domain, the advantage over R-Learning(c) and rmSMART(c) was small. Moreover, only in Condition 1 or Phase 2 was the DAR-IGE the best algorithm. In Phase 1, it was SMART(c), and in Condition 2 of Phase 2 it was rmSMART(c). The VQAR-IGE demonstrated a poorer performance compared to all other algorithms.

The advantage of the DAR-IGE over the other algorithms for all phases was stronger in the grid-world domain. Only for Condition 1 of Phase 2 was its performance worse than that of R-Learning(e). The VQAR-IGE demonstrated a better performance than most average-adjusted algorithms, but it was still below that of DAR-IGE. This might be the result of DAR-IGE's ability to perform the ϵ-greedy module selection over its learned set of policies and not modules, whereas the VQAR-IGE is performing exploration over modules. The AR-IGE has more modules with larger γ-values (Figs. 5.5 and 5.6, b). These prefer higher rewards with longer trajectories. Thus, using an exploration over modules is biased towards policies with more steps than shorter policies. This may have resulted in the decreased performance of the VQAR-IGE.

In summary, the DAR-IGE reached the highest average reward over all tasks in both problem domains. However, the difference from the other algorithms is statistically not significant, because of the high standard deviation of the mean average reward over all tasks. In terms of the average-adjusted algorithms, R-Leaning showed the best performance.

Difference between Average Reward per Task

The next measurement compares the average reward over all steps between the DAR-IGE and the average-adjusted algorithms for individual tasks. The difference per task shows for how many of the 100 tasks the DAR-IGE showed a better or worse performance compared to the other algorithms. The difference is measure by:

$$\bar{r}_m^{\text{Diff}} = \bar{r}_m^{\text{DAR-IGE}} - \bar{r}_m^{\text{Algorithm}} \ ,$$

where $\bar{r}_m^{\text{Algortihm}}$ is the normalized average reward of task m for a certain algorithm (Eq. 5.29). The differences per task are sorted from the largest to the smallest for each algorithm (Fig. 5.8). Positive differences show that the DAR-IGE performed better for a certain task compared to the average-adjusted algorithm.

Generally, the DAR-IGE showed a better performance for more tasks compared to the average-adjusted algorithms. In the decision-tree domain, DAR-IGE had a better performance compared to R-Learning(c) for all phases in 60 tasks. In Phase 1, it had a better performance in 72 tasks, in Condition 1 of Phase 2 in 87, and in Condition 2 in 48. For the grid-world domain, the DAR-IGE showed in 65 tasks a better performance compared to R-Learning(e) for all phases. For Phase 1, DAR-IGE was better in 81 tasks, in Condition 1 of Phase 2 in 47, and in Condition 2 in 83.

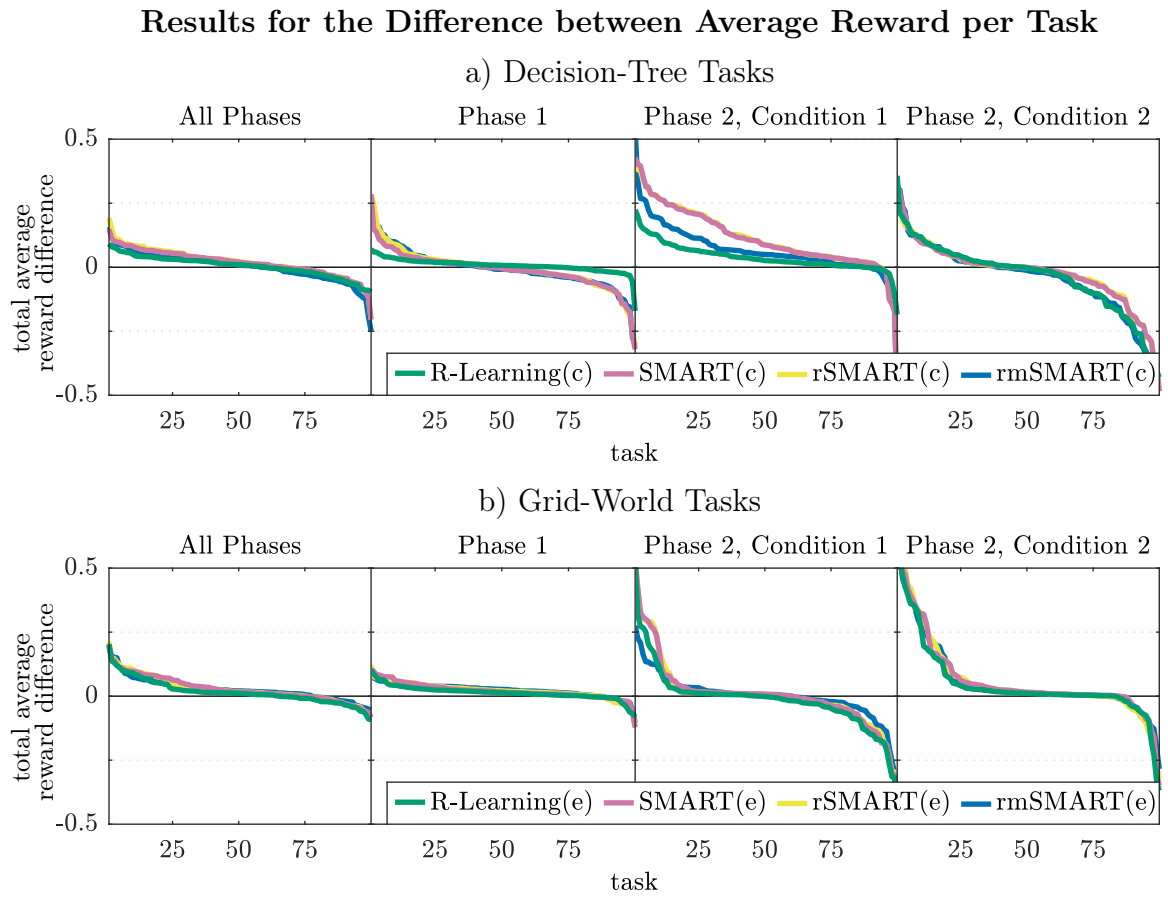


Figure 5.8: Comparison of the difference in average reward over all steps for all phases and per phase between the DAR-IGE and the average-adjusted algorithms. The differences per task are sorted by magnitude. A positive difference means the DAR-IGE performed better in a task, and a negative difference that the other algorithm performed better. The results show that the optimal algorithm depended on the task, but that on average the DAR-IGE performed best.

In summary, the DAR-IGE showed a better performance for more tasks than the average-adjusted algorithms. This tendency was stronger in the more complex domain of grid-world tasks. Nonetheless, the results also show that it depended on the task if a certain algorithm is better compared to another. The properties of a task that influence which algorithm is better suited for it is an open question requiring further investigation.

Average Reward per Episode

The previous measurements evaluated the average reward over all steps of an experiment. The next measurement compares the learning rate of the algorithms based on the average reward per episode. The average reward $\bar{r}_{m,k}$ per episode k was also normalized based on the optimal average reward per task and phase with:

$$\bar{r}_{m,k} = \frac{1}{T_{m,k}} \sum_{t=0}^{T_{m,k}-1} \hat{r}_{m,p(k)}(t) \ ,$$

where $T_{m,k}$ is the number of steps in episode k of task m, and $\hat{r}_{m,p(k)}(t)$ is the adjusted reward of step t by the optimal average reward of the active phase $p(k)$ (Eq. 5.28).

The results show that the DAR-IGE generally had a better learning rate compared to the average-adjusted algorithms (Fig. 5.9). In decision-tree tasks, the learning rate of DAR-IGE and R-Learning(c) were similar for Phase 1, but the DAR-IGE had a higher asymptotic performance. The other average-adjusted algorithms have initially a higher learning rate, but they are overtaken after 200 episodes. For Condition 1 of Phase 2, DAR-IGE outperformed all other algorithms, with R-Learning(c) in the second place. In Condition 2, the DAR-IGE performed worse than the average-adjusted algorithms. Additional experiments showed the performance in Condition 2 can be improved by using a stronger exploration for the module selection (ϵ_γ), but then the performance in the other phases suffered. In grid-world tasks, the DAR-IGE had generally the best learning rate.

In summary, in terms of the average reward per episode, the DAR-IGE showed a better ability to reach a higher asymptotic performance and to learn faster compared to the classical algorithms. Its advantages were especially visible in the more complex domain of grid-world tasks. By learning different policies to reach several goal states, the DAR-IGE could quickly change its policy if a change occurs in the task.

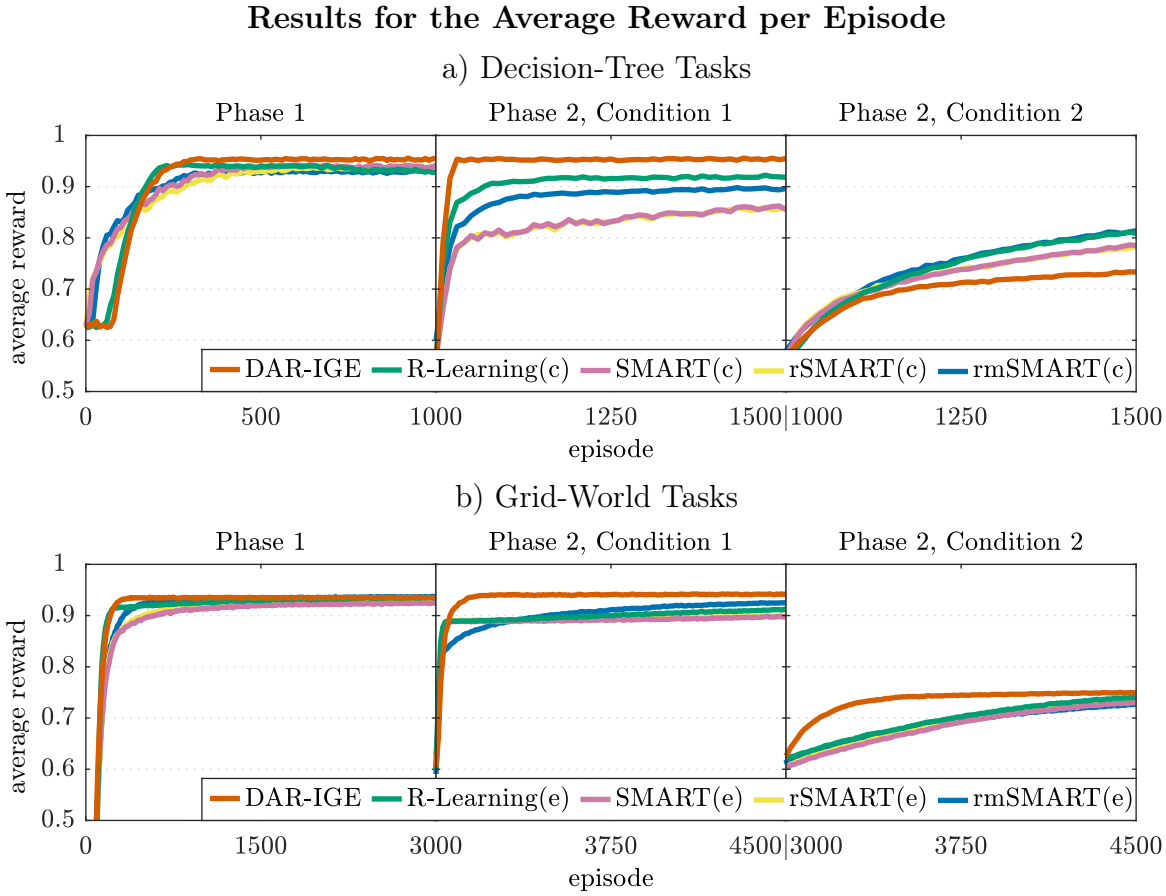


Figure 5.9: Results for the normalized average reward per episode during the learning. The DAR-IGE had a better performance than the average-adjusted algorithms, besides for Condition 2 of Phase 2 of the decision-tree tasks.

5.5.4 Conclusion

The above experiments showed that the AR-IGE outperformed the average-adjusted algorithms slightly in the decision-tree domain. In the more complex domain of grid-world tasks, it increased its performance advantage. However, this is only true for the DAR-IGE. The VQAR-IGE was generally outperformed by the DAR-IGE. One reason for this might be the different exploration strategies of the algorithms. The DAR-IGE explores over policies, whereas the VQAR-IGE explores over modules.

Which of the two variants of the average-adjusted algorithms, either the continuous and episodic form, was better depended for all algorithms on the task domain. For decision-tree tasks, the continuous form was better, and for grid-world tasks the episodic form. The reason for this behavior is an open question which requires further investigation.

5.6 Discussion

The following section discusses in more detail the differences between the AR-IGE and the average-adjusted algorithms in terms of their objectives, i.e. the optimization of average reward per episode compared to the average reward over the steps of all episodes. Afterward, the connection between the performance of the AR-IGE, and its number of γ-modules is discussed.

5.6.1 Average Reward per Episode vs Over All Steps

The difference between the average-adjusted algorithms and the AR-IGE is the objective that they maximize. The AR-IGE maximizes the expected average reward per individual episode:

$$\mathrm{E}\left[\frac{1}{T_k}\sum_{t=0}^{T_k-1} r_{k,t}\right] ,$$

where T_k is the number of steps, and $r_{k,t}$ is the reward at step t of episode k. In contrast, the average adjusted algorithms maximize the expected average reward over the steps of all episodes:

$$\lim_{K\to\infty} \mathrm{E}\left[\frac{1}{\sum_{k=1}^{K} T_k}\sum_{k=1}^{K}\sum_{t=0}^{T_k-1} r_{k,t}\right] ,$$

where K is the number of episodes.

Which objective is more advantageous depends on the problem to be solved. In most scenarios, the goal is to optimize the average reward over the steps of all episodes. The AR-IGE can only guarantee to optimize this goal if the start state is the same for all episodes (Theorem 16). For such problems the AR-IGE is able to yield a better performance compared to the average-adjusted algorithms as the experimental results for the two problem domains showed. However, for problems where the initial state changes according to a distribution $\Pr(s_0)$, the AR-IGE cannot guarantee optimality by using the optimal average reward policy per episode. The example in Fig. 5.10 shows such a case. Thus, only the average adjusted algorithms are guaranteed to converge to the overall optimal policy.

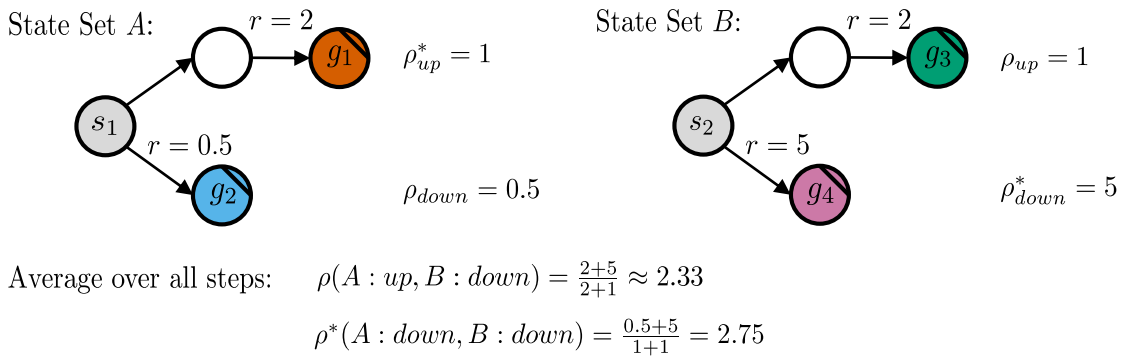


Figure 5.10: An example where the optimal average reward per episode does not optimize the average reward over the steps of all episodes. The MDP consists of two state sets. The initial state of the episodes alternate between s_1 and s_2. The policy to update the average reward per episode is to go up in state set A and down in set B. Nonetheless, the optimal policy to learn the average reward over the steps of all episode is to go down in A and B. The AR-IGE cannot find this policy in such MDPs.

Nonetheless, the AR-IGE might still be favorable for such tasks under certain conditions. The average-adjusted methods need an estimation of the average reward ρ to learn the value function. In contrast, the AR-IGE does not need such an estimation. This could be beneficial in scenarios where the estimation is difficult to compute, for example in cases where the distribution over the start states change over time. Another case would be where the distribution is not i.i.d., such as if the initial state changes only after several episodes. In such cases, the learning of the expected average reward ρ might not converge or is slow. The AR-IGE would not be prone to such problems. Thus, the approximation of the AR-IGE to optimize the overall average reward by learning the optimal average reward per episode could be beneficial. However, further investigations are necessary to identify the possible conditions and scenarios for such cases.

Another advantage of the AR-IGE is that it opens new possibilities that are not available with traditional algorithms. For example, in a multi-agent scenario where the agents operate in the same environment, but have different start locations, the AR-IGE allows them to learn a single shared value function. The value-functions of the IGE depend only on the future expected discounted reward. In contrast, the value function of the average-adjusted agents depends on the average reward that an individual agent gains. Thus, the values differ between the agents for the same state and action pair, because of their different expected average rewards. The sharing of a common value function is therefore not possible with average-adjusted algorithms.

5.6.2 Relationship between Number of γ-Modules and Performance

The AR-IGE is guaranteed to learn the optimal average-reward policy per episodes. For MDPs with choices that have up to N steps, at least one module must exist in the γ regions defined by $\frac{n-1}{n} < \gamma < \frac{n}{n+1}$ for each number of steps $n = (1, \ldots, N)$ (Theorem 17). Depending on the size of the state space, the number of modules to guarantee optimality

might be memory intensive and using less modules would be preferable. It can be shown the that number of modules can be significantly reduced while still having a lower bound on the final performance of the AR-IGE. The next section introduces this relationship and how the γ-modules need to be distributed. It is followed by a numerical evaluation of the performance bound in randomly generated MDPs.

The Performance Bound

Given the minimal performance that should be guaranteed for the policy to which an AR-IGE converges, the number of needed γ-modules can be reduced. The performance is measured by the expected average reward that the AR-IGE achieves after all its modules have converged to the optimal values:

$$J = \mathrm{E}_{\pi^*_\Gamma}\left[\frac{1}{T}\sum_{t=0}^{T-1} r_t\right] ,$$

where π^*_Γ is the optimal policy for an AR-IGE with a set of $\Gamma = (\gamma_1, \ldots, \gamma_M)$ modules. The optimal performance is given by the average reward of the optimal choice $c^* = (r^*, n^*)$ in the MDP with $J^* = \frac{r^*}{n^*}$.

Theorem 17 defines the minimal γ-region $\Gamma(n^*) = (\frac{n^*-1}{n*} < \gamma < \frac{n^*}{n*+1})$ in which modules are guaranteed to learn the optimal choice if it has n^* steps, i.e. the average reward of its choice is larger than any other: $\forall c = (r, n) : \frac{r^*}{n^*} \geq \frac{r}{n}$. The γ-regions for different steps do not overlap under this average reward criteria (Theorem 18). At least one module per region is therefore needed to cover all regions. The average reward criteria can be relaxed so that the modules in a region only guarantees to learn a choice c' with an average reward by a factor κ smaller than any other average reward choice, including the optimal choice: $\kappa\frac{r'}{n'} \geq \frac{r^*}{n^*}$. As a result, the AR-IGE can only guarantee a lower bound on the average-reward performance:

$$J \geq \kappa\frac{r^*}{n^*} .$$

However, the γ-regions $\Gamma(\kappa, n')$ that use this criteria are wider and overlap with each other. This allows to represent several regions by a single module, reducing the number of needed modules to cover all regions. Theorem 21 defines the upper and lower boundaries of such regions. It is a generalization of Theorem 17.

Theorem 21. *For any deterministic, goal-only-reward MDP exists a γ-region $\Gamma(\kappa, n')$ for which all its discount factors γ_κ are guaranteed to encode the choice $c' = (r', n')$ if its average reward is at most by a factor κ smaller than the optimal average reward:*

$$\kappa\frac{r'}{n'} \geq \frac{r^*}{n^*} ,$$

with $\kappa \in [0\ 1]$. The region is defined by:

$$\Gamma(\kappa, n') = \{\gamma_L(\kappa, n') < \gamma_\kappa < \gamma_U(\kappa, n')\} ,$$

where $\gamma_L(\kappa, n')$ and $\gamma_U(\kappa, n')$ are the lower and upper border of the region, defined by

$$\gamma_L(\kappa, n') = \max_n \left(\frac{\kappa(n)}{n'}\right)^{\frac{1}{n'-n}} \quad s.t.\ 1 \leq n < n' \tag{5.30}$$

and

$$\gamma_U(\kappa, n') = \min_n \left(\frac{n'}{\kappa n} \right)^{\frac{1}{n-n'}} \quad s.t.\ n' < n\ . \tag{5.31}$$

Proof. Because Theorem 21 is a generalization of Theorem 17, the proof follows the same steps. The γ region for which the choice c' is the solution given all other possible choices in the MDP is defined by:

$$\gamma_L(\kappa, n') = \max_{\forall c_i < c'} \gamma_E(c_i, c') < \gamma_\kappa < \gamma_U(\kappa, n') = \min_{\forall c_j > c'} \gamma_E(c', c_j)\ ,$$

where $\gamma_E(c_x, c_y)$ is the discount factor for which the choices c_x and c_y have the same value (Eq. 2.6, page 38). The lower bound can be solved by following the steps of Eq. 5.7 to Eq. 5.9 of the proof for Theorem 17:

$$\begin{aligned}
\gamma_L(\kappa, n') &= \max_{\forall c_i < c'} \gamma_E(c_i, c') && \text{s.t. } n_i < n',\ \kappa\tfrac{r'}{n'} \geq \tfrac{r_i}{n_i},\ n_i, n' \in N^+ \\
&= \max_{r', r_i, n_i} \left(\tfrac{r_i}{r'} \right)^{\frac{1}{n'-n_i}} \\
&= \max_{r', n_i} \left(\frac{\kappa n_i \frac{r'}{n'}}{r'} \right)^{\frac{1}{n'-n_i}} \\
&= \max_{n_i} \left(\tfrac{\kappa n_i}{n'} \right)^{\frac{1}{n'-n_i}} && \text{s.t. } n_i < n',\ n_i, n' \in N^+\ .
\end{aligned}$$

The upper bound can be proven in a similar way following the steps of Eq. 5.10 to Eq. 5.11:

$$\begin{aligned}
\gamma_L(\kappa, n') &= \min_{\forall c_j > c'} \gamma_E(c', c_j) && \text{s.t. } n_j > n',\ \kappa\tfrac{r'}{n'} \geq \tfrac{r_j}{n_j},\ n_j, n' \in N^+ \\
&= \min_{r', r_j, n_j} \left(\tfrac{r'}{r_j} \right)^{\frac{1}{n_j-n'}} \\
&= \min_{r', n_j} \left(\frac{r'}{\kappa n_j \frac{r'}{n'}} \right)^{\frac{1}{n_j-n'}} \\
&= \min_{n_j} \left(\tfrac{n'}{\kappa n_i} \right)^{\frac{1}{n_j-n'}} && \text{s.t. } n_j > n',\ n_j, n' \in N^+\ .
\end{aligned}$$

□

The lower (Eq. 5.30) and the upper bound (Eq. 5.31) of a region $\Gamma(\kappa, n')$ can be solved numerically by defining a maximum number of steps N in an MDP and identifying the maximum or minimum over all possible steps $n = (0, \ldots, N-1)$. Theorem 22 in Appendix C provides the analytical solution for the boundaries, but its proof is incomplete.

If the performance boundary parameter κ is smaller than 1, then the γ-regions that learn the optimal choice for a certain number of steps n', begin to overlap (Fig. 5.11). The smaller κ is, the more regions overlap. γ-Modules in an overlapping region are guaranteed to learn the optimal choice with the number of steps n' of all the overlapping regions. This allows a reduction of the number of needed modules to guarantee that every choice according to the lower bound can be learned in an MDP. Algorithm 5.5 is used to identify the reduced set of γ-regions $\hat{\Gamma}$ in which γ-modules are needed. At first the set of all regions $\Gamma(\kappa) = \big((\gamma_L(\kappa, 1), \gamma_U(\kappa, 2)),\ \ldots\ , (\gamma_L(\kappa, N), \gamma_U(\kappa, N))\big)$ for each possible n' is sorted according to its upper boundaries. Then, starting with the first region, all other regions in $\Gamma(\kappa)$ are identified which overlap with it, i.e. their lower bounds are smaller than γ_{U_1}, and their upper bounds are larger than γ_{U_1}. For all of these regions, a new

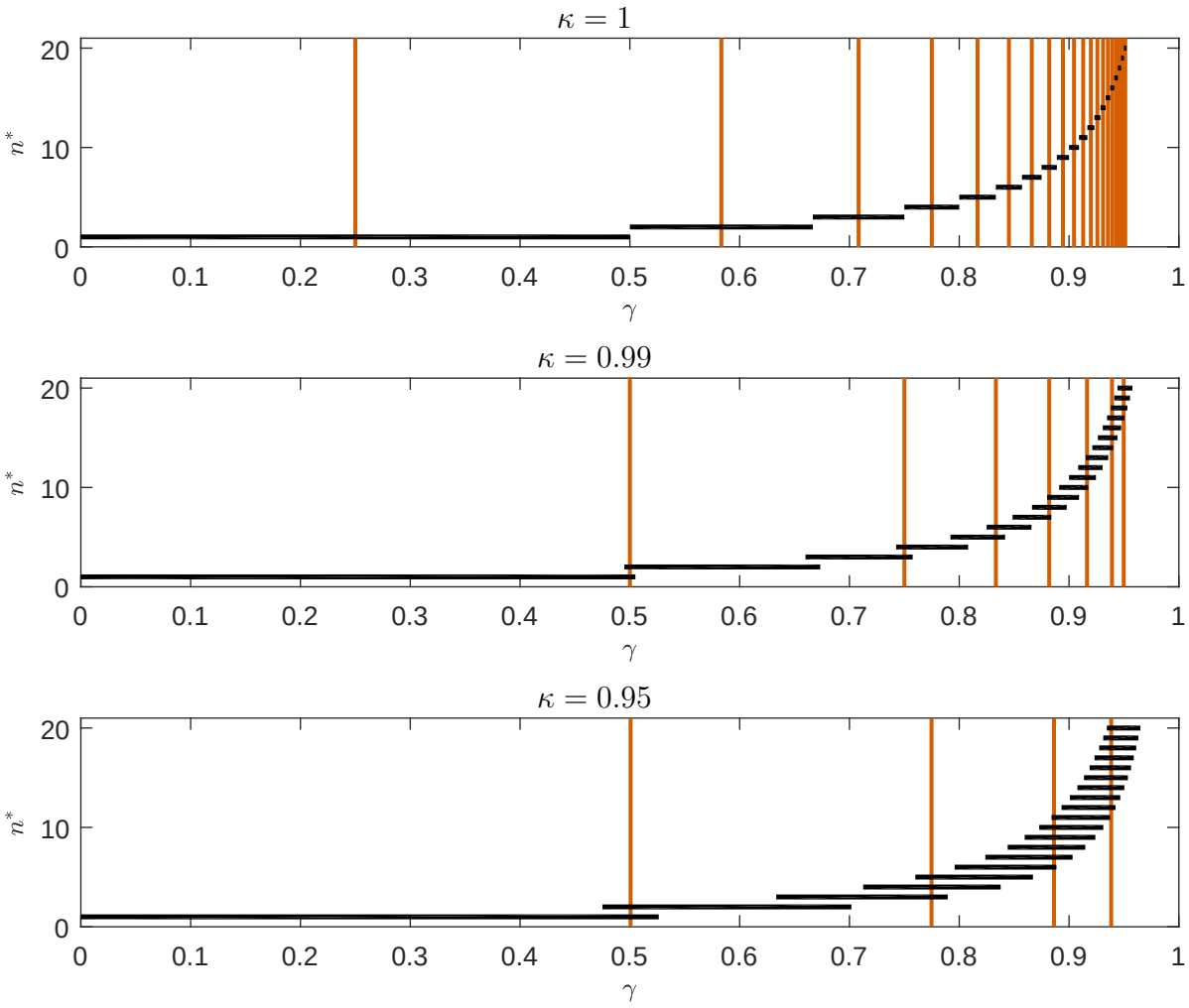


Figure 5.11: The minimum γ-regions that are guaranteed to learn the optimal average reward choice with different number of steps n' and for different performance bound parameters κ. The regions are shown as horizontal, black lines. For $\kappa = 1$, the regions do not overlap (Theorem 18). For $\kappa < 1$ the regions are larger and overlap which results in a smaller number of modules that are needed to cover all of them. The colored, vertical lines show the minimal number of γ-modules, needed to cover all regions.

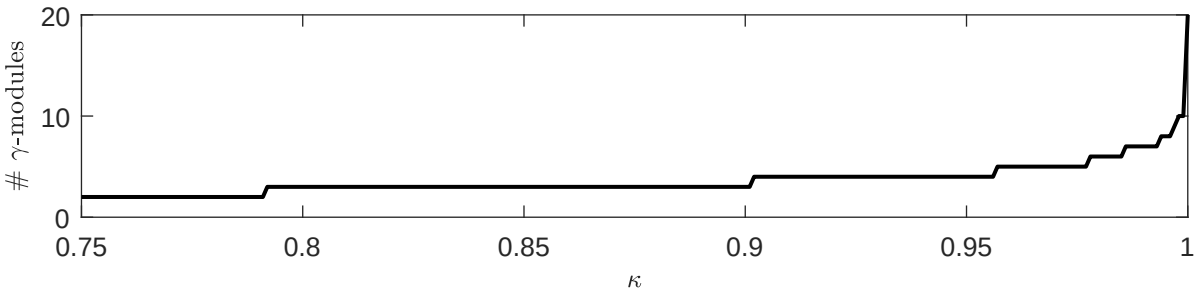


Figure 5.12: The number of γ-modules needed to guarantee a performance bound of $J \geq \kappa \frac{r^*}{n^*}$ decreases strongly for $\kappa < 1$. The example is for MDPs with up to 20 steps.

Algorithm 5.5: Identification of the reduced set of γ-regions $\hat{\Gamma}$

Input:

γ regions for each number of steps up to N:

$\Gamma(\kappa) = \Big(\big(\gamma_L(\kappa,1),\gamma_U(\kappa,2)\big),\ \ldots\ ,\big(\gamma_L(\kappa,N),\gamma_U(\kappa,N)\big)\Big)$

sort regions Γ_n according to upper borders γ_U

initialize $\hat{\Gamma}$ empty

while $\Gamma(\kappa)$ *is not empty* **do**

`// identify overlapping region`

$i \leftarrow$ first region in $\Gamma(\kappa)$

$K \leftarrow$ all regions in $\Gamma(\kappa)$ with $\gamma_L < \gamma_{U_i}$ and $\gamma_U \geq \gamma_{U_i}$

`// make a new region that encloses all overlapping regions`

add region $(\max_{k\in K}\gamma_{L_k}, \gamma_{U_i})$ to $\hat{\Gamma}$

`// remove overlapping regions from` $\Gamma(\kappa)$

remove regions i and K from $\Gamma(\kappa)$

end

return $\hat{\Gamma}$

region is added to the reduced set of regions with $\gamma_L = \max_{k\in K}\gamma_{L_k}$ and $\gamma_U = \gamma_{U_1}$. All overlapping regions are removed from $\Gamma(\kappa)$ and the next region in it is used to identify all other regions that overlap with it, until all regions are removed from $\Gamma(\kappa)$.

By introducing a performance bound $J \geq \kappa\frac{r^*}{n^*}$ with $\kappa < 1$, significantly fewer γ-modules are needed to guarantee it (Fig. 5.12). For example, in MDPs with up to 20 steps for its optimal choices, 20 modules are needed to guarantee the learning of the optimal average reward choice (Fig. 5.11, top). If a performance bound of $\kappa = 0.99$ is introduced, only 7 modules are needed (Fig. 5.11, middle). The number of modules reduces to 4 for $\kappa = 0.95$ (Fig. 5.11, bottom).

Numerical Evaluation

With γ-modules distributed according to Theorem 21, the AR-IGE can guarantee the lower performance bound of $J \geq \kappa\frac{r^*}{n^*}$ on its final performance. Nonetheless, for many MDPs the reachable performance might be better than the guaranteed lower bound. The final performance of randomly generated MDPs were evaluated to test how strong the final performance of the DAR-IGE and the VQAR-IGE is affected by using a reduced set of γ-modules.

The MDPs were represented by their choice sets that define all possible choices in an MDP. 10,000,000 choice sets where randomly generated. Each choice set had 4 to 6 choices. The reward and the number of steps of each choice were uniformly sampled with $r \sim U(0,100) \in \mathbb{R}$ and $n \sim U(1,20) \in \mathbb{N}$. The choices are sampled to be non-dominated (Theorem 4, page 39) by first drawing all rewards and steps and then sorting them. For each MDP, the optimal value function for its start state can be computed based on the

choice set by:

$$V_\gamma^* = \max_c \gamma^{n_c-1} r_c \ .$$

Based on the optimal value function, it was evaluated which module each algorithm would choose as active module $\tilde{\gamma}$. The average reward of the active module's choice $\tilde{c} = \text{argmax}_c \, \tilde{\gamma}^{n_c-1} r_c$ was compared to the optimal average reward for the MDP:

$$\tilde{\kappa} = \frac{\bar{r}(\tilde{c})}{\bar{r}(c^*)} \ , \text{with} \ \ \bar{r}(c) = \frac{r_c}{n_c} \ ,$$

showing the factor $\tilde{\kappa}$ by which its is smaller.

Both algorithms were tested with γ-modules based on three different performance bounds with $\kappa = (1, 0.99, 0.95)$ (Fig. 5.11). For the DAR-IGE three modules were used per γ-region $\hat{\Gamma}(\kappa)$ identified by Algorithm 5.5. The VQAR-IGE used only one module

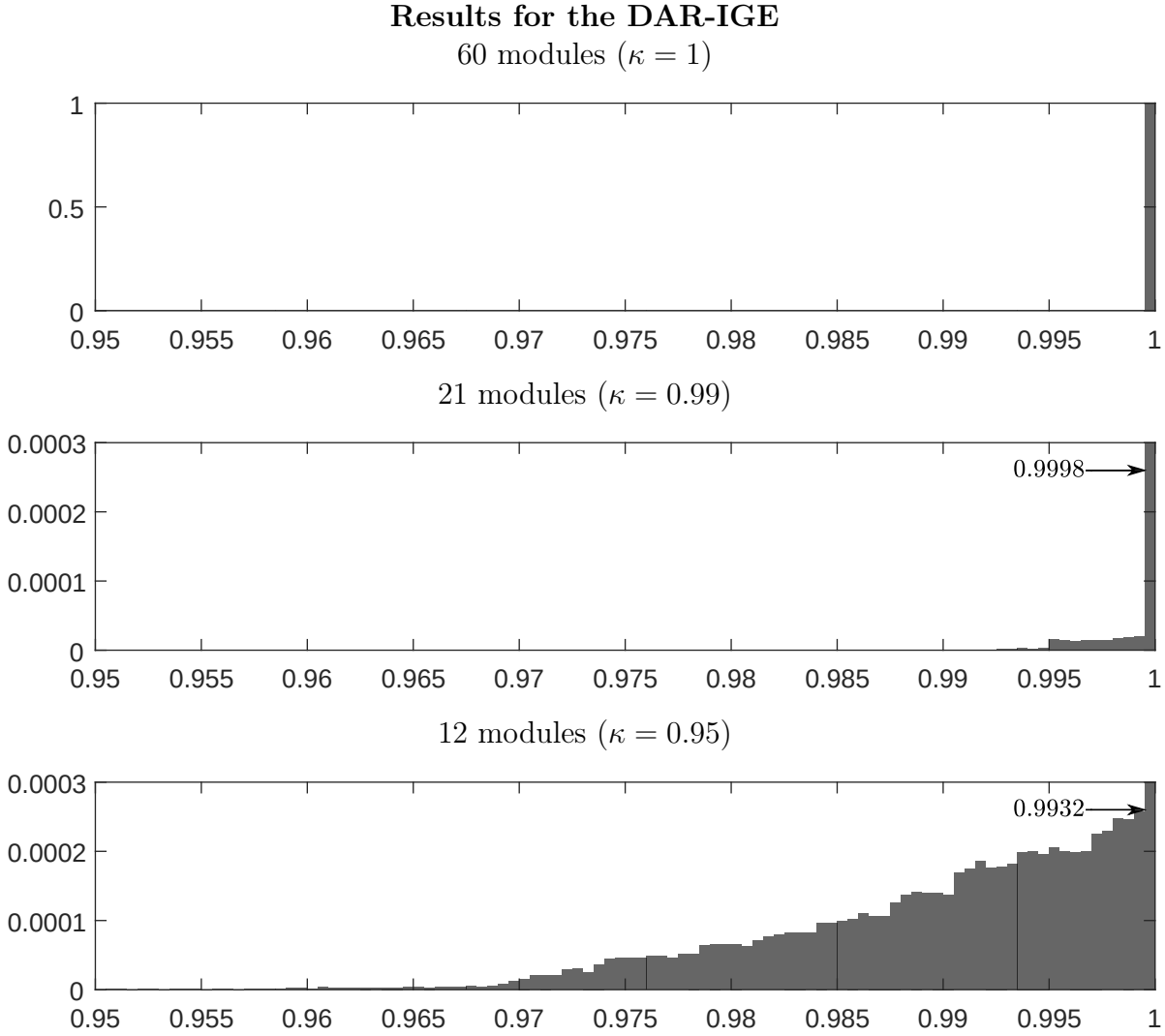


Figure 5.13: The DAR-IGE does not violate the performance bound $\kappa \frac{r^*}{n^*}$ used for deciding the number of modules. Furthermore, the majority of MDPs can be optimally solved. (n = 10,000,000, binsize = 0.005)

per region. The results show that both algorithms kept for most MDPs the lower performance bounds (Figs. 5.13 and 5.14). Moreover, for the majority of generated MDPs both algorithms guaranteed a performance with $\tilde{\kappa} \geq 0.9995$. For $\kappa = 1$, both algorithms guaranteed the optimal performance for all MDPs. In the case of $\kappa = 0.99$, the DAR-IGE guaranteed a performance with $\tilde{\kappa} \geq 0.9995$ in 99.98% of all generated MDPs. The VQAR-IGE guaranteed it in 98.7% of all MDPs. In the case of $\kappa = 0.95$, the DAR-IGE guaranteed optimal performance in 99.32% of all MDPs, and the VQAR-IGE in 94.05%.

For a few MDPs, the algorithms could not keep the bound $\kappa\frac{r^*}{n^*}$. In the case of the DAR-IGE, the reached performance was lower than the bound for 119 MDPs for $\kappa = 0.99$, and 646 MDPs for $\kappa = 0.95$. In these MDPs several choices exist that fulfill the boundary. As a result, some of the three modules in the γ-region that is guaranteed to learn them learns one choice and other modules the other choice. Because the DAR-IGE needs that

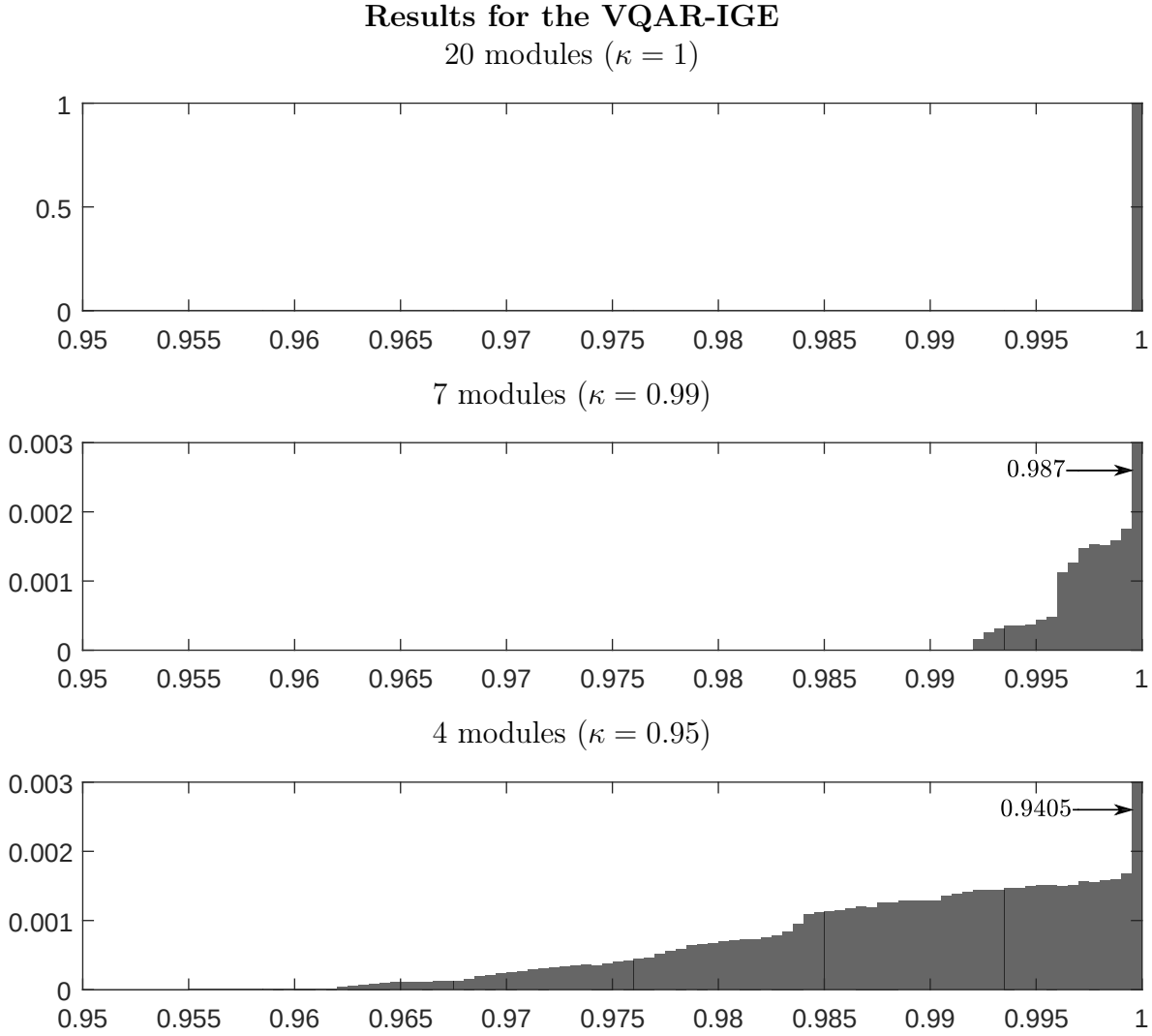


Figure 5.14: The VQAR-IGE does not violate the performance bound $\kappa\frac{r^*}{n^*}$ used for deciding the number of modules. Furthermore, the majority of MDPs can be optimally solved . (n = 10,000,000, binsize = 0.005)

all three modules learn the same choice to identify its properties, it can not identify those choices and selects therefore another module that results in a lower average reward. The VQAR-IGE reaches a lower performance than the given bound in 8 MDPs for $\kappa = 0.99$, and 23 MDPs for $\kappa = 0.95$. It performs generally more poorly compared to the DAR-IGE in cases with $\kappa < 1$, because its minimum value quotient F_V is defined based on the assumption that the optimal number of modules ($\kappa = 1$) is used. Therefore, it sometimes fails to identify the best policy its γ-modules have learned. Nonetheless, the cases where both algorithms fail to ensure the performance bound are minuscule in relation to the $10,000,000$ tested MDPs.

Conclusion

Based on Theorem 21 the number of γ-modules can be reduced while still guaranteeing that one module learns a policy that fulfills the minimum performance bound $J \geq \kappa \frac{r^*}{n^*}$. For MDPs with a maximum of 20 steps for the optimal choice, the number can be reduced from 20 modules to 7 modules with a performance guarantee of achieving at least 99% of the optimal average reward. The numerical evaluation showed that even in these cases, for 99% of the uniformly sampled MDPs, the optimal performance can still be reached. This allows a reduction of the number of modules in cases of high memory demands, with a minimal loss of performance.

5.7 Conclusion

The AR-IGE optimizes average reward in deterministic, goal-only-reward MDPs. It uses a different approach compared to the existing model-free, value-based algorithms. These algorithms learn an average-adjusted value function, similar to the discounted value function. In contrast to the AR-IGE, average-adjusted algorithms need to learn an extra estimate for the average reward of the current policy.

The AR-IGE optimizes the average reward based on its ensemble of γ-modules that use discounted value functions. Two variants of the AR-IGE exist, the DAR-IGE and the VQAR-IGE. The DAR-IGE decodes for each of its learned γ-policies the expected reward and number of steps required to reach their goal states. This information allows the calculation of the expected average reward per episode for each policy, and selection of the best one. The VQAR-IGE identifies the policy resulting in the maximum average reward by comparing the values of the γ-modules directly. The quotient between the module, that is guaranteed to learn the optimal policy if the trajectory has a certain number of steps, and all other modules has a lower bound. The lower bound is only fulfilled by the module that learned the optimal average-reward policy.

Both AR-IGE variants are guaranteed to learn the optimal policy if they distribute the γ factors of their modules in a specific way. In MDPs where the optimal trajectory has up to N steps, the IGE needs a module in each γ-region with $\frac{n-1}{n} < \gamma < \frac{n}{n+1}$ for all $n = (1, \ldots, N)$. If the optimal trajectory has n steps, all modules in the associated γ-region are guaranteed to learn it. As a result, the DAR-IGE needs $3N$ modules because it requires three modules that learned the same policy to guarantee its average reward can be decoded. The VQAR-IGE needs only N modules. However, the DAR-IGE allows

the decoding of the average reward for all of its policies, which can be used to improve, for example, the exploration. The experimental results showed the DAR-IGE generally has a better performance than the VQAR-IGE.

The number of modules needed can be reduced if a lower bound on the performance of the AR-IGE is accepted. Given the final average-reward performance J is bounded by $J \geq \kappa \frac{r^*}{n^*}$, where $\frac{r^*}{n^*}$ is the optimal average reward in the MDP, then fewer modules are needed to guarantee this performance. With $\kappa < 1$ the γ-regions that guarantee to learn the best policy enlarge and overlap with each other. This makes it possible to represent several regions by a single γ-module. In tasks with a maximum of $N = 20$ steps and $\kappa = 0.99$ the number of modules can be reduced by a factor of three.

The main difference between the AR-IGE and the average-adjusted algorithms is their objectives. The AR-IGE optimizes the average reward per episode. The average-adjusted algorithms optimize the average reward over the steps of all episodes. Only for the class of deterministic, goal-only-reward MDPs, where each episode starts in the same initial state, result both objectives in the same optimal policy. Generally, the average reward over the steps of all episodes is the objective that should be optimized. Nonetheless, there are cases where the solution of the AR-IGE might be of benefit. These could be in tasks where the estimate of the average reward of the current policy, as needed by the average-adjusted algorithms, might be difficult to obtain. However, further investigations are necessary to identify such cases and their characteristics.

A limitation of the AR-IGE is that it can guarantee optimality only for deterministic, goal-only-reward MDPs. This is a strong restriction on the type of tasks the framework can solve. The AR-IGE could be applied to the general MDPs by learning predictions for the expected average reward and number of steps of each policy, as done by the Objective-Adaptive IGE (Section 4.4). Nonetheless, in this case it cannot be guaranteed that the optimal policy will be learned by γ-modules.

Although the AR-IGE has limitations, it provides an interesting new perspective to the optimization of average reward. One advantage over average-adjusted algorithms is that the AR-IGE does not need to learn an estimate for the average reward ρ of its current policy. This reduces the complexity of the algorithm and makes it unnecessary to find a good learning rate parameter α_ρ for the estimate. Moreover, the AR-IGE learns not only one policy but several, between which the agent can choose. This proved to be useful in MDPs where the optimal goal changes, because the AR-IGE can quickly adapt by using one of its γ-policies. It can also obtain detailed information about the different policies, i.e. the expected reward sum and number of steps. This is useful if the average reward should be optimized under some restrictions, for example if at least a certain reward per episode should be reached, as shown in Section 4.4. The experimental results also showed that the AR-IGE performs better compared to the average-adjusted algorithms in deterministic, goal-only-reward MDPs in the domain of decision-tree tasks and grid-world tasks.

Conclusion

The Independent γ-Ensemble (IGE) is a brain-inspired reinforcement learning framework. It allows artificial systems to be more adaptive to changes in tasks and objectives. The IGE is composed of several γ-modules representing independent learning agents. Each module learns a policy to optimize the discounted expected reward ($\sum_{t=0}^{\infty} \gamma^t r_t$) with a different discount factor γ.

The IGE is not the first framework based on modules with different discount factors. The Horde architecture uses the same idea to predict future events and sensor values on different time-scales (White, Modayil, & Sutton 2012). Another approach, the Dependent γ-Ensemble, models hyperbolic discounting as observed in human decision-making with help of the modules (Kurth-Nelson & Redish 2009). However, the IGE is the first framework based on different discounting modules, that uses their abilities to adapt to different task contexts and objectives.

The IGE modules learn different policies, called γ-policies, which represent different solutions to the trade-off between the amount of reward that can be gained and the time to acquire it. Large γ's optimize long-term reward, preferring policies that yield a high reward sum, even though the agent may need more time to collect it. Smaller γ's optimize short-term reward, maximizing reward on a short timescale, leading to behaviors that provide reward with a short delay. Each module uses Q-learning as its learning algorithm. Because Q-learning is an off-policy algorithm, the IGE learns its policies in parallel based on the same observations. Additionally, the IGE can decode the expected reward and the number of steps to gain it for its policies in deterministic, goal-only-reward MDPs. Decoding is possible from values of two modules that learned the same policy, i.e. to reach the same goal state in an MDP.

Learning different policies and decoding their properties allows the IGE to adapt to different task contexts and objectives. It also allows optimization of the average reward. The following sections summarize the application of the IGE to these scenarios and discuss open questions and limitations. Future research directions, and decision-making in humans are discussed at the end of the section.

Transfer Learning between Different Task Contexts

The IGE is used to transfer knowledge between different tasks contexts (Chapter 3). In a certain task context, the IGE learns not only the optimal policy for that context, but also several other policies. If the task context changes, the agent can select the most appropriate policy for the new context. Given a context signal, a γ-map from the context signal to the module with the most appropriate policy can be constructed. For example, in an energy foraging task, the agent has to collect energy from different sources in its

surroundings, but it also expends energy moving around. The agent's energy level restricts the distance it can travel. If it has a high energy level, it can travel long distances to reach high energy sources. With a small energy-level, it has to secure a nearby energy sources before it runs out of energy, even though these energy sources might not have as much energy as the more distant ones. The context for the agent is its energy level. A γ-map can be constructed that selects policies preferring nearby energy sources (small γ) if the agent's energy level is low, and distant high-energy sources (large γ) if the agent's energy level is high.

Three tasks have been introduced for which an optimal γ-map can be analytically constructed. Interruption Risk MDPs (IR-MDPs) have a constant risk that the agent gets interrupted during the task, meaning it can not get the reward. Reward Risk MDPs (RR-MDPs) are similar. They have the constant risk that all future rewards becomes zero. The context in these MDPs is the risk of interruption or loss of future reward. In Constant Punishment MDPs (CP-MDPs), a constant punishment is given for each step. The strength of the punishment is the context. For all three MDPs, a mapping from the risk or punishment level to the optimal γ-module can be constructed. It is proven that the maps select the optimal module for any MDP of these types.

For general MDPs, the γ-map has to be learned. Given a new task context, the agent has to learn which of its γ-policies is the most appropriate under the new context. The learning of the map is a reinforcement learning problem by itself and can be solved by various methods. Policy-based (PGPE, EPHE) and value-based (Q-learning) methods have been introduced. They were able to successfully learn γ-mappings for several energy foraging tasks. The IGE had a significantly higher learning speed compared to agents that have to adapt to new task contexts without prior knowledge.

One of the critical points for the transfer is that the task stays similar enough between contexts to allow a successful transfer of policies. In the analyzed tasks, only the reward function changed between the contexts. Moreover, the reward function used to learn the policies of modules should be invariant to changes in the context. Otherwise, modules would unlearn and relearn their policies depending on the context. To avoid this, policies are only learned under one context, or the reward function was decomposed in a part that is invariant to context changes. Further investigation into methods that can cope with reward functions that change over task contexts is necessary in order for the IGE to be applied to a wider range of tasks. Such methods could include selective learning, where the agent does not include certain observations from certain contexts that would destroy its learned policies. Although the question about when a transfer can be helpful remains open. The IGE showed, as the first approach in the field of transfer learning, that policies based on different discounting factors can be successfully transferred between task contexts.

Zero-shot Learning of Different Task Objectives

Beside the adaptation to different task contexts, the IGE is also able to adapt to different task objectives (Chapter 4). In this case, the IGE can immediately adapt without learning a γ-map, as in the case of context adaptation. This form of learning is also known as zero-shot learning.

Task objectives are formulated as functions f that should be maximized. They take

as input the reward sum r the agent gains in an MDP, and the time steps n that the agent needs, such as $f(r, n) = r - \frac{1}{2}n$. An example scenario is an agent that should find the best way to a restaurant, where the reward strength describes the food quality of the restaurant. In this case, the objective might change from day to day. One day the restaurant with the best food should be visited. Another day there is some time restriction, and the agent should take the best restaurant that can be reached within a certain time limit. Again, the IGE can adapt to changes in the objective by using different γ-policies. The IGE selects the most appropriate policy according to its resulting expected reward sum, and the time needed to obtain the rewards.

For deterministic, goal-only-reward MDPs, the IGE can decode this information directly from its learned Q-functions. For general MDPs, the IGE uses an approach inspired by the Horde architecture. For each module, it learns a prediction of the reward sum that the module's policy achieves, and the number of steps needed to acquire the rewards. Based on this prediction, the IGE selects the most appropriate policy to maximize a given objective.

However, the approach has a limitation in regard to the selection of the most appropriate policy. The aim is to optimize the expectation over the outcome of an objective: $\text{argmax}_\gamma \, \text{E}_{\pi_\gamma}[f(r, n)]$. But, the IGE learns the expectation only over the reward sum $\text{E}_{\pi_\gamma}[r]$ and the number of steps $\text{E}_{\pi_\gamma}[n]$, and not over the objective. It can only approximate the expected outcome of the objective by $\text{E}[f(r, n)] \approx f(\text{E}[r], \text{E}[n])$. For some types of environments and objectives, such as deterministic MDPs, the correctness of the approximation can be guaranteed, but not for MDPs and objectives in general. Nonetheless, the IGE showed that it could immediately adapt to a new objective, in contrast to a classical algorithm that needed several hundred episodes before it could reach the same performance as the IGE.

Average Reward Optimization

One objective that the IGE is guaranteed to solve is the optimization of average reward per episode in deterministic, goal-only-reward MDPs (Chapter 5). Optimization of average reward is especially important in tasks that are repeated in a loop, because it results in the highest reward for the invested time. In MDPs where each episode starts at the same initial state, the IGE also optimizes the average reward over the steps of all episodes.

The IGE provides a new way to optimize average reward, compared to existing model-free, value-based methods. Instead of learning only a single policy, it learns several, which can be helpful to adapt the agent to changes in the task. Furthermore, the IGE does not need to learn an approximation of the average reward ρ of the agent's policy, as existing methods have to do to learn their value functions. This reduces learning complexity and the number of learning parameters. The IGE outperformed existing methods in decision-tree and grid-world tasks.

The convergence of the IGE to the optimal average-reward policy could be proven. Furthermore, which modules, i.e. which γ factors are needed guarantee that the IGE is optimal, can be defined exactly. If the MDP has a maximum of N steps for its optimal trajectory, then the IGE needs either N or $3N$ modules to guarantee optimality, depending on the IGE variant used. Nonetheless, the number of modules needed can be reduced while still guaranteeing a lower bound for the final performance.

General Limitations

The IGE in its current form has two major limitations. The first is with respect to the policies it is able to learn. Its modules can learn a subset of policies that optimize the discounted reward sum ($\sum_{t=1}^{\infty} \gamma^t r_t$) for different γ's. These γ-policies form the behavioral repertoire that the IGE uses to adapt to different task contexts and objectives. However, generally it cannot guarantee that the γ-policies represent the optimal solution for each context or objective.

This can be analyzed in more depth in the class of deterministic, goal-only-reward MDPs, where the policies and trajectories the IGE can learn are represented as choices. The IGE learns to reach nearby choices with modules that have small γ's and more distant choices that give more reward with larger γ's. However, the IGE is not guaranteed to learn every possible choice. For one thing, the IGE does not learn dominated choices, which give less reward and need longer to reach than other choices in the MDP (Theorem 4). Nonetheless, such choices are not important because generally only choices in the set of non-dominated choices are solutions for different task contexts or objectives. However, the IGE does not learn the whole set of non-dominated choices. As shown by Theorem 7 some non-dominated choices are excluded.

However, for some task contexts and objectives, the IGE is guaranteed to learn the optimal policy. It is guaranteed to learn the path to the choice with the fewest steps and to the choice with the highest reward. Most interesting, it can guarantee learning the choice with the highest average reward in an episode (Theorem 17). In terms of tasks, the IGE is guaranteed to learn optimal policies for IR-MDPs, RR-MDPs, and CP-MDPs.

Although the IGE is not guaranteed to learn optimal policies with regard to objectives with arbitrary temporal weighting, it can still improve the learning performance under a new task context or objective by using its most appropriate policy. The experimental results in Chapters 3 and 4 show that this helps to adapt faster to a new situation than a standard agent that learns the new situation without any prior information. In some cases, this standard agent finally achieved a higher performance, but it usually needed hundreds or thousands of episodes to do so. The IGE could also be allowed to find the optimal policy if it is combined with a general reinforcement learning agent. In such a case, the IGE could be used to jump-start or inform the general learner in a new task context or objective by providing a good initial policy. Similar methods have been already explored in the field of transfer learning (Fernández & Veloso 2006; Mehta et al. 2008). In contrast to the IGE, they used policies that were solutions for previous tasks, and not alternative solutions for the current task.

A second general limitation of the IGE exists in regard to decoding extra information for its policies. Theoretically, if the IGE has several modules that learned the same or a similar policy, its resulting expected reward trajectory could be decoded. Decoding is based on the value function from all these modules, using the inverse of a Vandermonde matrix, constructed from the modules discount factors. Unfortunately, this method has problems, such as that the inverse of the Vandermonde matrix is ill-conditioned, making it difficult to apply in practice.

Nonetheless, decoding can be successfully done in deterministic, goal-only-reward MDPs. Unfortunately, such MDPs represent only a very restricted class of problems. However, a similar approach as used by the Horde architecture, with its control and

prediction demons (Sutton, Modayil, et al. 2011), can be applied to predict information needed about the policies. For example, it is possible to predict the expected reward sum and number of steps until a goal state is reached for any γ-policy in general MDPs. This knowledge has been successfully applied to identify an appropriate policy for new task objectives (Chapter 4).

Future Research

In addition to future research into approaches to overcome current limitations of the IGE, several other research directions have been opened by the IGE. The following section lists some of these potential research directions.

The basic idea of the IGE is to have an ensemble of individual reinforcement learning modules, each optimizing discounted reward with a different discount factor. There is no restriction on how to implement the modules, making the IGE a very general framework that can be combined with many approaches. In this work, Q-learning was used as the learning algorithm for modules. It allowed learning in a model-free approach, requiring no model of the environment. Moreover, in contrast to policy-based algorithms, it learns value functions that can be used to decode further information about trajectories resulting from policies. Q-learning is also a off-policy method, allowing all modules to learn their optimal value function from any observations.

However, the IGE can be combined with any reinforcement learning algorithm that optimizes the discounted reward sum. It could be combined with model-based algorithms to improve the learning of the value function. In this case, the IGE needs only a single model for all modules. Each module learns its individual value function based on the shared model. The IGE could also be combined with policy-based methods, such as policy gradient approaches. This allows the application of the IGE to domains where such methods are successfully used, for example for the control of robots (Deisenroth, Neumann, & Peters 2011). A problem is that policy gradient approaches are usually on-policy methods. Thus, modules cannot learn in parallel from observations produced by the policy of another module. A potential solution would be to use importance sampling (Rubinstein & Kroese 2016) to allow off-policy updates of modules in parallel (Uchibe & Doya 2004). Future research should identify how resilient such methods are if used with the IGE.

The general nature of the IGE also allows use of different model-free, value-based algorithms and techniques. In this first study of the IGE, only basic Q-learning with a tabular value function was used. Nonetheless, the IGE could be combined with more sophisticated approaches such as eligibility traces (Precup, Sutton, & Singh 2000) to improve its learning speed, or function approximation to allow continuous state spaces and generalization over states (Sutton & Barto 1998). In particular, the combination with deep neural networks (Goodfellow, Bengio, & Courville 2016) to represent the value function seems promising. Deep Q-learning has been recently successfully applied to different complex tasks, such as video games (Mnih et al. 2015), Go (Silver, Schrittwieser, et al. 2017), or other board games (Silver, Hubert, et al. 2017). The IGE can be combined with such methods. All modules could share the same first layers of a deep neural network to allow learning of shared low-level features. Higher layers could then differentiate into separate modules, each learning a value function based on a different discount factor. This

would allow to use the IGE for complex tasks, based on high-dimensional state spaces, such as screen images from computer games. In summary, the IGE is a very general framework that can be combined with many approaches. Future research should identify which combinations are useful, and should determine whether the unique structure of the IGE might provide special synergy with specific approaches.

A further research direction is to identify further task types in which the IGE can be successfully applied. The IGE learns policies that represent different strategies for the trade-off between the amount of reward and the time to acquire it. In this work, it was mainly applied to MDPs with several goal states that give different reward amounts and that have different distances. This allows the IGE to learn policies to reach some of these goal states. However, the IGE is not restricted to this scenario. It can also learn different policies in tasks in which rewards are distributed over the MDP. This includes the maze example from the introduction, where an agent has to collect coins in a maze and find the exit. Finding the exit gives a large reward, whereas coins give smaller rewards. A module with a large γ that optimizes long-term reward will learn to collect many coins and then go to the exit, whereas for a module with stronger discounting, the higher reward of reaching the exit would be strongly discounted if it is too far in the future. For these modules the reward sum would be higher if the reward for the exit is reached as rapidly as possible. Future investigations should explore the types of tasks in which the IGE can be advantageous, and how these tasks have to be formalized, for example, in respect to reward functions that allow the IGE to learn benefical behaviors.

This work concentrated on the ability of the IGE to learn policies that represent different strategies, and that can be reused in different task contexts and objectives. However, the choice of the discount factor can also affect the learning speed and quality of policies in regular single-task settings. For example, using a smaller γ usually results in faster learning speed. François-Lavet, Fonteneau, & Ernst (2015) used this effect and increased the discount factor slowly during the learning of a deep Q-learning approach. This resulted in improved learning performance compared to using only a constant discount factor. Moreover, Jiang, Kulesza, et al. (2015) proved that if learning is based on an inaccurate model, such as is the case during on-line learning in a stochastic environment, then learning a policy to optimize a specific γ_{target} can be improved by learning it with values based on a smaller γ. The discount factor γ controls the complexity of the learnable policy class, i.e. the number of possible policies that can be learned, because γ controls the time time-scale on which optimal policies are searched. A smaller discount factor reduces the number of potential policies, because it only regards policies that can be optimal on a short time-scale. This makes the learning problem less complex and less prone to overfitting to an inaccurate model. In both cases, the choice of the optimal γ needs to be experimentally observed, by testing several options. The IGE uses different discount factors in parallel. Predicting their performance might help to determine which of the discount factors is best for a task. Future research should investigate whether the IGE might be useful to improve learning speed in regular tasks, and not only in transfer learning settings.

Relationship to Models of Human Decision-Making

Besides being a new framework for adaptive reinforcement learning, the IGE may provide further insight into learning and decision-making processes in humans and animals.

The inspiration for the IGE came from findings about inter-temporal decision-making in humans. Tanaka, Doya, et al. (2004) and Tanaka, Schweighofer, et al. (2007) found evidence for a modular structure in the brain. Different brain areas seem to encode values for choices computed with area-specific discount factors. The IGE might help to understand the function of this brain structure.

An existing model by Kurth-Nelson & Redish (2009), the Dependent γ-Ensemble (DGE), proposed that this brain structure might be responsible for hyperbolic discounting, observed during human decision-making. They showed that hyperbolic discounting can be replicated using the integral over the values of an ensemble of exponentially discounted value functions. In their model, functions depend upon each other by learning values for a single policy. The IGE differs from this model by learning individual policies for each module. This allows the IGE to adapt to different situations, by having a behavioral repertoire from which it can select the most appropriate policy for a given situation. This idea provides an alternative hypothesis about the potential function of the modular structure in the brain. Instead of being responsible for learning hyperbolic discounting, it can learn different behaviors in parallel.

The brain structure in question is the striatum. It has been linked to model-free decision-making in humans and animals (Section 1.4.2). Such model-free processes have been linked to habitual behavior (Daw, Niv, & Dayan 2005). The IGE can be understood as a framework that learns different behaviors for the same environment. In the same way, the striatum can learn several habitual behaviors in parallel. Depending on the situation, the habitual behavior that is most appropriate is then activated.

Tanaka, Schweighofer, et al. (2007) and Schweighofer et al. (2008) showed that serotonin, a neurotransmitter, control the overall behavior. In participants that had higher serotonin levels, induced by a tryptophan-rich diet, brain activities correlated with values using large γ's were increased in the dorsal striatum. Participants also showed a higher preference for long-term, high-reward choices. Whereas, in participants with reduced serotonin levels, brain activities correlated with values using small γ factors were increased in the ventral striatum, and the choice preference shifted toward short-term, low-reward choices. This suggests that serotonin controls which areas are more active and control the behavior. This is analogous to the γ-map used to select the active γ-module dependent on a context signal. Serotonin can be understood as such a context signal. It has been linked to stress and depression, varying under stressful situations. Stress is a useful indicator to switch between habits. Under low stress it makes sense to opt for long-term, high-reward choices, whereas under high stress, it may be more beneficial to seek a more readily achievable reward target. A modular system similar to the IGE would have the advantage of not needing to relearn habits if its stress level and therefore the context changes. Instead, the agent could learn a repertoire of habits useful for different stress levels.

In summary, the IGE not only provides a brain-inspired framework to adapt reinforcement learning between different task contexts and objectives. It also provides a new hypothesis about the learning of habits in the brain, in the form of an ensemble that learns habits in parallel for different stress levels.

Appendix A

Supplementary Data for the Context Adaptive Energy Foraging Task

The appendix lists additional results of the energy foraging task discussed in Section 3.4.3.

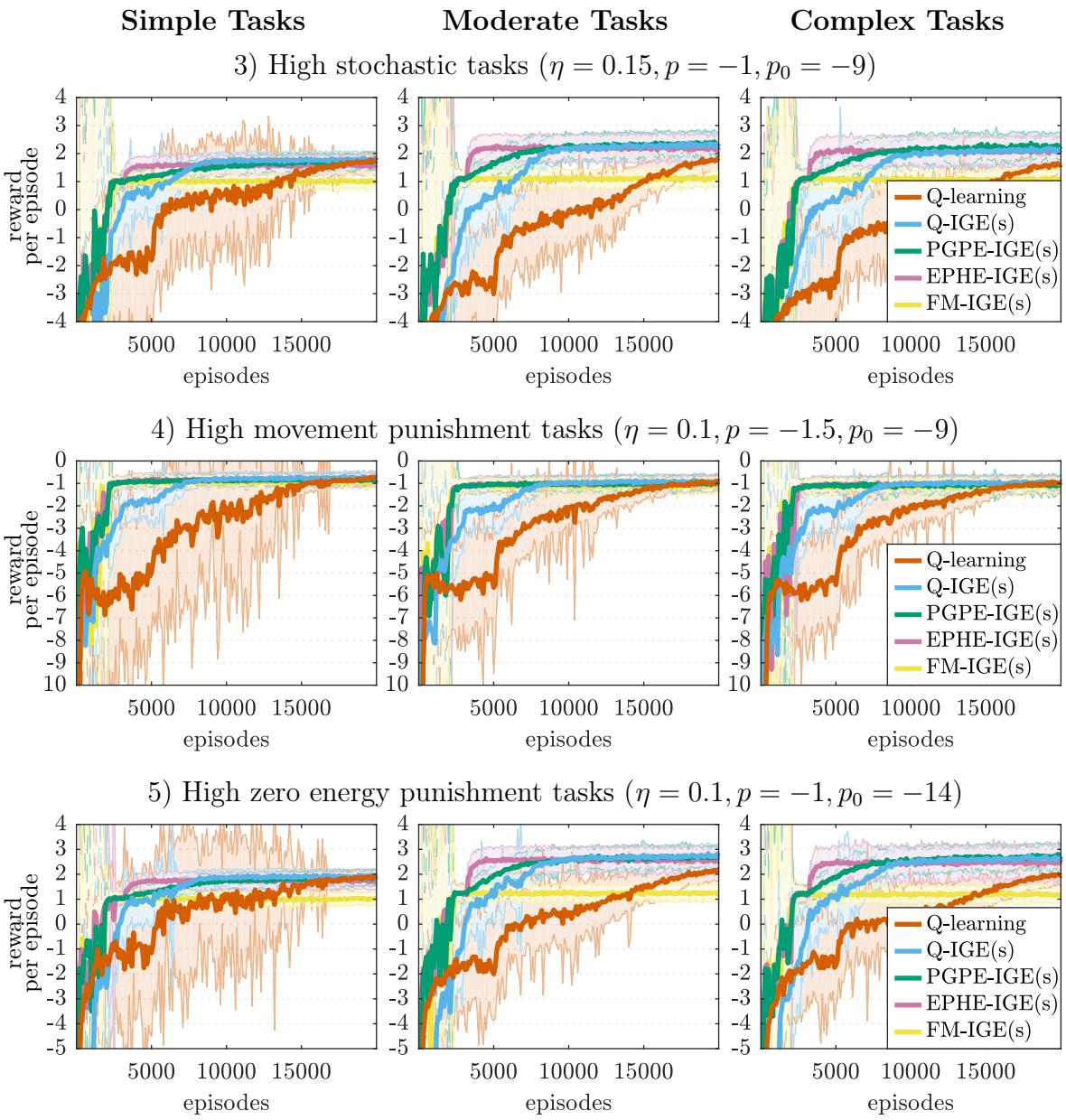


Figure A.1: The CA-IGE variants have a better learning rate than classical Q-learning. Q-learning is only in the simple, deterministic task better. The average reward per episode was measured for the greedy policy after every 100 episodes of learning. The greedy policy was evaluated for 200 episodes. Learning was stopped during the evaluation. The lines show the mean of the average over 100 independent runs. The shaded area shows the standard deviation. Only the CA-IGE variants are shown that learn their modules values under a sub-context.

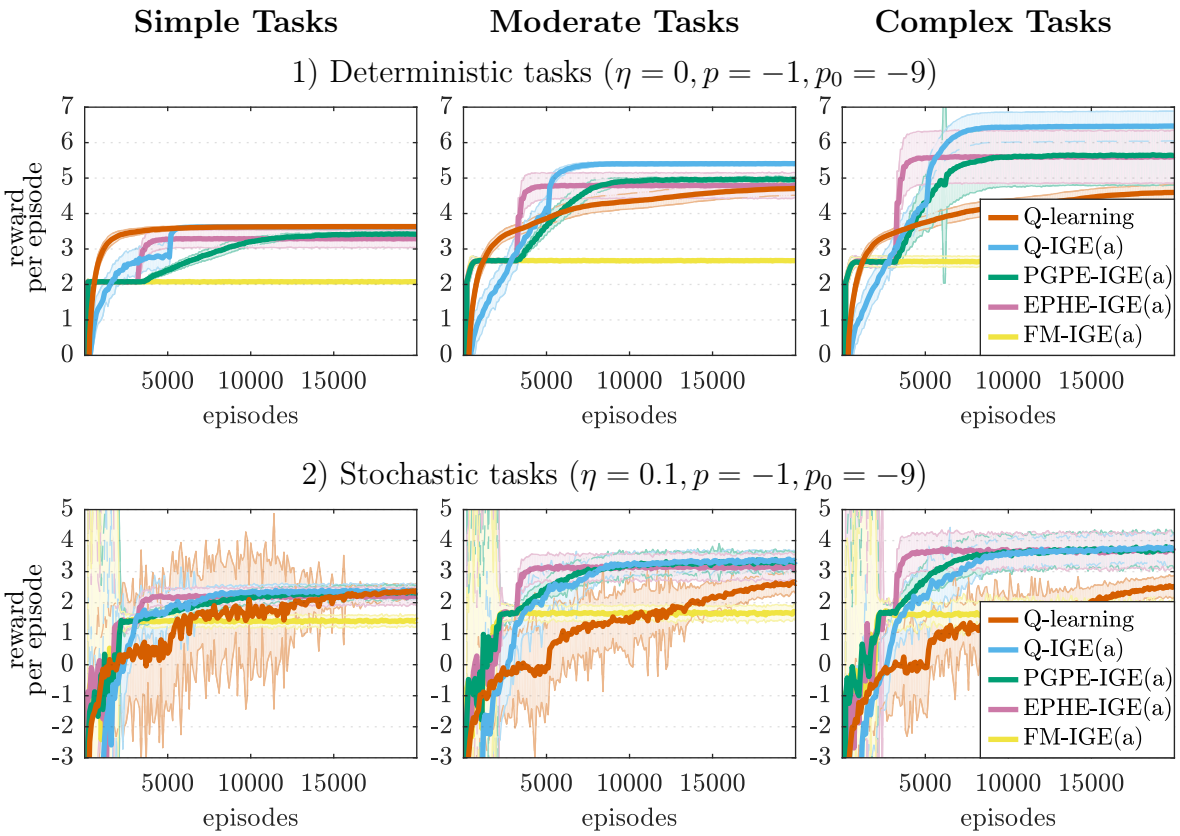


Figure A.2: The CA-IGE variants have a better learning rate than classical Q-learning. Q-learning is only in the simple, deterministic task better. The average reward per episode was measured for the greedy policy after every 100 episodes of learning. The greedy policy was evaluated for 200 episodes. Learning was stopped during the evaluation. The lines show the mean of the average over 100 independent runs. The shaded area shows the standard deviation. Only the CA-IGE variants are shown that learn their modules values under all contexts.

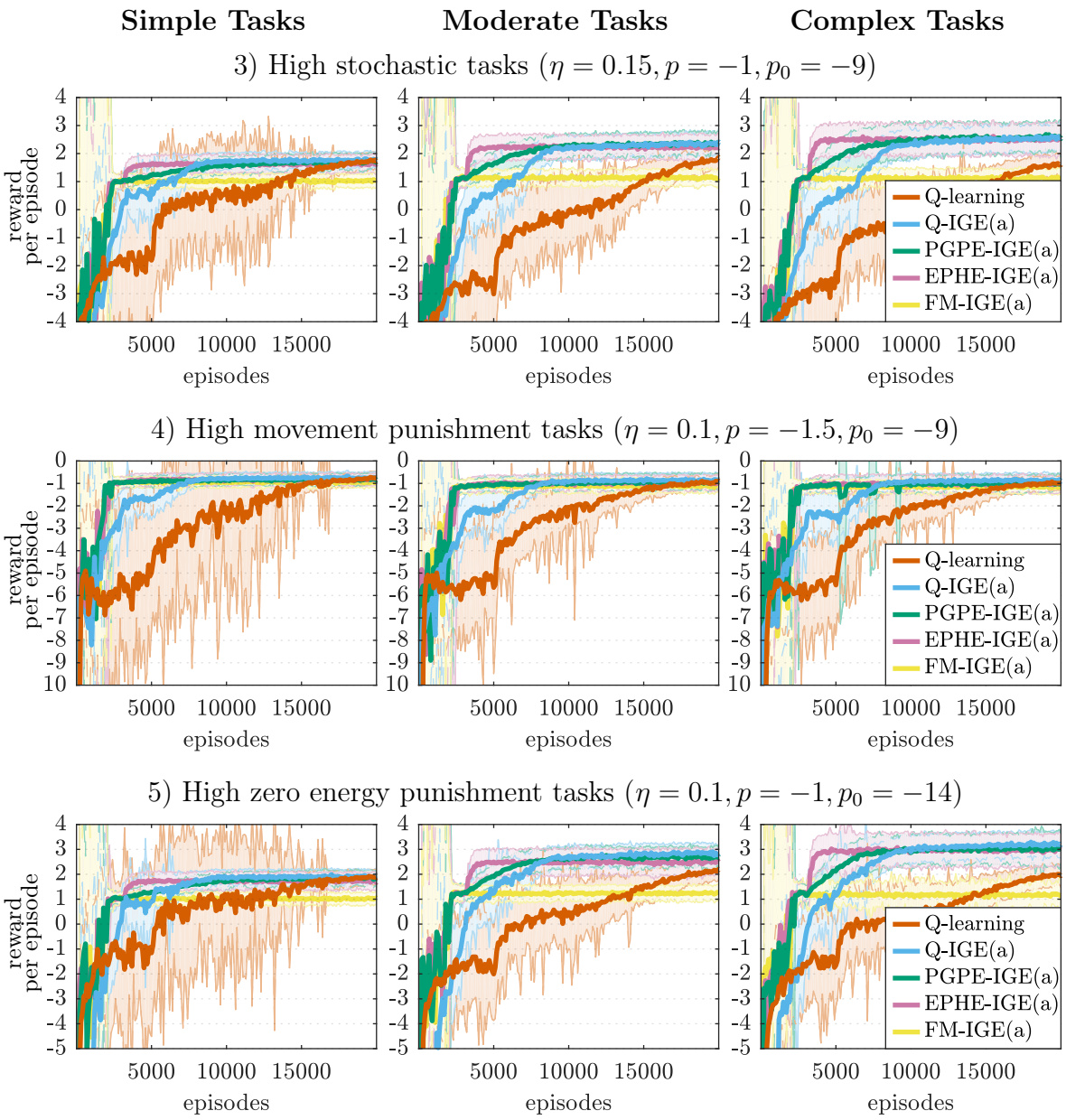


Figure A.3: The CA-IGE variants have a better learning rate than classical Q-learning. Q-learning is only in the simple, deterministic task better. The average reward per episode was measured for the greedy policy after every 100 episodes of learning. The greedy policy was evaluated for 200 episodes. Learning was stopped during the evaluation. The lines show the mean of the average over 100 independent runs. The shaded area shows the standard deviation. Only the CA-IGE variants are shown that learn their modules values under all contexts.

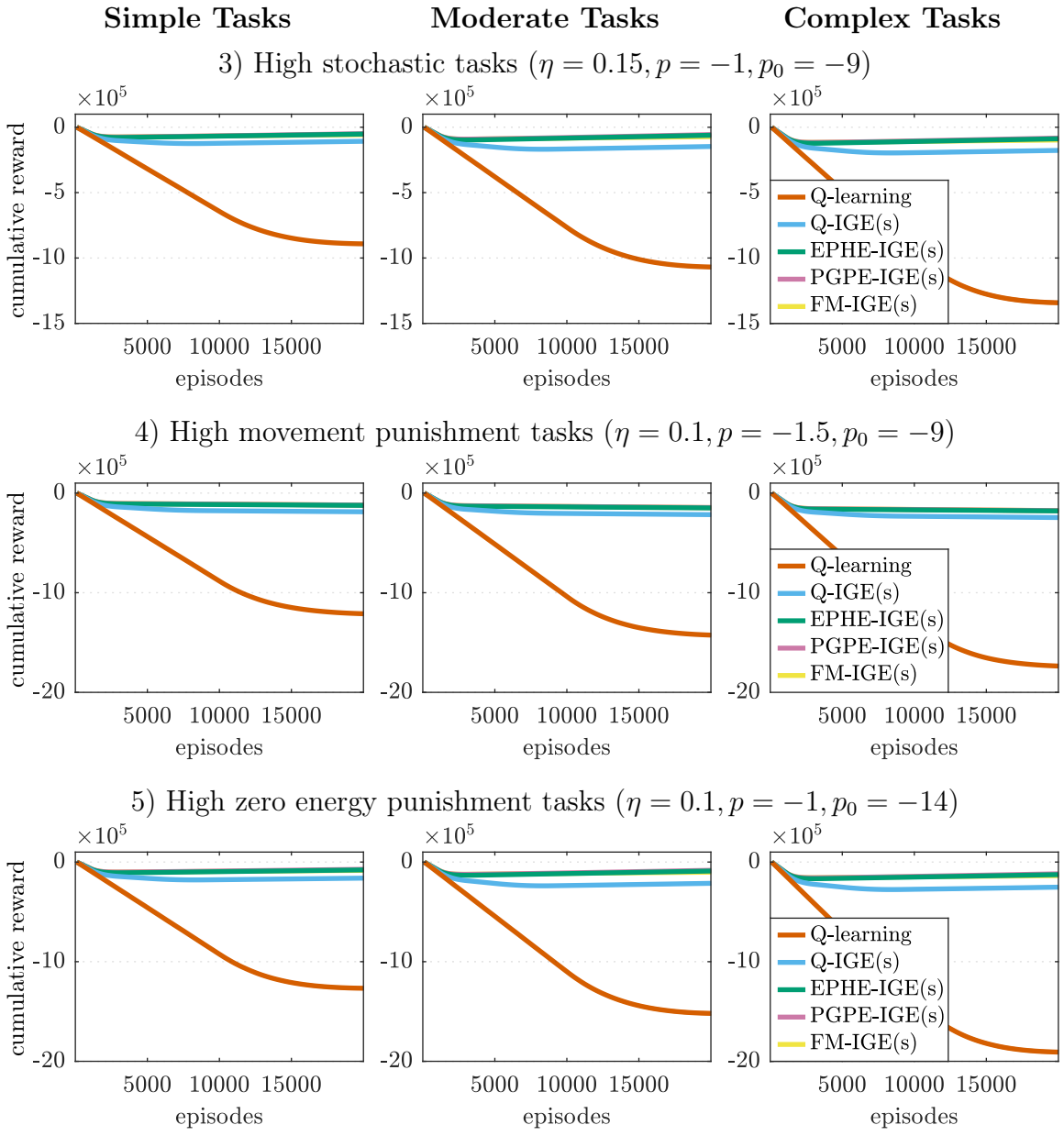


Figure A.4: The CA-IGE variants have a better cumulative reward compared to classical Q-learning for all tasks. The cumulative reward was measured over the curse of learning. The lines show the mean of the cumulative reward over 100 independent runs. The standard deviations are too small to be plotted. Only the CA-IGE variants are shown that learn their modules values under a sub-context.

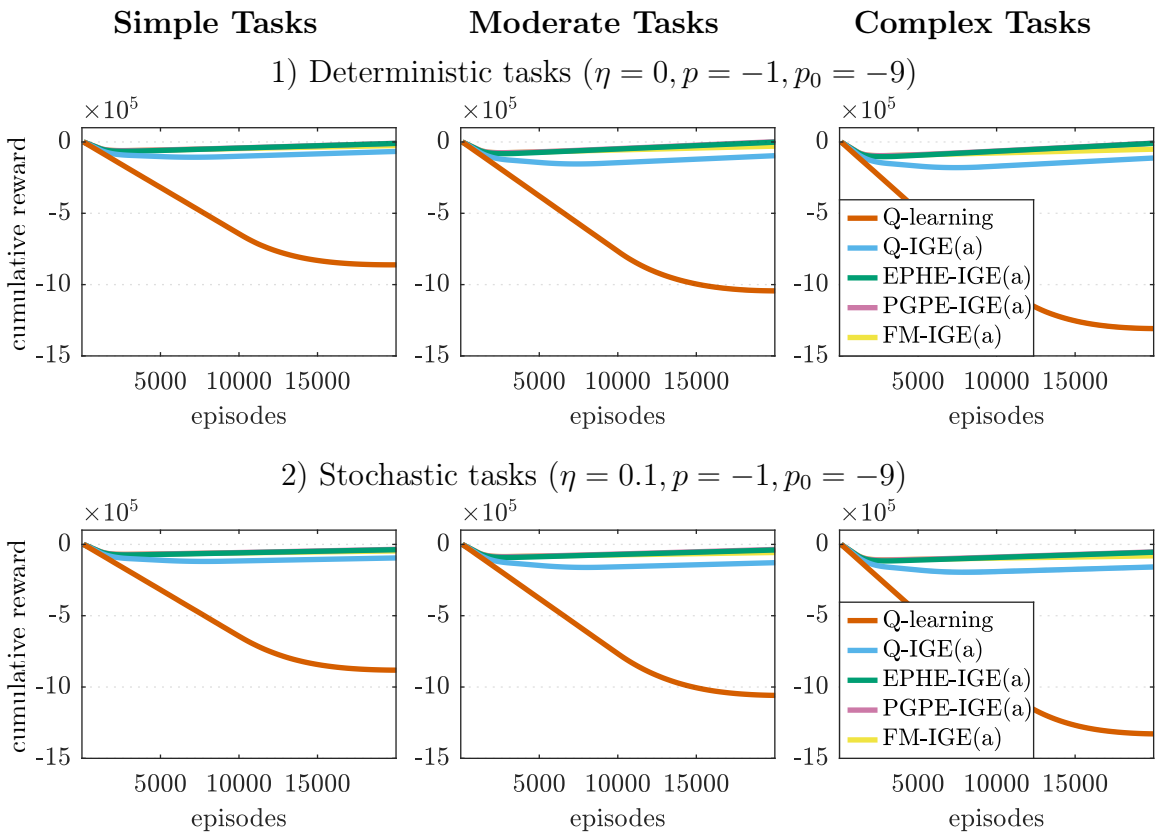


Figure A.5: The CA-IGE variants have a better cumulative reward compared to classical Q-learning for all tasks. The cumulative reward was measured over the curse of learning. The lines show the mean of the cumulative reward over 100 independent runs. The standard deviations are too small to be plotted. Only the CA-IGE variants are shown that learn their modules values under all contexts.

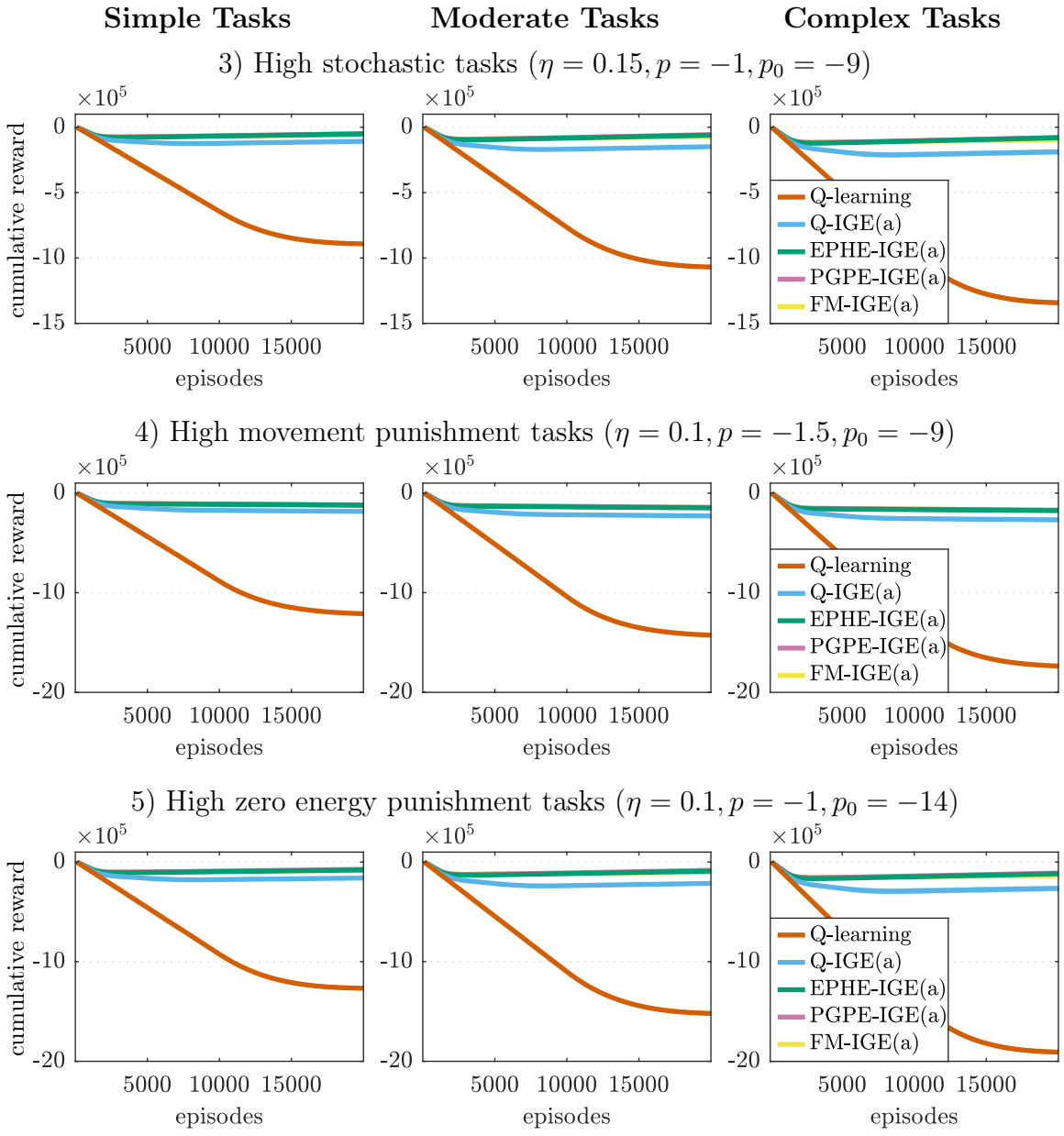


Figure A.6: The CA-IGE variants have a better cumulative reward compared to classical Q-learning for all tasks. The cumulative reward was measured over the curse of learning. The lines show the mean of the cumulative reward over 100 independent runs. The standard deviations are too small to be plotted. Only the CA-IGE variants are shown that learn their modules values under all contexts.

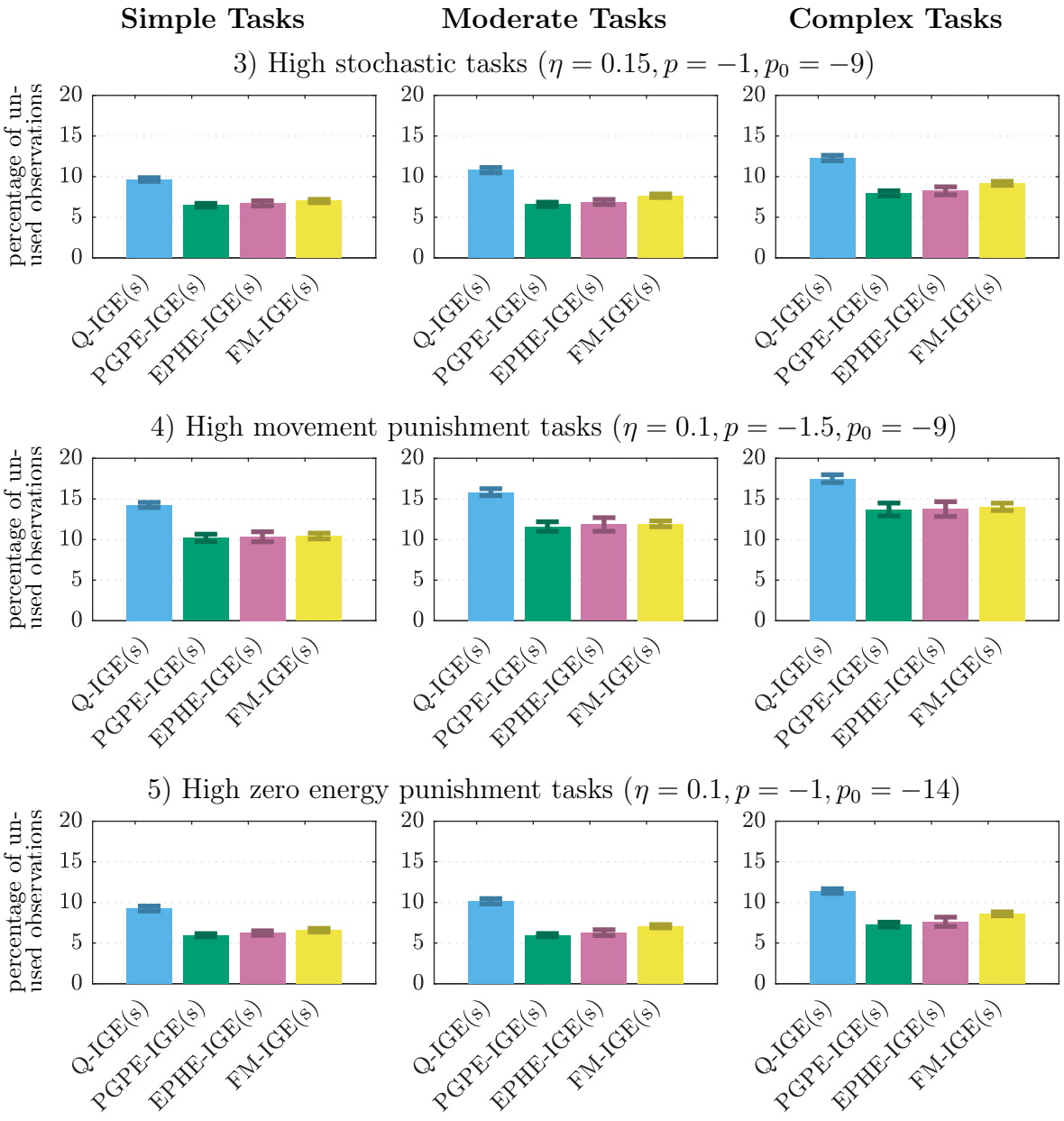


Figure A.7: The approach of learning values of γ-modules only for non-zero energy levels ($s^C > 0$) can not use between 5 to 15 percent of the observations. The bars show the mean percentage of observations that could be used to update the modules over 100 independent runs. The error bar shows the standard deviation.

Appendix B

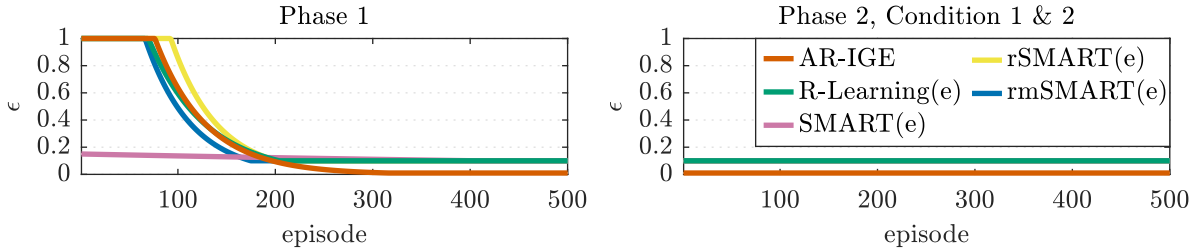
Supplementary Data for the Average Reward Experiments

The appendix lists further results for the experimental comparison between the AR-IGE and existing average-adjusted algorithms (Section 5.5.2). The results are for the episodic variants of the average-adjusted algorithms for decision-tree tasks, and the continuous variants for grid-world tasks.

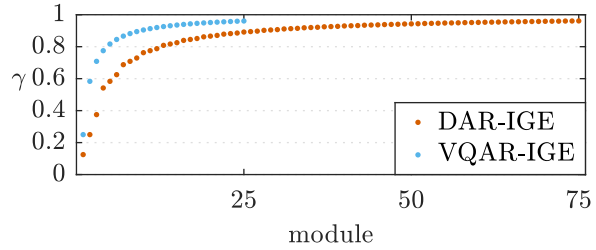
Figs. B.1 and B.3 show the learning parameters used for the experiments. The results are shown in Figs. B.2 and B.4.

Learning Parameters for Decision-Tree Tasks

a) Exploration rates



b) AR-IGE Γ-sets



c) Remaining parameters

Algorithm	**Parameters**
All	$\alpha_Q = 1$
AR-IGE	$\epsilon_\gamma = 0.1$
R-Learning(e)	$\alpha_\rho = 0.001$
rSMART(e)	$\alpha_\rho = 0.2$
rmSMART(e)	$\alpha_\rho = 0.001$

Figure B.1: The learning parameters for decision-tree tasks for both AR-IGE variants and the episodic variants of the average-adjusted algorithms.

Results for Episodic Average-Adjusted Algorithms in Decision-Tree Tasks

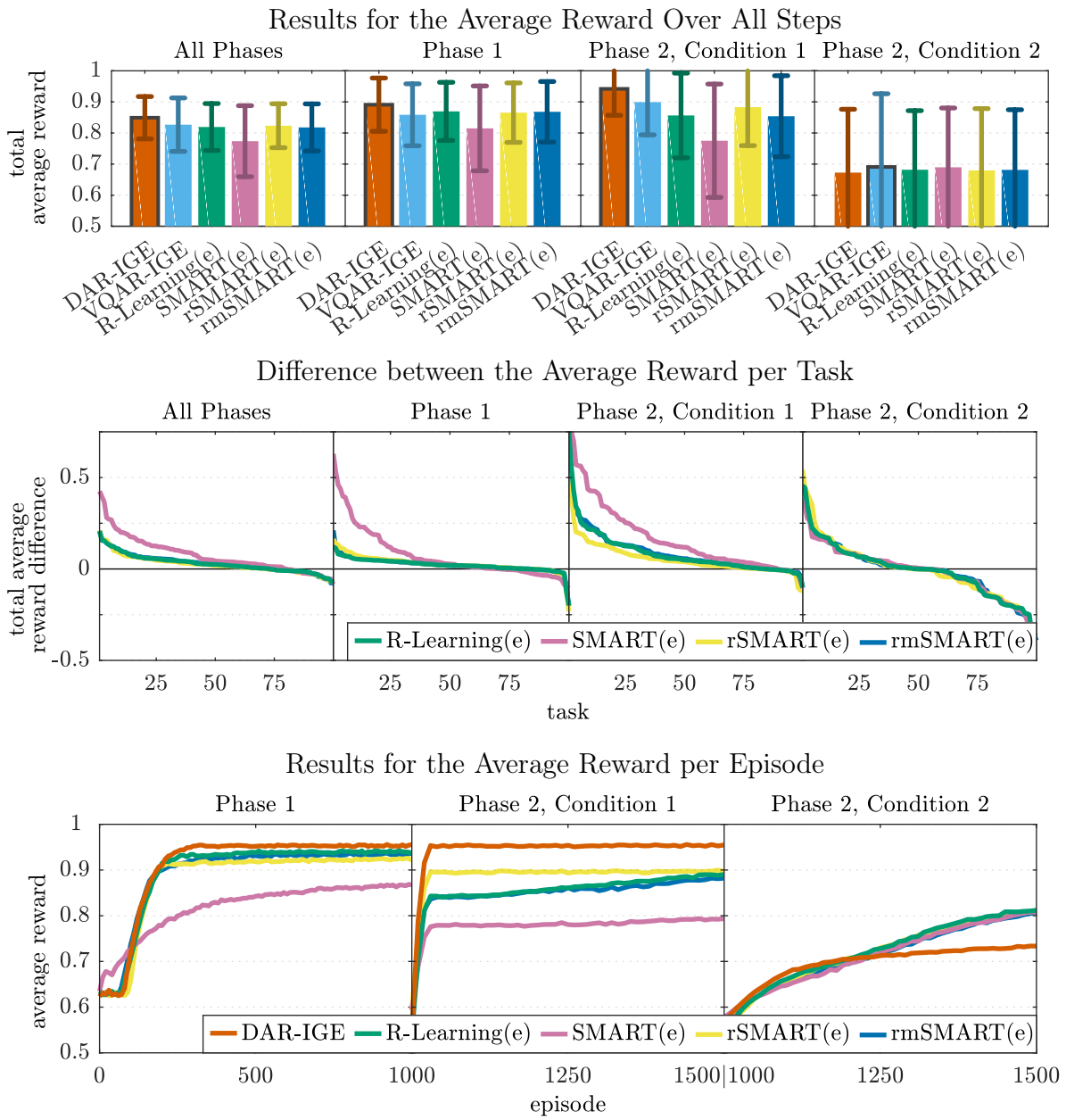
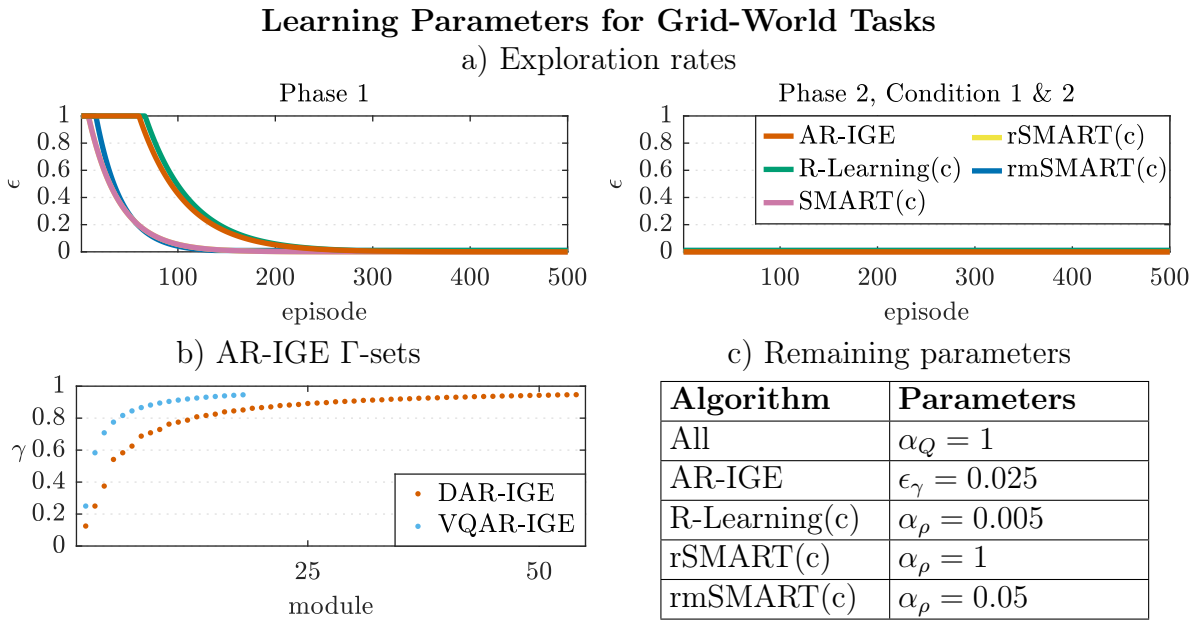


Figure B.2: **(top)** Average reward over all steps of all phases and per individual phase. The bar shows the mean over the 100 tasks and the error-bar the standard deviation. **(middle)** The difference between the average reward per task between the DAR-IGE and the average-adjusted algorithms. **(bottom)** The average reward per episode during the learning. The DAR-IGE could outperform all episodic average-adjusted algorithms with exception of Condition 2 of Phase 2.



Algorithm	**Parameters**
All	$\alpha_Q = 1$
AR-IGE	$\epsilon_\gamma = 0.025$
R-Learning(c)	$\alpha_\rho = 0.005$
rSMART(c)	$\alpha_\rho = 1$
rmSMART(c)	$\alpha_\rho = 0.05$

Figure B.3: The learning parameters for grid-world tasks for both AR-IGE variants and the continuous variants of the average-adjusted algorithms.

Results for Continuous Average-Adjusted Algorithms in Grid-World Tasks

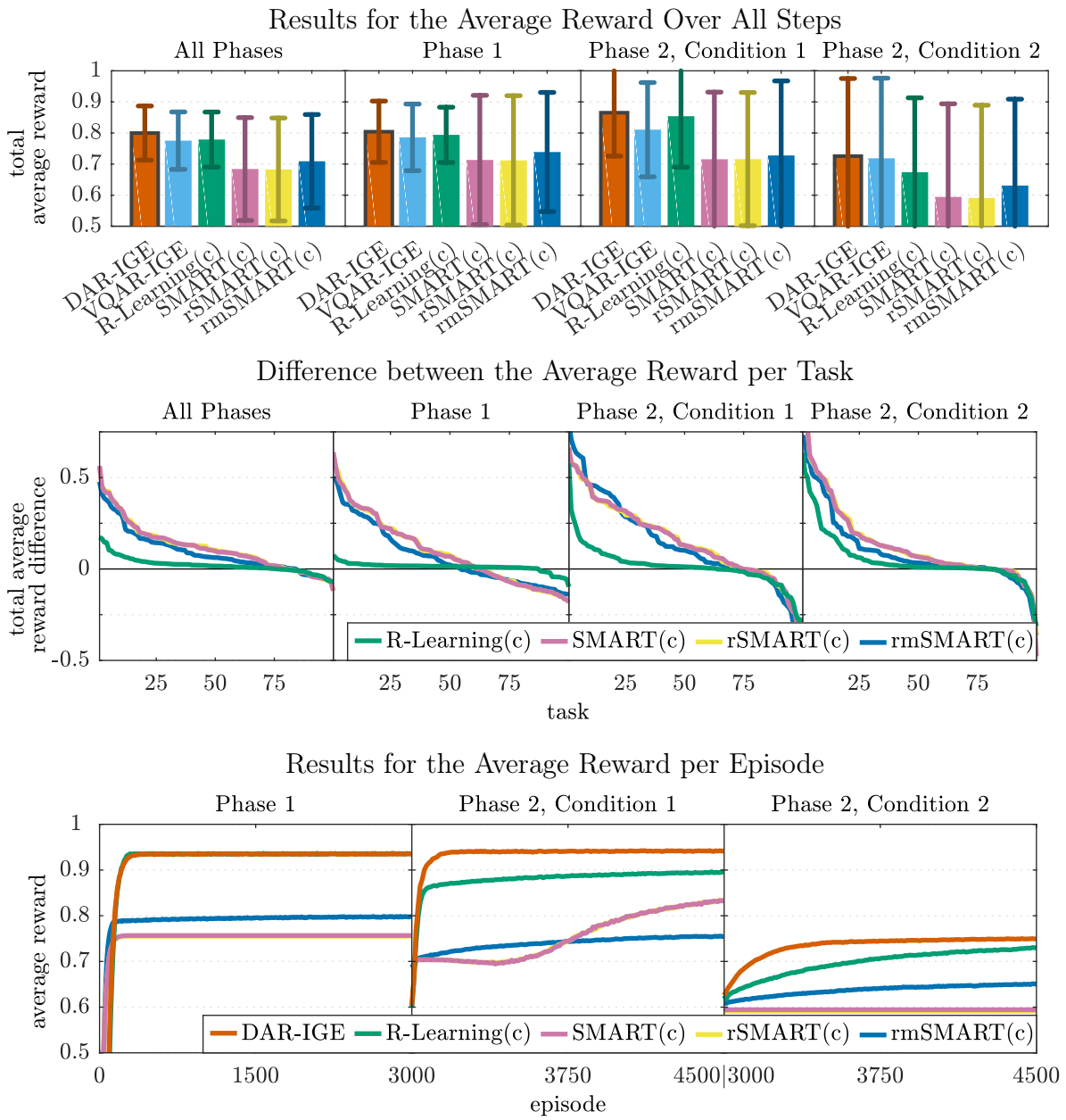


Figure B.4: **(top)** Average reward over all steps of all phases and per individual phase. The bar shows the mean over the 100 tasks and the error-bar the standard deviation. **(middle)** The difference between the average reward per task between the DAR-IGE and the average-adjusted algorithms. **(bottom)** The average reward per episode during the learning. The DAR-IGE could consistently outperform all continuous average-adjusted algorithms.

Appendix C

Analytical Solution to Theorem 21

To determine the lower and upper bound ($\gamma_L(\kappa)$, $\gamma_U(\kappa)$) of the γ-region $\Gamma(\kappa, n^*)$ that guarantees to learn the optimal average reward solution in deterministic, goal-only-reward MDPs according to Theorem 21, several combinations for the number of steps of the optimal choice n^* and the number of steps of any other choices n need to be tested. An analytical solution is given by Theorem 22, but the proof of the theorem is at the moment incomplete.

Theorem 22. *For any deterministic, goal-only-reward MDP exists a γ-region $\Gamma^*(\kappa, n^*)$ for which all its discount factors γ^* are guaranteed to encode the optimal average reward choice $c^* = (r^*, n^*)$ if its average reward is at most by a factor κ smaller than the average reward of any other choice $c = (r, n)$:*

$$\kappa \frac{r^*}{n^*} \geq \frac{r}{n} \ ,$$

with $\kappa \in [0\ 1]$. The region is defined by:

$$\Gamma^*(\kappa, n^*) = \{\gamma_L(\kappa) < \gamma^* < \gamma_U(\kappa)\} \ ,$$

where $\gamma_L(\kappa)$ and $\gamma_U(\kappa)$ are the lower and upper border of the region. The lower border is defined by:

$$\gamma_L(\kappa) = \max\left(\left(\kappa \frac{n'_L}{n^*}\right)^{\frac{1}{n^* - n'_L}}, \left(\kappa \frac{n''_L}{n^*}\right)^{\frac{1}{n^* - n''_L}}\right) \ ,$$

with

$$n'_L = \min\left(\lfloor n_L \rfloor, n^* - 1\right) \ \textit{and}\ n''_L = \min\left(\lceil n_L \rceil, n^* - 1\right) \ ,$$

and

$$n_L = -n^* \frac{1}{W_{-1}\left(-\frac{\kappa}{e}\right)} \ ,$$

where W is the Lambert-W, or product log function.

The upper border of the γ-region $\Gamma^*(\kappa, n^*)$ is defined by:

$$\gamma_U(\kappa) = \min\left(\left(\frac{n^*}{\kappa \cdot n'_U}\right)^{\frac{1}{n'_U - n^*}}, \left(\frac{n^*}{\kappa \cdot n''_U}\right)^{\frac{1}{n''_U - n^*}}\right) \ ,$$

with

$$n'_U = \max\left(\lfloor n_U \rfloor, n^* + 1\right) \;\; \textit{and} \; n''_U = \max\left(\lceil n_U \rceil, n^* + 1\right) \;\; ,$$

and

$$n_U = -n^* \frac{1}{W_0\left(-\frac{\kappa}{e}\right)} \;\; .$$

Bibliography

Adamantidis, Antoine R et al. (2011). "Optogenetic interrogation of dopaminergic modulation of the multiple phases of reward-seeking behavior". In: *The Journal of Neuroscience* 31.30, pp. 10829–10835.

Andreas, Jacob, Dan Klein, & Sergey Levine (2016). *Modular multitask reinforcement learning with policy sketches*. arXiv:1611.01796.

Asadi, Mehran & Manfred Huber (2007). "Effective Control Knowledge Transfer through Learning Skill and Representation Hierarchies." In: *Proceedings of the Twentieth International Joint Conference on Artificial Intelligence*, pp. 2054–2059.

Bacon, Pierre-Luc, Jean Harb, & Doina Precup (2017). "The Option-Critic Architecture". In: *Proceedings of the Thirthy-First AAAI Conference On Artificial Intelligence*, pp. 1726–1734.

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

Barto, Andrew G, Satinder Singh, & Nuttapong Chentanez (2004). "Intrinsically motivated learning of hierarchical collections of skills". In: *Proceedings of the 3rd International Conference on Development and Learning*, pp. 112–19.

Bellman, R.E. (1957). *Dynamic programming*. Princeton University Press.

Bernstein, Daniel S (1999). *Reusing old policies to accelerate learning on new MDPs*. Tech. rep. Department of Computer Science, University of Massachusetts at Amherst.

Bishop, Christopher M. (2006). *Pattern Recognition and Machine Learning*. Springer-Verlag New York, Inc.

Bonarini, Andrea, Alessandro Lazaric, & Marcello Restelli (2006). "Incremental skill acquisition for self-motivated learning animats". In: *Proceedings of the 9th International Conference on From Animals to Animats: Simulation of Adaptive Behavior*. Springer, pp. 357–368.

Brunskill, Emma & Lihong Li (2014). "Pac-inspired option discovery in lifelong reinforcement learning". In: *Proceedings of the 31st International Conference on Machine Learning*, pp. 316–324.

Da Silva, Bruno, George Konidaris, & Andrew Barto (2012). *Learning parameterized skills*. arXiv:1206.6398.

Daniel, Christian et al. (2016). "Probabilistic inference for determining options in reinforcement learning". In: *Machine Learning* 104.2-3, pp. 337–357.

Das, Tapas K et al. (1999). "Solving semi-Markov decision problems using average reward reinforcement learning". In: *Management Science* 45.4, pp. 560–574.

Daw, Nathaniel D, Yael Niv, & Peter Dayan (2005). "Uncertainty-based competition between prefrontal and dorsolateral striatal systems for behavioral control". In: *Nature Neuroscience* 8.12, pp. 1704–1711.

Deisenroth, Marc, Gerhard Neumann, & Jan Peters (2011). "A Survey on Policy Search for Robotics". In: *Foundations and Trends in Robotics* 2.1-2, pp. 1–142.

Deisenroth, Marc & Carl E Rasmussen (2011). "PILCO: A model-based and data-efficient approach to policy search". In: *Proceedings of the 28th International Conference on Machine Learning*, pp. 465–472.

Devin, Coline et al. (2017). "Learning modular neural network policies for multi-task and multi-robot transfer". In: *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 2169–2176.

Digney, Bruce L (1998). "Learning hierarchical control structures for multiple tasks and changing environments". In: *From Animals to Animats 5: Proceedings of the Fifth International Conference on Simulation of Adaptive Behavior*, pp. 321–330.

Doya, Kenji (1999). "What are the computations of the cerebellum, the basal ganglia and the cerebral cortex?" In: *Neural Networks* 12.7, pp. 961–974.

— (2002). "Metalearning and neuromodulation". In: *Neural Networks* 15.4-6, pp. 495–506.

Ferguson, Kimberly & Sridhar Mahadevan (2006). "Proto-transfer learning in markov decision processes using spectral methods". In: *Workshop on Structural Knowledge Transfer for Machine Learning at the Twenty-Third International Conference on Machine Learning*.

Fernández, Fernando, Javier García, & Manuela Veloso (2010). "Probabilistic policy reuse for inter-task transfer learning". In: *Robotics and Autonomous Systems* 58.7, pp. 866–871.

Fernández, Fernando & Manuela Veloso (2006). "Policy reuse for transfer learning across tasks with different state and action spaces". In: *Workshop on Structural Knowledge Transfer for Machine Learning at the Twenty-Third International Conference on Machine Learning*.

Ferns, Norm, Prakash Panangaden, & Doina Precup (2004). "Metrics for finite Markov decision processes". In: *Proceedings of the 20th conference on Uncertainty in artificial intelligence*, pp. 162–169.

François-Lavet, Vincent, Raphael Fonteneau, & Damien Ernst (2015). *How to discount deep reinforcement learning: Towards new dynamic strategies*. arXiv:1512.02011.

Frederick, Shane, George Loewenstein, & Ted O'Donoghue (2002). "Time discounting and time preference: A critical review". In: *Journal of Economic Literature* 40.2, pp. 351–401.

Gautschi, Walter (1990). "How (un) stable are Vandermonde systems". In: *Asymptotic and Computational Analysis*. Vol. 124. Lecture Notes in Pure and Applied Mathematics, pp. 193–210.

Gläscher, Jan et al. (2010). "States versus rewards: dissociable neural prediction error signals underlying model-based and model-free reinforcement learning". In: *Neuron* 66.4, pp. 585–595.

Goodfellow, Ian, Yoshua Bengio, & Aaron Courville (2016). *Deep Learning*. MIT Press.

Gosavi, Abhijit (2004). "Reinforcement learning for long-run average cost". In: *European Journal of Operational Research* 155.3, pp. 654–674.

Hallak, Assaf, Dotan Di Castro, & Shie Mannor (2015). *Contextual Markov decision processes*. arXiv:1502.02259.

Hawasly, Majd & Subramanian Ramamoorthy (2013). "Lifelong transfer learning with an option hierarchy". In: *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 1341–1346.

Hengst, Bernhard (2002). "Discovering hierarchy in reinforcement learning with HEXQ". In: *Proceedings of the Nineteenth International Conference on Machine Learning*. Vol. 2, pp. 243–250.

— (2003). "Discovering hierarchy in reinforcement learning". PhD thesis.

Higgins, Irina et al. (2017). *Darla: Improving zero-shot transfer in reinforcement learning*. arXiv:1707.08475.

Isele, David, Mohammad Rostami, & Eric Eaton (2016). "Using Task Features for Zero-Shot Knowledge Transfer in Lifelong Learning". In: *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence*, pp. 1620–1626.

Ito, Makoto & Kenji Doya (2009). "Validation of decision-making models and analysis of decision variables in the rat basal ganglia". In: *The Journal of Neuroscience* 29.31, pp. 9861–9874.

Jiang, Nan, Akshay Krishnamurthy, et al. (2017). "Contextual Decision Processes with low Bellman rank are PAC-Learnable". In: *Proceedings of the 34th International Conference on Machine Learning*, pp. 1704–1713.

Jiang, Nan, Alex Kulesza, et al. (2015). "The dependence of effective planning horizon on model accuracy". In: *Proceedings of the 2015 International Conference on Autonomous Agents and Multiagent Systems*, pp. 1181–1189.

Jonsson, Anders & Andrew Barto (2006). "Causal graph based decomposition of factored mdps". In: *Journal of Machine Learning Research* 7, pp. 2259–2301.

Kable, Joseph W & Paul W Glimcher (2007). "The neural correlates of subjective value during intertemporal choice". In: *Nature Neuroscience* 10.12, pp. 1625–1633.

Kakade, Sham (2001). "A Natural Policy Gradient." In: *Advances in Neural Information Processing Systems 14*, pp. 1531–1538.

Konidaris, George & Andrew G Barto (2009). "Skill discovery in continuous reinforcement learning domains using skill chaining". In: *Advances in Neural Information Processing Systems 22*, pp. 1015–1023.

Konidaris, George, Scott Kuindersma, et al. (2010). "Constructing skill trees for reinforcement learning agents from demonstration trajectories". In: *Advances in Neural Information Processing Systems 23*, pp. 1162–1170.

Kurth-Nelson, Zeb & A David Redish (2009). "Temporal-difference reinforcement learning with distributed representations". In: *PloS One* 4.10, e7362.

— (2010). "A reinforcement learning model of precommitment in decision making". In: *Frontiers in Behavioral Neuroscience* 4.

Lakshminarayanan, Aravind S et al. (2016). *Option Discovery in Hierarchical Reinforcement Learning using Spatio-Temporal Clustering*. arXiv:1605.05359.

Langford, John & Tong Zhang (2008). "The epoch-greedy algorithm for multi-armed bandits with side information". In: *Advances in Neural Information Processing Systems 21*, pp. 817–824.

Lazaric, Alessandro (2012). "Transfer in Reinforcement Learning: A Framework and a Survey". In: *Reinforcement Learning*. Vol. 12. Adaptation, Learning, and Optimization. Springer, pp. 143–173.

Lazaric, Alessandro, Marcello Restelli, & Andrea Bonarini (2008). "Transfer of samples in batch reinforcement learning". In: *Proceedings of the 25th international conference on Machine learning*, pp. 544–551.

Liu, Chunming, Xin Xu, & Dewen Hu (2015). "Multiobjective reinforcement learning: A comprehensive overview". In: *IEEE Transactions on Systems, Man, and Cybernetics: Systems* 45.3, pp. 385–398.

Machado, Marlos C., Marc G. Bellemare, & Michael H. Bowling (2017). *A Laplacian Framework for Option Discovery in Reinforcement Learning*. arXiv:1703.00956.

Macon, Nathaniel & Abraham Spitzbart (1958). "Inverses of Vandermonde matrices". In: *The American Mathematical Monthly* 65.2, pp. 95–100.

Maei, Hamid Reza & Richard S Sutton (2010). "GQ (λ): A general gradient algorithm for temporal-difference prediction learning with eligibility traces". In: *Proceedings of the Third Conference on Artificial General Intelligence*, pp. 91–96.

Mahadevan, Sridhar (1996). "Average reward reinforcement learning: Foundations, algorithms, and empirical results". In: *Machine Learning* 22.1-3, pp. 159–195.

— (2005). "Proto-value functions: Developmental reinforcement learning". In: *Proceedings of the 22nd international conference on Machine learning*, pp. 553–560.

Mahadevan, Sridhar & Mauro Maggioni (2007). "Proto-value functions: A Laplacian framework for learning representation and control in Markov decision processes". In: *Journal of Machine Learning Research* 8, pp. 2169–2231.

Mahadevan, Sridhar, Nicholas Marchalleck, et al. (1997). "Self-improving factory simulation using continuous-time average-reward reinforcement learning". In: *Proceedings of the 14th International Conference on Machine Learning*, pp. 202–210.

Mahmud, MM et al. (2013). *Clustering markov decision processes for continual transfer*. arXiv:1311.3959.

Mankowitz, Daniel J, Timothy A Mann, & Shie Mannor (2016). "Adaptive Skills Adaptive Partitions (ASAP)". In: *Advances in Neural Information Processing Systems 29*, pp. 1588–1596.

Mannor, Shie et al. (2004). "Dynamic abstraction in reinforcement learning via clustering". In: *Proceedings of the twenty-first international conference on Machine learning*, p. 71.

McGovern, Amy & Andrew G Barto (2001). "Automatic discovery of subgoals in reinforcement learning using diverse density". In: *Proceedings of the Eighteenth International Conference on Machine Learning*, pp. 361–368.

Mehta, Neville et al. (2008). "Transfer in variable-reward hierarchical reinforcement learning". In: *Machine Learning* 73.3, pp. 289–312.

Menache, Ishai, Shie Mannor, Nahum Shimkin, et al. (2002). "Q-cut-dynamic discovery of sub-goals in reinforcement learning". In: *Machine Learning: ECML 2002*. Vol. 2430. Lecture Notes in Computer Science. Springer, pp. 295–306.

Mnih, Volodymyr et al. (2015). "Human-level control through deep reinforcement learning". In: *Nature* 518.7540, p. 529.

Modayil, Joseph, Adam White, & Richard S Sutton (2014). "Multi-timescale nexting in a reinforcement learning robot". In: *Adaptive Behavior* 22.2, pp. 146–160.

Niv, Yael (2009). "Reinforcement learning in the brain". In: *Journal of Mathematical Psychology* 53.3, pp. 139–154.

Oh, Junhyuk et al. (2017). *Zero-shot task generalization with multi-task deep reinforcement learning*. arXiv:1706.05064.

Osband, Ian & Benjamin Van Roy (2017). *On Optimistic versus Randomized Exploration in Reinforcement Learning*. arXiv:1706.04241.

Pagnoni, Giuseppe et al. (2002). "Activity in human ventral striatum locked to errors of reward prediction". In: *Nature Neuroscience* 5.2, pp. 97–98.

Pavlov, Ivan Petrovich (1927). *Conditional reflexes: An investigation of the physiological activity of the cerebral cortex*. Oxford Univ. Press.

Perkins, Theodore J & Doina Precup (1999). *Using options for knowledge transfer in reinforcement learning*. Tech. rep. University of Massachusetts, Amherst, MA, USA.

Pessiglione, Mathias et al. (2006). "Dopamine-dependent prediction errors underpin reward-seeking behaviour in humans". In: *Nature* 442.7106, pp. 1042–1045.

Peters, Jan & Stefan Schaal (2007). "Reinforcement learning by reward-weighted regression for operational space control". In: *Proceedings of the 24th international conference on Machine learning*, pp. 745–750.

Phillips, Caitlin (2006). *Knowledge transfer in Markov decision processes*. Tech. rep. McGill University, School of Computer Science.

Pickett, Marc & Andrew G Barto (2002). "Policyblocks: An algorithm for creating useful macro-actions in reinforcement learning". In: *Proceedings of the Nineteenth International Conference on Machine Learning*, pp. 506–513.

Precup, Doina, Richard S Sutton, & Satinder P Singh (2000). "Eligibility Traces for Off-Policy Policy Evaluation". In: *Proceedings of the Seventeenth International Conference on Machine Learning*, pp. 759–766.

Press, William H et al. (2007). *Numerical Recipes 3rd Edition: The Art of Scientific Computing*. Cambridge University Press.

Puterman, Martin L. (1994). *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. John Wiley & Sons, Inc.

Reinke, Chris, Eiji Uchibe, & Kenji Doya (2015). "Maximizing the average reward in episodic reinforcement learning tasks". In: *2015 International Conference on Intelligent Informatics and Biomedical Sciences (ICIIBMS)*, pp. 420–421.

— (2017). "Average Reward Optimization with Multiple Discounting Reinforcement Learners". In: *Neural Information Processing - 24th International Conference, ICONIP*. Vol. 10634. Lecture Notes in Computer Science. Springer, pp. 789–800.

Rescorla, Robert A, Allan R Wagner, et al. (1972). "A theory of Pavlovian conditioning: Variations in the effectiveness of reinforcement and nonreinforcement". In: *Classical Conditioning II: Current Theory and Research*, pp. 64–99.

Reynolds, John NJ & Jeffery R Wickens (2002). "Dopamine-dependent plasticity of corticostriatal synapses". In: *Neural Networks* 15.4, pp. 507–521.

Robbins, Herbert & Sutton Monro (1951). "A stochastic approximation method". In: *The Annals of Mathematical Statistics* 22.3, pp. 400–407.

Roijers, Diederik M et al. (2013). "A survey of multi-objective sequential decision-making". In: *Journal of Artificial Intelligence Research* 48, pp. 67–113.

Rubinstein, Reuven Y & Dirk P Kroese (2016). *Simulation and the Monte Carlo method*. Vol. 10. John Wiley & Sons, Inc.

Samejima, Kazuyuki et al. (2005). "Representation of action-specific reward values in the striatum". In: *Science* 310.5752, pp. 1337–1340.

Schaul, Tom et al. (2015). "Universal value function approximators". In: *Proceedings of the 32nd International Conference on Machine Learning*, pp. 1312–1320.

Schönberg, Tom et al. (2007). "Reinforcement learning signals in the human striatum distinguish learners from nonlearners during reward-based decision making". In: *The Journal of Neuroscience* 27.47, pp. 12860–12867.

Schultz, Wolfram, Peter Dayan, & P Read Montague (1997). "A neural substrate of prediction and reward". In: *Science* 275.5306, pp. 1593–1599.

Schwartz, Anton (1993). "A reinforcement learning method for maximizing undiscounted rewards". In: *Proceedings of the Tenth International Conference on Machine Learning*, pp. 298–305.

Schweighofer, Nicolas et al. (2008). "Low-serotonin levels increase delayed reward discounting in humans". In: *The Journal of Neuroscience* 28.17, pp. 4528–4532.

Sehnke, Frank et al. (2010). "Parameter-exploring policy gradients". In: *Neural Networks* 23.4, pp. 551–559.

Shelton, Christian R (2001). "Balancing multiple sources of reward in reinforcement learning". In: *Advances in Neural Information Processing Systems*, pp. 1082–1088.

Silver, David & Kamil Ciosek (2012). *Compositional planning using optimal option models*. arXiv:1206.6473.

Silver, David, Thomas Hubert, et al. (2017). *Mastering Chess and Shogi by Self-Play with a General Reinforcement Learning Algorithm*. arXiv:1712.01815.

Silver, David, Julian Schrittwieser, et al. (2017). "Mastering the game of go without human knowledge". In: *Nature* 550.7676, pp. 354–359.

Şimşek, Özgür & Andrew G Barto (2004). "Using relative novelty to identify useful temporal abstractions in reinforcement learning". In: *Proceedings of the twenty-first international conference on Machine learning*, p. 95.

— (2009). "Skill characterization based on betweenness". In: *Advances in Neural Information Processing Systems 22*, pp. 1497–1504.

Şimşek, Özgür, Alicia P Wolfe, & Andrew G Barto (2005). "Identifying useful subgoals in reinforcement learning by local graph partitioning". In: *Proceedings of the 22nd international conference on Machine learning*, pp. 816–823.

Singh, Satinder (1992). "Transfer of learning by composing solutions of elemental sequential tasks". In: *Machine Learning* 8.3-4, pp. 323–339.

Singh, Satinder, Andrew G Barto, & Nuttapong Chentanez (2005). "Intrinsically motivated reinforcement learning". In: *Advances in Neural Information Processing Systems 18*, pp. 1281–1288.

Skinner, B Frederic (1935). "Two types of conditioned reflex and a pseudo type". In: *The Journal of General Psychology* 12.1, pp. 66–77.

Sorg, Jonathan & Satinder Singh (2010). "Linear options". In: *Proceedings of the 9th International Conference on Autonomous Agents and Multiagent Systems*. Vol. 1, pp. 31–38.

Stolle, Martin & Doina Precup (2002). "Learning options in reinforcement learning". In: *Proceedings of the 5th International Symposium on Abstraction, Reformulation and Approximation*, pp. 212–223.

Sutton, Richard S & Andrew G Barto (1990). "Time-derivative models of Pavlovian reinforcement". In: *Learning and Computational Neuroscience: Foundations of Adaptive Networks*. MIT Press, pp. 497–537.

— (1998). *Reinforcement Learning: An Introduction*. MIT Press, Cambridge, MA.

Sutton, Richard S, Joseph Modayil, et al. (2011). "Horde: A scalable real-time architecture for learning knowledge from unsupervised sensorimotor interaction". In: *Proceedings of the 10th International Conference on Autonomous Agents and Multiagent Systems*. Vol. 2, pp. 761–768.

Sutton, Richard S, Doina Precup, & Satinder Singh (1999). "Between MDPs and semi-MDPs: A framework for temporal abstraction in reinforcement learning". In: *Artificial Intelligence* 112.1-2, pp. 181–211.

Tanaka, Saori C, Kenji Doya, et al. (2004). "Prediction of immediate and future rewards differentially recruits cortico-basal ganglia loops". In: *Nature Neuroscience* 7.8, p. 887.

Tanaka, Saori C, Nicolas Schweighofer, et al. (2007). "Serotonin differentially regulates short-and long-term prediction of rewards in the ventral and dorsal striatum". In: *PLoS One* 2.12, e1333.

Tangkaratt, Voot, Jun Morimoto, & Masashi Sugiyama (2016). "Model-based reinforcement learning with dimension reduction". In: *Neural Networks* 84, pp. 1–16.

Taylor, Matthew E & Peter Stone (2009). "Transfer learning for reinforcement learning domains: A survey". In: *The Journal of Machine Learning Research* 10, pp. 1633–1685.

Thorndike, Edward Lee (1911). *Animal Intelligence: Experimental Studies*. Macmillan.

Thrun, Sebastian & Anton Schwartz (1995). "Finding structure in reinforcement learning". In: *Advances in Neural Information Processing Systems 8*, pp. 385–392.

Tolman, Edward C (1948). "Cognitive maps in rats and men." In: *Psychological Review* 55.4, p. 189.

Tsai, Hsing-Chen et al. (2009). "Phasic firing in dopaminergic neurons is sufficient for behavioral conditioning". In: *Science* 324.5930, pp. 1080–1084.

Tsitsiklis, John N (1994). "Asynchronous stochastic approximation and Q-learning". In: *Machine Learning* 16.3, pp. 185–202.

Turner, L Richard (1966). *Inverse of the Vandermonde matrix with applications*. Tech. rep. NASA.

Uchibe, Eiji & Kenji Doya (2004). "Competitive-cooperative-concurrent reinforcement learning with importance sampling". In: *From animals to animats 8: Proceedings of the Eighth International Conference on the Simulation of Adaptive Behavior*, pp. 287–296.

Wang, Jiexin, Eiji Uchibe, & Kenji Doya (2017). "Adaptive Baseline Enhances EM-Based Policy Search: Validation in a View-Based Positioning Task of a Smartphone Balancer". In: *Frontiers in Neurorobotics* 11.

Watkins, Christopher John Cornish Hellaby (1989). "Learning from delayed rewards". PhD thesis. University of Cambridge England.

Watkins, Christopher John Cornish Hellaby & Peter Dayan (1992). "Q-learning". In: *Machine Learning* 8.3-4, pp. 279–292.

White, Adam, Joseph Modayil, & Richard S Sutton (2012). "Scaling life-long off-policy learning". In: *2012 IEEE International Conference on Development and Learning and Epigenetic Robotics (ICDL)*.

Williams, Ronald J (1992). "Simple statistical gradient-following algorithms for connectionist reinforcement learning". In: *Machine Learning* 8.3-4, pp. 229–256.

Yang, Shangdong et al. (2016). "Efficient Average Reward Reinforcement Learning Using Constant Shifting Values". In: *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*.

Yao, Hengshuai et al. (2014). "Universal option models". In: *Advances in Neural Information Processing Systems 27*, pp. 990–998.

Zhou, Li (2015). *A survey on contextual multi-armed bandits*. arXiv:1508.03326.