Learning and Planning with the Average-Reward Formulation

by

Yi Wan

A thesis submitted in partial fulfillment of the requirements for the degree of

Doctor of Philosophy

Department of Computing Science

University of Alberta

Abstract

The average-reward formulation is a natural and important formulation of learning and planning problems, yet has received much less attention than the episodic and discounted formulations. This dissertation makes three areas of contributions to algorithms and their theories concerning the average-reward formulation, primarily through the lens of reinforcement learning. The first area of contributions is a family of tabular learning and planning average-reward algorithms and their convergence theories. The second area of contributions of this dissertation is a complete extension of the options framework (Sutton, Precup, and Singh 1999) for temporal abstraction from the discounted formulation to the average-reward formulation. The extension includes general convergent off-policy inter-option learning algorithms, intra-option algorithms for learning values and models, as well as incremental planning variants of the learning algorithms, an option-interrupting algorithm, and convergence theories of the algorithms. The third area of contributions includes an average-reward prediction function approximation algorithm, its convergence analysis, and an error bound for the convergence point.

Preface

The chapters of this dissertation are based on four papers that I co-authored. More specifically, Chapter 3 is based on my contribution to two papers (Wan, Naik, and Sutton 2021a; Wan, Yu, and Sutton 2023). The work by Wan, Yu, and Sutton (2023) is currently prepared to be submitted. Chapter 4 is also based on this paper. In addition, Chapter 4 is based on my contribution to the paper by Wan, Naik, and Sutton (2021b). Chapter 5 is based on my contribution to a joint work by Zhang, Wan, Sutton, and Whiteson (2021).

Wan, Y., Naik, A., Sutton, R. S. (2021a). Learning and planning in average-reward Markov decision processes. In *Proceedings of the 38th International Conference on Machine Learning*, 10653–10662.

Wan, Y., Naik, A., Sutton, R. S. (2021b). Average-reward learning and planning with options. In *Advances in Neural Information Processing Systems, 34*, 22758–22769.

Zhang, S., Wan, Y., Sutton, R. S., Whiteson, S. (2021). Average-reward off-policy policy evaluation with function approximation. In *Proceedings of the 37th International Conference on Machine Learning*, 12578–12588.

Wan, Y., Yu, H., Sutton, R. S. (2023). On Convergence of Average-reward off-policy control algorithms in weakly communicating MDPs. *To Be Submitted. An Earlier Version in ArXiv:2209.15141.*

During the development of the research presented in my dissertation, I co-authored the following papers. Most of them (Wan et al. 2018; Wan et al.

2019; Kudashikina et al. 2021; Wan et al. 2022; Wan and Sutton 2022; He, Wan, and Mahmood 2022; He et al. 2023) are related to this dissertation and are mentioned in this dissertation while one (Hou et al. 2020) is not related and is therefore omitted.

Wan, Y., Zaheer, M., White, M., Sutton, R. S. (2018). Model-based reinforcement learning with non-linear expectation models and stochastic environments. In *The Joint IJCAI/ECAI/AAMAS/ICML Conference Workshop on Prediction and Generative Modeling in Reinforcement Learning.*

Wan, Y., Abbas, Z., White, A., White, M., Sutton, R. S. (2019). Planning with expectation models. In *Proceedings of the 28th International Joint Conference on Artificial Intelligence*, 3649–3655.

Hou, Z., Zhang, K., Wan, Y., Li, D., Fu, C., Yu, H. (2020). Off-policy maximum entropy reinforcement learning: Soft actor-critic with advantage weighted mixture policy (sac-awmp). *ArXiv:2002.02829.*

Kudashkina, K., Wan, Y., Naik, A., Sutton, R. S. (2021). Planning with expectation models for control. *ArXiv:2104.08543.*

Wan, Y., Rahimi-Kalahroudi, A., Rajendran, J., Momennejad, I., Chandar, S., van Seijen, H. (2022). Towards evaluating adaptivity of model-based reinforcement learning methods. In *Proceedings of the 39th International Conference on Machine Learning*, 22536–22561.

Wan, Y., Sutton, R. S. (2022). Toward discovering options that achieve faster planning. *ArXiv:2205.12515.*

He, J., Wan, Y., Mahmood, R. (2022), The Emphatic Approach to Average-Reward Policy Evaluation. In *NeurIPS 2022 Workshop on DeepRL.*

He, J., Che, F., Wan, Y., Mahmood, R. (2023), Consistent Emphatic Temporal-Difference Learning. Accepted by *the 39th Conference on Uncertainty in Artificial Intelligence.*

To my parents Junbiao Wan and Qun Su

for giving me the best they could.

The noblest pleasure is the joy of understanding.

– Leonardo da Vinci.

Acknowledgements

No words can express my gratitude to my supervisor Richard S. Sutton. Rich is one of the few who have influenced me the most. As a supervisor, Rich is admirable, inspiring, and extremely supportive. I still clearly remember the day when Rich taught Abhishek and me how to write a research paper for six straight hours. As a scientist, he is a genuine thinker, a respectful pioneer, and a visionary architect. Working with him is the best thing I have ever had in my life. I'd like to extend my heartfelt thanks to Huizhen (Janey) Yu. While she played an unofficial supervisory role during my Ph.D., her contributions were invaluable. As a globally recognized expert in MDP and RL theory, Janey consistently provided timely and insightful feedback on my queries. She patiently navigated me through the average-reward literature, enhancing my grasp of this domain and mentoring me in the clear presentation of technical results. I thank Csaba Szepevari, Martha White, Dale Schuurmans, and Benjamin Van Roy for carefully examining my candidacy and dissertation and sharing insightful opinions and advice. Csaba's and Ben's questions raised in the candidacy exam inspired me to rethink the problem a generally intelligent system solves. Martha's and Dale's comments sharpen my understanding of both the algorithms I studied and existing ones. Special thanks to Abhishek Naik, Huizhen (Janey) Yu, Richard S. Sutton, and Shangtong Zhang for their close collaboration on the papers that this dissertation is based on. I gained valuable insights from our close collaboration with each of them and truly cherished the time we spent working together. I thank Arsalan Sharifnassab and Abhishek Naik for reading the early versions of this dissertation and sharing great comments. I thank Martha Steenstrup for teaching me how to write in her writing boot camp and for always giving me valuable feedback for my presentations. I also thank Beverly Balaski, our great lab administrator, who repeatedly helped me with the logistics work. Without her, work could not be as easy as it was. I thank Alberto Pozanco Lancho, Daniel Borrajo, Leo Ardon, Sumitra Ganesh, and Manuela Veloso for hosting my internship at JP Morgan AI Research and for their firm support. I thank Sarath Chandar and Harm van Seijen for hosting my internship at Mila and for their great guidance on my research during and after the internship. I thank Hengshuai Yao for his support and great advice during my internship at Huawei. I thank all my wonderful collaborators: Richard S. Sutton, Abhishek Naik, Huizhen (Janey) Yu, Zaheer Abbass, Shangtong Zhang, Ali Rahimi-Kalahroudi, Zhimin Hou, Katya Kudashkina, Kris De Asis, Alan Chan, Adam White, Martha White, Shimon Whiteson, Janarthanan Rajendran, Ida Momennejad, Sarath Chandar, Harm van Seijen, Brian Tanner, Daniel Plop, Jiamin He, Fengdi Che, and Rupam Mahmood. Working with them has been a fantastic experience. I thank people in the RL-planning group for discussing fundamental problems in planning. They are Richard S. Sutton, Mike Bowling, Katya Kudashkina, Joseph Modayil, John Martin, Arsalan Sharifnassab, Esra'a Saleh, and Alex Ayoub. I thank all members of our Reinforcement Learning and Artificial Intelligence (RLAI) lab. I also thank my undergraduate advisor Kai Yu and master advisor Benjamin Kuipers. Kai introduced me to artificial intelligence and unreservedly shared his wise advice on my research and career. Ben guided me to start studying reinforcement learning and introduced me to Rich. Without them, I may not even consider pursuing a Ph.D. in artificial intelligence or reinforcement learning. Finally, I thank my family and friends, without them, I can not imagine how to get through all the hardness.

Contents

List of Figures

Chapter 1

Average-Reward Reinforcement Learning

This dissertation develops the average-reward formulation of learning and planning problems, primarily from the reinforcement learning perspective. The average-reward formulation is a simple and natural formulation for problems in which long-term performance is preferred. It has been extensively researched in planning problems, which constitute the core focus of operations research—a field that develops and implements analytical methodologies to enhance decision-making in various domains, including business, industry, and society. However, the average-reward formulation has been underdeveloped in learning problems, which are the primary focus of reinforcement learning—a subfield of artificial intelligence concerned with constructing intelligent systems.

The primary objective of this dissertation is to explore and develop theoretically sound *learning* algorithms concerning the average-reward formulation, thereby contributing to the advancement of reinforcement learning. The *planning* variant of the learning algorithms and the associated theories could also be of interest to readers from operations research.

The contributions of this dissertation include both algorithmic and theoretical improvements for several average-reward sub-problems. The algorithmic contributions include learning algorithms that are less sensitive to hyper-parameters and that work under a wider range of training conditions (i.e., off policy), planning algorithms that perform more incremental and flexible updates, learning and planning algorithms that work with temporally extended courses of action, and a learning algorithm that works well with approximations. The theoretical contributions include convergence results for all the new algorithms. My most important theoretical contribution is to show that certain new and existing average-reward algorithms converge under a wider range of training conditions than has been used in previous work.

1.1 The Reinforcement Learning Approach to Intelligence

The importance of intelligence has been recognized for a long time. For example, in the 4th century BC, Socrates said that "all philosophers agree—whereby they really exalt themselves—that intelligence is the king of heaven and earth. Perhaps they are right" (documented by Plato in his book Philebus). Later, Socrates affirmed that the subsequent discussion "confirms the utterances of those who declared of old that intelligence always rules the universe".

It is useful to have a definition for intelligence, for the same reason that it is useful to have definitions for other important concepts such as economics, the universe, and the arts. A clear definition of a concept establishes the boundaries for exploring and comprehending the concept and facilitates effective communication among individuals while discussing the concept.

"The computational part of the ability to achieve goals" (McCarthy 2007) is the definition of intelligence that I choose in this dissertation. This definition is particularly relevant to my discussion and aligns well with my perspective on intelligence. By emphasizing the computational aspect, this definition underscores that intelligence is not based on physical ability, such as strength or speed, which I find reasonable. Moreover, McCarthy's definition also underscores that intelligence is about the accomplishment of *goals*, which includes a great number of possibilities and is not limited to achieving certain goals like what we humans can do. In my view, this more comprehensive way to define intelligence is, again, more reasonable.

McCarthy's definition also echoes the ancient philosophers' belief that intelligence is the most formidable power in the universe, because achieving goals, regardless of their nature, is arguably the most powerful thing in the world.

Many fields talks about goals/intelligence in terms of a number to be optimized. For example, the field of optimal control theory deals with finding a control for a dynamic system over a period of time so that the "performance measure" (also known as "cost to go") is minimized. The field of operations research concerns how to conduct operations within an organization in order to maximize a number called the "overall measure of performance", which can be a composite of multiple objectives of interest such as time and profit. In the field of economics, this number is called the "utility" and measures the satisfaction or pleasure that consumers receive for consuming a good or service.

Reinforcement learning (RL) is also a field that has the intelligence goal being optimizing a number. In RL, an agent observes an indication of the state of the world and takes an action, which is received by the world. The world then emits a reward and the next state in response to the agent's action. In order to choose actions, the agent uses a special function named a *policy*, which maps from a state to an action. In RL, the number to be optimized is called the "value", which is the cumulative reward of the policy.

I will work within the field of RL in this dissertation because I find it a very appealing approach to intelligence. The RL agent achieves its goal by learning which actions to choose from its own experience in the world, much like how natural intelligence systems do. And natural intelligence systems, like humans and animals, are widely believed as being successful. This similarity between the RL agent and natural intelligence systems makes it convenient to gain inspiration from research fields that study natural intelligence like psychology and neuroscience, which had a long history. In fact, some of the key ideas of RL, including temporal-difference error and eligibility trace, had been related to what psychologists and neuroscientists had discovered. For a detailed discussion on how RL ideas are related to those studied in psychology and neuroscience, see Chapter 14 and Chapter 15 of the book by Sutton and Barto (2018).

The field of reinforcement learning (RL) is different from other fields that study goals/intelligence by emphasizing a bit more methods that can be used in a wide range of domains without relying on domain knowledge. While some RL methods leverage domain knowledge or apply only to certain domains, it is more common in RL to develop methods that do not need such knowledge and can be used widely. These general methods include classic RL methods like Temporal-Difference learning (Sutton 1988), Q-learning (Watkins and Dayan 1992), and the actor-critic method (Barto, Sutton, and Anderson 1983), as well as more modern ones like the Proximal Policy Optimization algorithm (Schulman et al. 2017). Other fields are more commonly concerned with specialized methods for certain domains and methods that leverage domain knowledge. For example, it is common in the field of operations research to study methods for certain domains like queuing and inventory control. For another example, in the field of Markov Decision Process, methods typically have full knowledge of the domain.

Many impressive successes pertained to intelligence involve reinforcement learning. For example, RL has been involved in the development of a program developed by the OpenAI company in 2023, called ChatGPT, which generates responses to human instructions. This system can produce surprisingly good responses to a very wide range of instructions, including drafting an email, fixing bugs in code, and summarizing key points of an article. Previous systems either can not take human instruction as input or can not provide useful responses to human instructions. OpenAI built ChatGPT, which can understand human instructions and provide instructs that humans like, by combining self-supervised learning, supervised learning, and reinforcement learning. For another example, RL has been involved in a Go program called AlphaGo (Silver et al. 2016), which won the game against the world Go champion Lee Sedol in 2016. The game of Go has been a difficult challenge for artificial intelligence over the years due to its enormous search space. Before AlphaGo, no Go program was able to achieve a place near the level of a human Go master. The creators of AlphaGo addressed this challenge by combining supervised learning, Monte Carlo tree search, and reinforcement learning. In addition to ChatGPT and AlphaGo, RL has been involved in a number of impressive successes including achieving human-level performance in video game playing (Mnih et al. 2015; Berner et al. 2019), regulating temperatures and airflow inside a large-scale data center (Lazic et al. 2018), controlling an autonomous helicopter (Kim et al. 2003), and so on.

1.2 The Average-Reward Formulation of Reinforcement Learning

In reinforcement learning, there are three main ways to summarize a sequence of rewards into a number to be optimized. The first way computes the expectation of the discounted total reward, with weights decaying exponentially over time. The rate of decay is controlled by a scalar called *discount factor*, which is always non-negative and less than one. This way of summarizing rewards is called the *discounted* formulation. Another way, known as the *average-reward* formulation, computes the average-reward rate per step, with equal weights given to immediate and future rewards. The third way is used in worlds with a special terminal state, at which point the world is reset to the start state. This formulation is called *episodic* because the agent's experience can be organized into a series of episodes, each of which begins at the start state and ends at the terminal state. The maximized number in this formulation is the expected total reward within each episode.

The average-reward rate concerned by the average-reward formulation is a simple and straightforward measure of long-term performance, which can be preferred in many real-world problems. For example, for an intelligent system that recommends content to users, it is common to maximize the click-through rate (see e.g. Warlop, Lazaric, and Mary 2018), which is the ratio between the number of users who click on a specific link of a recommended content and the number of users who view the content. The click-through rate is just the average-reward rate if we assign each click with a reward of one and each view without a click with a reward of zero. For another example, in an intelligent *call admission control and routing* system (Marbach, Mihatsch, and Tsitsiklis

2000), the agent determines whether to accept a call and chooses a route to send the call to the desired destination if the call is accepted. Each call has an associated reward and the goal of the intelligent system is to maximize the long-term average reward.

The average-reward formulation has been less well-developed than the other two formulations. For example, existing average-reward algorithms with theoretical guarantees either require choosing a reference function or require the data used by the algorithms to be generated in a specific way. The choice of the reference function can have a significant effect on the algorithms' performance. But it is not clear how to choose it well. The requirement for data generation limits the application of algorithms. For example, algorithms with such a requirement can not be applied to data generated by human experts. With the episodic and discounted formulations, we have algorithms (e.g., the Q-learning algorithm by Watkins and Dayan (1992)) that have theoretical guarantees, do not use a reference function, and work with data generated flexibly.

This dissertation concerns several sub-problems within the average-reward formulation. To better understand these sub-problems, the next section provides a taxonomy of sub-problems concerning the average-reward formulation and existing algorithms for these sub-problems. Useful surveys of average-reward learning are given by Mahadevan (1996a) and Dewanto et al. (2020).

1.3 Taxonomy of Average-Reward Sub-Problems

Problems within the field of RL can be divided into prediction problems and control problems. The goal of prediction problems is to evaluate the performance of a given *target* policy. The goal of control problems is to obtain the most performant policy. Prediction problems are not only interesting by themselves (Sutton et al. 2011) and are also solved by many control methods as a sub-step (see, e.g., Howard (1960); Konda (2002); Abbasi-Yadkori et al. (2019a;b)). Control problems are generally more challenging than prediction problems.

With the average-reward formulation, the goal of prediction problems is to estimate the differential value function (or bias in some literature) up to some additive constant and the average-reward rate for a given target policy. The differential value function summarizes the expected cumulative future excess rewards, which are the differences between received rewards and the reward rate. A policy better than the target policy can be derived from the target policy's differential value function, in which way prediction methods serve as a sub-step of control methods with the average-reward formulation.

Control problems with the average-reward formulation concern obtaining a policy that maximizes the average-reward rate.

Another way to classify average-reward problems is based on what representation is allowed to use. For *tabular* problems, each function of interest (e.g., the estimated differential values for all states) can be represented exactly using a look-up table. For function approximation (FA) problems, functions of interest may only be represented approximately. FA methods include tabular methods as a special case. Tabular methods are typically better understood in theory. FA methods are necessary for the great ambition of intelligence to achieve goals in worlds that are much larger than what the agent could completely represent (e.g., Mnih et al. 2015; Silver et al. 2016).

Problems within the field of RL can also be divided into *learning* problems, which require only the agent's interaction with the world, and *planning* problems which require only a model of the world. A planning method can be an essential part of a learning method. In particular, learning methods may learn a world model and then plan with the learned model. Such methods are therefore called *model-based* methods. Other learning methods are called *model-free* methods. Both model-based and model-free methods have been actively studied in RL.

Both prediction and control learning problems can be subdivided into *on-policy* problems, in which the data is generated by following the target policy [1] , and *off-policy* problems, in which the data is generated by a second *behavior* policy. In general, both policies may be non-stationary. For example, in control problems, the target policy is learned and gradually becomes more

[1] In the control case, the target policy is the current learned policy.

performant. Off-policy problems include on-policy problems as a special case, where the target and the behavior policies are identical. Off-policy methods are more flexible because the behavior policy can be different from and even independent of the target policy. A direct consequence of the flexibility is that by using off-policy prediction methods, multiple different policies can be evaluated using one stream of experience that is not generated from any of these policies. The flexibility also makes off-policy prediction methods key to efficient learning world models for temporally extended courses of action (see Section 17.2 by Sutton and Barto 2018). For control problems, this flexibility allows learning from experience generated by a non-learning policy, such as a human expert's policy.

On-policy prediction algorithms in the literature include average-cost TD(λ) (Tsitsiklis and Van Roy 1999), LSTD(λ) (Konda 2002), and LSPE(λ) (Yu and Bertsekas 2009). The theoretical properties of these algorithms have been well-understood with linear function approximation. These algorithms estimate the reward rate and the differential value function directly from data, thus are all model free. They all estimate the reward rate using the average of the received rewards. The estimated reward rate then converges to the true reward rate. They all use linear function approximation to estimate the differential value function. Their estimated differential value functions all converge to the same point. Because of the limited capacity of linear function approximation, in general, the convergent point is not the differential value function up to an additive constant. Two upper bounds of the distance between the convergent point and the differential value function up to an additive constant have been provided by Tsitsiklis and Van Roy (1999) and Yu and Bertsekas (2008). Rates of convergence of average-cost TD(λ) and LSTD(λ) have been established by Konda (2002). The rate of convergence of LSPE(λ) has been established by Yu and Bertsekas (2009). For the average-cost TD(λ) algorithm, an upper bound of the number of samples required to achieve a certain level of closeness to the final convergent point, which is also known as the sample complexity, has been established by Zhang, Zhang, and Maguluri (2021).

Existing off-policy prediction algorithms (e.g., Wen et al. 2020; Liu et al. 2018; Lazic et al. 2020; Tang et al. 2019; Mousavi et al. 2020; Zhang et al. 2020a,b) operate on a fixed given batch of data. These algorithms are called *batch* algorithms. Because the batch can not be infinitely large, batch algorithms would inevitably suffer from an error because the batch data only approximates the data distribution following the behavior policy. Increasing the size of the batch reduces this error with the cost of more memory usage.

Existing off-policy prediction algorithms all use function approximation to estimate the reward rate and do not seek to approximate the differential value function. To estimate the reward rate, many of them (e.g., Wen et al. 2020; Liu et al. 2018; Tang et al. 2019; Zhang et al. 2020a,b) first approximate the ratio of the stationary distribution under the target policy and the behavior policy, and then use that ratio to estimate the reward rate. The first step is difficult, after which the second step is straightforward. Alternatively, to estimate the reward rate, there also exists an algorithm that uses the estimated stationary distribution of the target policy (Mousavi et al. 2020) and an algorithm (Lazic et al. 2020) that uses an estimated world model.

Various kinds of theoretical results have been developed for off-policy prediction algorithms, but most results do not directly tell how well the algorithms estimate the reward rate, and there is only one algorithm that has been proven to obtain the true reward rate asymptotically. For example, Zhang et al. (2020b) provided a convergence result and a finite-sample analysis for their density-ratio-estimation algorithm in the linear FA setting but didn't state how accurate the estimated density ratios and the estimated reward rate are. For another example, Zhang et al. (2020a) provided a bound for the difference between the optimal solution of an optimization problem and the true density ratio, but the optimal solution may not be obtained using the optimization method that they used, as observed by Zhang et al. (2020b). Liu et al. (2018) also proposed to solve an optimization problem and related the solution of the optimization problem to the density ratio, but didn't state how to reach that optimal solution. Wen et al. (2022) showed that their algorithm converges to the true density ratio, but the algorithm uses an exact world model which the paper does not specify how to obtain. The only algorithm that has been proven to obtain the true reward rate asymptotically is by Lazic et al. (2020). They proposed a model-based algorithm, which estimates the world model and then computes the reward rate using the model. They further provided a bound of the error of the estimated reward rate w.r.t. the actual reward rate, for a setting that slightly generalizes over the tabular setting.

Various kinds of theoretical results have been shown for many on-policy control algorithms, including the asymptotic convergence, upper bounds of sample complexity, and upper bounds of regret. For formal definitions of sample complexity and regret, see Szepesvari (2010). On-policy algorithms that have been proven to have such theoretical results include tabular model-free algorithms (Wheeler and Narendra 1986; Wei et al. 2020), tabular model-based algorithms (Kearns and Singh 2002; Brafman and Tennenholtz 2002; Auer and Ortner 2006; Bartlett and Tewari 2009; Jaksch et al. 2010), and FA algorithms (Marbach and Tsitsiklis 2001; Konda 2002; Abbasi-Yadkori et al. 2019a;b; Hao et al. 2021).

There are only a few off-policy control algorithms that have a theoretical result for the obtained policy, and the theoretical results are available only for the tabular, discrete setting without function approximation. The only known convergent algorithms are SSP Q-learning and RVI Q-learning, both by Abounadi, Bertsekas, and Borkar (2001), and the algorithm by Ren and Krogh (2001). Others either do not have an associated theoretical result (Schwartz 1993; Singh 1994; Bertsekas and Tsitsiklis 1996; Das et al. 1999) or have incorrect proofs (Yang et al. 2016; Gosavi 2004). [2] The algorithm by Ren and Krogh (2001) requires knowledge of properties of the world which is not typically known. SSP Q-learning needs to know a special state in the world, which is again typically unknown to the agent. The RVI Q-learning algorithm is, therefore, the only known convergent off-policy control algorithm that does not require knowledge of the world.

Average-reward *planning* algorithms have been extensively studied in the field of Markov decision processes (MDPs). These algorithms have been known

[2]See Appendix D in Wan et al. (2021a) for a discussion about Yang's proof and see Section 3.6 of this dissertation for a discussion about Gosavi's proof.

at least since the setting was introduced by Howard in 1960. Early algorithms include policy iteration (Howard 1960), value iteration (Bellman 1957), relative value iteration (RVI, White 1963), and contracting value iteration (CVI, Bertsekas 1998).

The planning algorithms introduced above are ill-suited when real-time decision-making is desired because they involve sub-steps whose complexity is the order of the number of states or more. These algorithms are called synchronous algorithms, because they proceed in large iterations, each comprising a sweep over the full state (or state-action) space. On the other hand, some planning algorithms are asynchronous or incremental, which implies that they progress in small iterations, with each update addressing only a few states (or state-action pairs). By updating only a few states in each iteration, asynchronous/incremental planning algorithms can produce a policy within a brief amount of time. The policy improves as more time is devoted to the planning process. Therefore, asynchronous/incremental algorithms are more suitable for real-time decision-making.

More incremental average-reward planning algorithms have been under-explored. Almost all of the algorithms that have an associated theory either update for states/state-action pairs in a specific way or require knowledge of the world that is usually not available to the agent. For example, Jalali and Ferguson (1989, 1990) were among the first to explore more incremental algorithms, though their algorithms require knowledge of a special state in the world. For another example, Bertsekas (1998) then showed the convergence of a special asynchronous version of the CVI algorithm. This asynchronous algorithm needs to update states according to their order, and, just like the algorithms by Jalali and Ferguson, requires knowledge of a special state in the world. For another example, the Primal-Dual π Learning algorithm (Wang 2017) and the stochastic mirror descent algorithm by Jin and Sidford (2020) have been shown convergent. For both algorithms, the state-action pairs to update are not arbitrarily chosen, but sampled from the algorithms' estimate of the optimal state-action-pair distribution. The only incremental planning algorithm that can update for states/state-action pairs in a flexible way and does not require knowledge of the world is again RVI Q-learning, now applied as a planning algorithm to simulated experience generated by the model.

Extending all of the aforementioned average-reward sub-problems, which only involve primitive actions, we obtain average-reward sub-problems that involve temporally abstracted courses of action, or options (Sutton, Precup, and Singh 1999). Each option is defined as a tuple consisting of a policy that determines the option's behavior, an initiation set that includes states at which the option can be initiated, and a termination condition that describes how to terminate the option. Options include actions as special cases. Unlike actions, options can take multiple time steps to finish. Using options can have multiple potential benefits, which will be discussed in Chapter 4.

With options, it is natural to consider more generalized policies, which choose from options instead of primitive actions. Such policies are also called hierarchical policies. The agent may choose to follow a hierarchical policy, which means that it first chooses an option according to the hierarchical policy, and then follow the option's policy until the option terminates, at which point it chooses a new option. The goal of average-reward prediction sub-problems is then to obtain the reward rate and the differential value function (up to an additive constant) of the hierarchical policy. And the goal of average-reward control sub-problems is then to obtain a hierarchical policy that achieves the highest reward rate.

For both learning and planning sub-problems, options sub-problems can be subdivided into two classes: *inter-option* and *intra-option* sub-problems. In inter-option sub-problems, the experience used for learning/planning is a stream of option transitions. An option transition characterizes the execution of the option. It is a tuple consisting of five elements: the option being executed, the start and terminal states of the option, the duration (number of time steps) of the execution of the option, and the cumulative reward during the execution of the option. In intra-option sub-problems, the experience used for learning/planning is a stream of action transitions, each of which is a tuple consisting of four elements: the action being executed, the start and resulting states of the action, and the reward as a result of the action. Each option transition is a result of a series of action transitions. Inter- and intra-option *learning* sub-problems are different because, in the inter-option case, the behavior policy needs to be hierarchical so that option transitions can be obtained while it does not have to be in the intra-option case. Being constrained to a hierarchical behavior policy would be just fine when the agent has control over its behavior policy but can be an issue when the behavior policy is provided by a human expert and the policy is a non-hierarchical one.

Inter- and intra-option *planning* sub-problems are different because they involve different kinds of world models. Inter-option planning sub-problems involve models that predict outcomes of options. In contrast, intra-option planning sub-problems work with models that predict outcomes of actions. Because option models are jumpy, inter-option planning algorithms could be more efficient computationally. Finally, note that while action models can only be learned from the agent's experience, option models can be learned from real experience or be obtained by planning with action models.

Average-reward options algorithms have been underdeveloped in the literature. There are only a few existing control learning algorithms (Das et al. 1999; Gosavi 2004; Vien and Chung 2008), and none of them have been proven to obtain the optimal hierarchical optimal policy [3]. In addition, all of the existing average-reward learning algorithms are inter option and there are no known intra-option average-reward algorithms in the literature. Further, the existing proven-convergent planning algorithms (e.g., Schweitzer 1971, Puterman 1994, Li and Cao 2010) are all synchronous because they perform a full sweep over states for each planning step.

Finally, it might be tempting to ask if algorithms that work with primitive actions can be directly applied by replacing actions with options. In general, with the average-reward formulation, it is not appropriate to directly apply algorithms that work with actions, like those mentioned above, to a set of options. Algorithms that operate with actions do not take into consideration the amount of time when executing an action, because all actions take the same amount of time to execute (more specifically, a single time step). However,

[3]Gosavi's (2004) proof of his options algorithm is incorrect, just as in the action case.

different options usually take different amounts of time to execute and it is not appropriate to ignore the difference between the amounts of execution time. For example, two options that accumulate the same reward and reach the same state are not necessarily equally good because one might take more steps to finish than the other one. The option that takes fewer steps to finish is better because the reward rate with this option is higher.

1.4 Contributions

This dissertation contributes to several average-reward sub-problems by introducing novel average-reward algorithms and their corresponding theories for these sub-problems. These sub-problems include several tabular sub-problems and a function approximation sub-problem. The tabular sub-problems include off-policy learning and planning prediction and control sub-problems as well as the options versions of these sub-problems. The function approximation sub-problem is the off-policy prediction learning sub-problem, which, as discussed before, generalizes the on-policy prediction learning sub-problem.

All of the newly developed algorithms distinguish themselves from most of their predecessors by their approach to estimating the average-reward rate. Most previous methods either update the estimate of the average-reward rate using the difference between the actual reward and the estimated average-reward rate or do not maintain an estimate at all. On the other hand, the newly proposed algorithms update the reward rate estimate using Temporal-Difference (TD) errors. These TD errors involve the differential rewards, i.e., the difference between the actual reward and the estimated reward rate, rather than the rewards themselves. Consequently, I have classified algorithms that update the average-reward rate estimate using TD errors as *differential algorithms*. While two earlier algorithms (Schwartz 1993; Singh 1994) also adopted the differential idea and are therefore also differential algorithms, they lack associated theoretical analyses, unlike the algorithms introduced in this dissertation.

The contributions of this dissertation can be grouped into three areas.

The first area of contributions of this dissertation (Chapter 3), based on two papers I co-authored (Wan, Naik, and Sutton 2021a; Wan, Yu, and Sutton 2023), is *a family of average-reward tabular learning and planning algorithms and their convergence theories*. The algorithms were developed by Abhishek Naik, Richard S. Sutton, and myself. Most of the theoretical results were developed by myself. Huizhen Yu corrected errors in early versions of the results and helped improve the presentation of the results.

Among the family of algorithms, our prediction learning algorithm, *Differential TD-learning*, is the first off-policy model-free prediction algorithm *proved* to obtain the reward rate of the target policy asymptotically. It is also the first off-policy prediction algorithm proved to obtain the *differential value function* up to some additive constant asymptotically (remember that all existing methods do not estimate the differential value function). Differential TD-learning is also different from existing algorithms in that existing off-policy algorithms either operate on a fixed batch of data or operate on i.i.d. data generated from a sampling distribution, while Differential TD-learning is fully online and operates on a single sample trajectory.

Our control learning algorithm, *Differential Q-learning*, is the first off-policy model-free control algorithm proved to achieve the optimal reward rate *without using a reference function*. In addition, the convergence analysis of Differential Q-learning only assumes *weakly communicating MDPs*, which are more general than the unichain assumption made in the convergence theory of the most important existing off-policy model-free control algorithm, RVI Q-learning, by Abounadi, Bertseka, and Borkar (2001). Weakly communicating MDPs are the most general class of MDPs in which a learning algorithm with a single stream of experience could achieve the optimal reward rate, regardless of the start state of the experience. Using a similar analysis, the dissertation shows that RVI Q-learning also converges in weakly communicating MDPs. These are the first results showing that model-free off-policy average-reward algorithms converge in weakly communicating MDPs.

The planning versions of the learning algorithms, *Differential TD-planning* and *Differential Q-planning* are fully incremental and well suited for use in reinforcement learning architectures (e.g., Dyna (Sutton 1990)) and are proved convergent just as learning algorithms. The new planning algorithms, just like the planning version of RVI Q-learning, update the state-action pairs flexibly and do not require special knowledge of the world.

All the aforementioned average-reward algorithms converge not to the actual differential value function, but to the differential value function plus an offset that depends on initial conditions or the reference function. The offset is not necessarily a problem because only the relative values of states (or of state-action pairs) are used to determine policies. However, the actual differential value function of any policy is *centered*, meaning that the mean value of states encountered under the policy is zero. Although it is easy to center an estimated value function in the on-policy case, in the off-policy case it is not. As part of the first contribution, this dissertation provides centered versions of our off-policy algorithms and their proof of convergence to the actual differential value function.

The second contribution of this dissertation (Chapter 4), based on two papers I co-authored (Wan, Naik, and Sutton 2021b; Wan, Yu, and Sutton 2023), is *a complete extension of the options framework (Sutton, Precup, and Singh 1999) from the discounted formulation to the average-reward formulation*, enabling widespread use of options for the average-reward formulation. The extension includes general convergent off-policy inter-option learning algorithms, intra-option algorithms for learning values and models, as well as incremental planning variants of our learning algorithms, and an option-interrupting algorithm. There is yet another contribution independent of the options algorithms—I proved a bound of sub-optimality of the reward rate achieved by greedy policies induced by any action-value estimate in weakly communicating SMDPs. This result generalizes Puterman's (1994) Theorem 8.5.5 from unichain Markov Decision Processes to weakly communicating Semi-Markov Decision Processes.

The inter-option prediction and control learning algorithms are the first such algorithms that learn with off-policy data and are guaranteed to converge. These inter-option algorithms extend Differential TD-learning and Differential Q-learning. Extending these algorithms to inter-option algorithms is, however, non-trivial, as highlighted in Chapter 4. The planning version of our inter-option learning algorithms is the first convergent *incremental* planning algorithm.

The proposed intra-option learning and planning algorithms for both values and models are the first such algorithms, filling a gap in the average-reward literature. These algorithms are stochastic approximation algorithms solving the average-reward intra-option value and model equations. The intra-option value algorithms, just like their inter-option counterparts, extend Differential Q-learning and Differential TD-learning from actions to options.

This dissertation extends an *interruption* algorithm from the discounted to the average-reward formulation. The idea of interruption, introduced by Sutton, Precup, and Singh (1999), is simple and natural. Instead of letting an option execute to termination, interruption algorithms involve potentially interrupting an option's execution to check if starting a new option might yield a better outcome. If so, then the currently executing option is terminated, after which the new option is executed. An interruption mechanism determines how the agent should behave and are independent of how the option values are obtained. However, it is usually not convenient to perform inter-option learning, because the execution of options can be interrupted and complete option transitions are not necessarily obtained.

The third contributions of this dissertation (Chapter 5), based on a paper I co-authored (Zhang, Wan, Sutton, and Whiteson 2021), include *an average-reward prediction function approximation algorithm, called Differential GQ1, its convergence analysis, and an error bound for the convergent point.* Differential GQ1, together with its sibling algorithm Differential GQ2 (Zhang et al. 2021), are the first proven convergent algorithms that estimate both the reward rate and the differential value function via off-policy linear function approximation. The credit for the development of Differential GQ2 and its theory should go to Shangtong Zhang. Both Differential GQ1 and Differential GQ2 are inspired by the celebrated Gradient TD family of algorithms (Sutton et al. 2009), which optimize the mean squared projected Bellman error (MSPBE) for the *discounted* formulation. While for the discounted formulation, there is only one MSPBE, for the average-reward formulation we developed two different MSPBEs. The two average-reward algorithms optimize the two MSPBEs respectively. Interestingly, while the two MSPBEs are different, their solutions are the same if the solutions uniquely exist.

Finally, to empirically verify convergence theories proved in the dissertation, I also performed proof-of-concept experiments and provide experiment results for most of the proposed learning algorithms. Most of these empirical results are new—they have not been presented in the three published papers this dissertation is based on (Wan, Naik, and Sutton 2021a;b; Zhang et al. 2021).

Chapter 2

Markov Chains and Markov Decision Processes

This chapter presents some basic definitions and existing technical results that serve as a basis for presenting the contributions of the dissertation. The definitions and results presented in this chapter are widely used in the literature and can be found in textbooks like Lawler (2018) and Puterman (1994). Readers who are familiar with these topics may choose to skip this chapter.

2.1 Markov Chains

A finite Markov chain is a discrete-time stochastic process, S_n, $n = 0, 1, 2, \ldots$, where S_n takes value from a state space with finite elements, or a finite state space $\mathcal{S}$. This stochastic process satisfies the Markov property—the probability of each sequence of realizations can be defined using a product of transition probabilities. Formally, for each $n \geq 1$, the probability of observing a sequence of realizations $s_0, s_1, \ldots, s_n$,

$$\Pr(S_0 = s_0, S_1 = s_1, \ldots, S_n = s_n) \doteq d(s_0) \prod_{t=0}^{n-1} p(s_{t+1} \mid s_t).$$

Here $d_0 \in \Delta(\mathcal{S})$ [1] is the *initial state distribution* with $d_0(s)$ [2] defined as the probability that the stochastic process starts from state s and $p : \mathcal{S} \to \Delta(\mathcal{S})$ is the *transition function* with $p(s' \mid s)$ defined as the probability of transitioning to s' from s. Write the transition probability in the matrix form: $P[s, s'] \doteq p(s' \mid s)$. We call P the transition matrix of the Markov chain. A probability vector d is a *stationary* distribution of a Markov chain with transition matrix P if $d^\top = d^\top P$.

Given a Markov chain with a finite number of states, we say a state is *recurrent* if starting from this state the Markov chain returns to this state infinitely often, with probability one. A state is called *transient* if it is not recurrent.

We say two states of a Markov chain communicate with each other, if starting from each state, the Markov chain reaches the other state with a positive probability. It is clear that if state a communicates with state b and state b communicates with state c, then state a communicates with state c. States in the Markov chain can then be partitioned into one or multiple communication classes. Within each communicating class, every state communicates with all other states in the class. If the Markov chain only has one communicating class, the chain is called *irreducible*. Otherwise, it is called *reducible*.

A communicating class is called *transient* if, with probability one, the Markov chain eventually leaves this class and never returns. A communicating class is called *recurrent* otherwise. States in *transient* classes are called transient states and states within recurrent classes are called recurrent states. A Markov chain is recurrent if all states belong to the same recurrent class. If a finite Markov chain is recurrent, it is irreducible and vice versa. A finite Markov chain is called *unichain* if there is a recurrent class and a possibly empty set of transient states. If a Markov chain is unichain, there is a unique stationary distribution.

[1] $\Delta(\mathcal{S})$ is the probability simplex over $\mathcal{S}$. A probability simplex over a set $\mathcal{X}$ is the set of all possible probability vectors over $\mathcal{X}$. Given a countably infinite set $|\mathcal{X}|$, denote $|\mathcal{X}|$ as the cardinality of $\mathcal{X}$, any vector $x \in \mathbb{R}^{|\mathcal{X}|}$ satisfying $x(i) \geq 0, \forall i = 1, 2, \ldots, |\mathcal{X}|$ and $\sum_{i=1}^{|\mathcal{X}|} x(i) = 1$ is a probability vector over $\mathcal{X}$.

[2] Through this dissertation, vectors are always written in columns. In addition, this dissertation uses lower cases for vectors and upper cases for matrices, both in plain text.

The period of a state s in a Markov chain is the greatest common divisor of $\{n \geq 0 : p^{(n)}(s \mid s) > 0\}$, where $p^{(n)}(s \mid s)$ is the transition probability from state s back to itself in n steps. It is known that all states within the same communicating class share the same period. We say an irreducible Markov chain *aperiodic* if the period shared by all states is 1. If a Markov chain is irreducible and aperiodic, it is called *ergodic*.

2.2 Markov Decision Processes

Markov decision processes extend Markov chains by adding actions and rewards. A finite Markov decision process (MDP) is defined by the tuple $\mathcal{M} \doteq (\mathcal{S}, \mathcal{A}, \mathcal{R}, d_0, p)$, where $\mathcal{S}$ is a finite set of states, $\mathcal{A}$ is a finite set of actions, $\mathcal{R}$ is a finite set of rewards, $d_0 \in \Delta(\mathcal{S})$ is an initial state distribution, and $p : \mathcal{S} \times \mathcal{A} \to \Delta(\mathcal{S} \times \mathcal{R})$ is a transition function with $p(s', r \mid s, a)$ defined as the probability of transitioning to state $s' \in \mathcal{S}$ and observing a reward $r \in \mathcal{R}$ when taking action $a \in \mathcal{A}$ from state $s \in \mathcal{S}$. All MDPs considered in this dissertation are finite.

Given an MDP, a *stationary policy* $\pi : \mathcal{S} \to \Delta(\mathcal{A})$ is a function with $\pi(a \mid s), \forall s \in \mathcal{S}, a \in \mathcal{A}$ denotes the probability of choosing action a at state s. Denote the set of all stationary policies Π. Given an MDP $\mathcal{M} \doteq (\mathcal{S}, \mathcal{A}, \mathcal{R}, d_0, p)$, a policy $\pi \in \Pi$ induces a Markov chain, which has state space $\mathcal{S}$, initial state distribution $\phi = d_0$, and transition function $p(s' \mid s) = \sum_{a,r} \pi(a \mid s) p(s', r \mid s, a)$.

An MDP is called unichain/recurrent/ergodic if the Markov chain induced by any stationary policy in the MDP is unichain/recurrent/ergodic. Given an MDP, a policy is said to be unichain/recurrent/ergodic if the Markov chain induced by any stationary policy in the MDP is unichain/recurrent/ergodic.

Given an MDP, we say two states communicate with each other if there exists a stationary policy that transitions from each one to the other one with non-zero probability in a finite number of steps. A set of states is called communicating if every state within the set communicates with all other states. An MDP is said to be *communicating* if all states belong to the same communicating class. An MDP is said to be *weakly communicating* if there is a *closed* communicating class and a possibly empty set of states that are transient under all stationary policies. A set is closed if starting from any state within the set, there is a zero probability of transiting to states outside the set, regardless of the sequence of actions. Examples of recurrent, unichain, communicating, weakly communicating, and other MDPs are shown in Figure 2.1.

There is a special class of MDPs considered by several average-reward algorithms. These MDPs are unichain and have a state being recurrent under *every stationary policy*. These MDPs are considered in the convergence results of SSP Q-learning (Abounadi et al. 2001), Gosavi's (2004) algorithm, and Contractive Value Iteration (Bertsekas 1998). Note that a unichain MDP does not necessarily have a common recurrent state (e.g., Figure 2.2) and that the existence of a common recurrent state does not imply that the MDP is unichain (e.g., Figure 2.3).

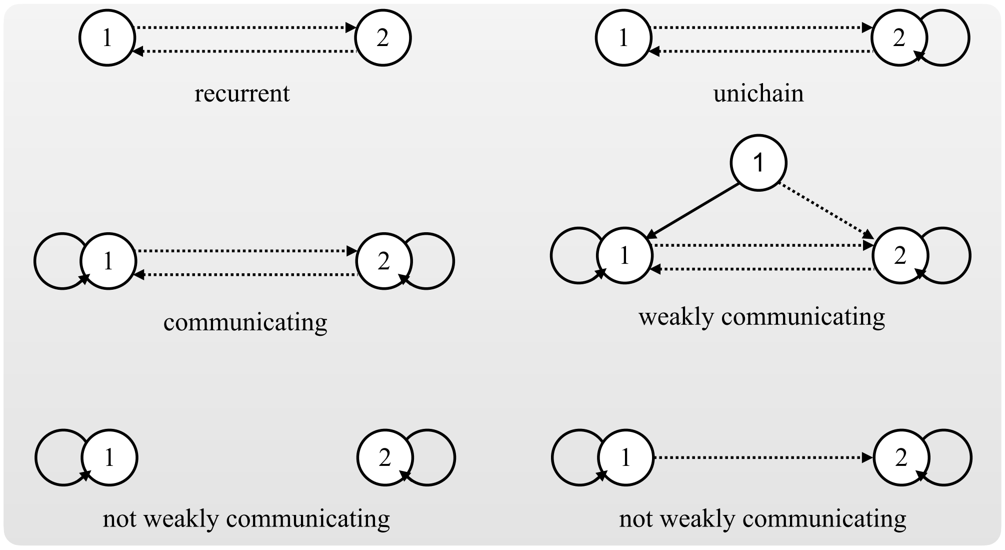


Figure 2.1: Examples of recurrent, unichain, communicating, weakly communicating, and other MDPs.

There is yet another class of MDPs considered in the convergence theorem of Differential Q-learning by Wan et al. (2021a). And it can be shown that RVI Q-learning also converges in this set of MDPs. For this class of MDPs, the solution set of q in the optimality equation (3.7) (introduced in the next chapter) has one degree of freedom (i.e., all solutions are only different by a constant). This class of MDPs is more general than unichain MDPs but is less general than weakly communicating MDPs.

An illustration of the hierarchy of MDP classes is shown in Figure 2.4.

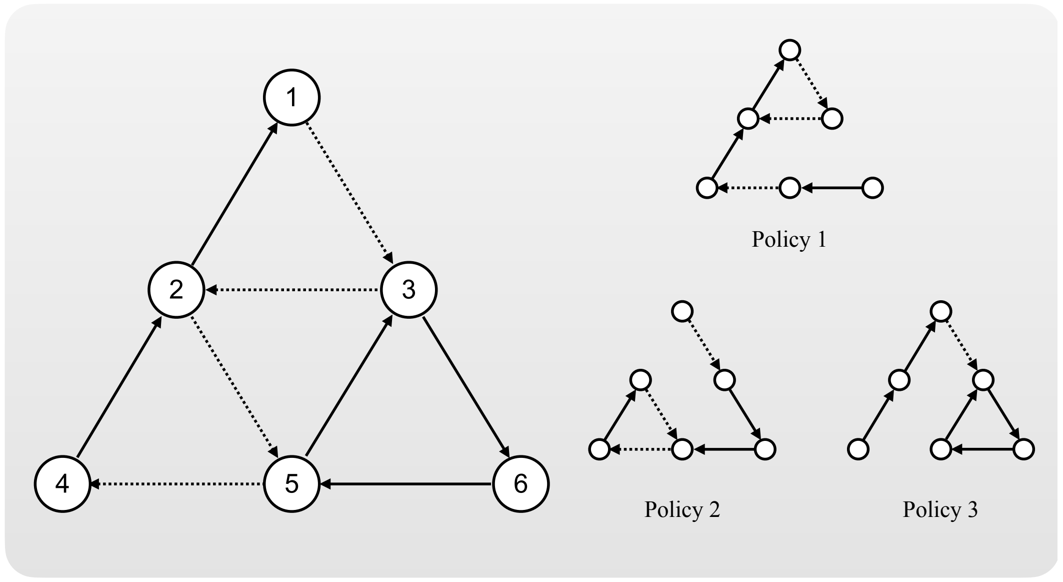


Figure 2.2: A unichain MDP that does not have a state being recurrent under every stationary policy.

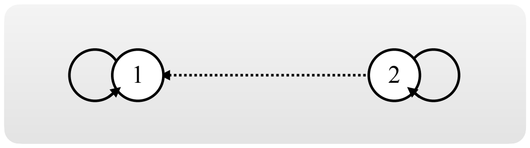


Figure 2.3: A multi-chain MDP that has state 1 being recurrent under every stationary policy.

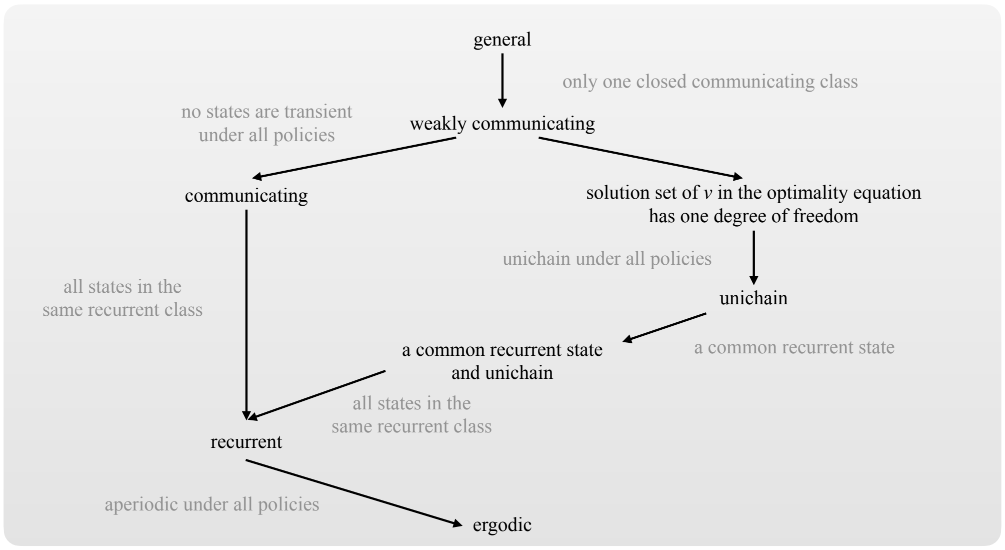


Figure 2.4: Hierarchy of MDP classes

Chapter 3

Tabular Algorithms

This chapter presents the first area of contributions of this dissertation: a family of tabular algorithms for the average-reward formulation as well as the algorithms' convergence theories. This family of algorithms includes both prediction and control algorithms. And for each of the prediction and control cases, this family includes an off-policy learning algorithm and a corresponding planning algorithm. Both prediction and control learning algorithms are off policy. These algorithms all use value estimates, which are guaranteed to converge to the true value function up to an additive constant. For all of the algorithms, this chapter further presents their centered versions, which produce the true differential value function instead of one with an offset. This chapter presents convergence theories for all of the algorithms, including both uncentered and centered versions. In order to do so, this chapter proves the convergence theory of a general algorithm, which encompasses all of the algorithms introduced in this chapter as special cases. Finally, several sets of simple experiments are presented in this chapter to demonstrate the empirical performance of some of the proposed algorithms.

3.1 Problem Setup

This section sets up the tabular learning and planning problems with the average-reward formulation.

Reinforcement learning (RL) is a formulation of the problem of an agent achieving its goal in the world. Classic RL formalizes the agent's interaction with the world by a finite Markov decision process $\mathcal{M} = (\mathcal{S}, \mathcal{A}, \mathcal{R}, d_0, p)$. The agent's interaction with the world starts from one of the initial states S_0, sampled from the initial state distribution d_0. At each of a sequence of discrete time steps $t = 0, 1, 2, \ldots$, the agent observes a state S_t and selects, using a behavior policy $b : \mathcal{S} \to \Delta(\mathcal{A})$, an action $A_t \in \mathcal{A}$, then receives from the world a reward $R_{t+1} \in \mathcal{R}$ and the next state $S_{t+1} \in \mathcal{S}$, and so on. The transition probability follows p: $\Pr(S_{t+1} = s', R_{t+1} = r \mid S_t = s, A_t = a) = p(s', r \mid s, a)$ for all $s, s' \in \mathcal{S}, a \in \mathcal{A}$, and $r \in \mathcal{R}$. For convenience, with a bit of abuse of notation, I will also use $p(s' \mid s, a)$ to denote the state part of the transition function (i.e., $p(s' \mid s, a) \doteq \sum_r p(s', r \mid s, a)$ for all s, a). Learning algorithms only use the agent's real experience $\{S_t, A_t, R_{t+1}, S_{t+1}\}_{t \geq 0}$. Planning algorithms use a model of the MDP, $\hat{p} \approx p$.

The average-reward formulation seeks a policy π within the class of stationary policies Π that maximizes, for each state $s \in \mathcal{S}$, the agent's average-reward rate following a policy π starting from s:

$$r(\pi, s) \doteq \lim_{n \to \infty} \frac{1}{n} \sum_{t=1}^{n} \mathbb{E}[R_t \mid S_0 = s, A_{0:t-1} \sim \pi], \tag{3.1}$$

where the limit always exists. The reward rate measures the long-run performance of a policy.

Another important quantity, called *differential value function* (also called bias; see, e.g., Puterman 1994), represents the relative "goodness" of states and can be used to rank states. The differential value function $v_\pi : \mathcal{S} \to \mathbb{R}$ for a policy $\pi \in \Pi$ is:

$$v_\pi(s) \doteq \lim_{n \to \infty} \frac{1}{n} \sum_{k=1}^{n} \sum_{t=1}^{k} \mathbb{E}\left[R_t - r(\pi, s) \mid S_0 = s, A_{0:t-1} \sim \pi\right],$$

for all $s \in \mathcal{S}$, where the limit always exists. Tabular *prediction* learning and planning algorithms seek, given a policy $\pi \in \Pi$, $r(\pi, s)$ and $v_\pi(s) + c$ for all $s \in \mathcal{S}$ for some constant c.

Note that by the above definition, prediction algorithms do not seek the differential value function but only the differential value function plus some constant vector. This is because the differential value function is typically used by control algorithms that iteratively evaluate a policy and improve the policy (e.g. policy iteration by Howard (1960), actor-critic by Konda (2002)). For this purpose, it is not necessary to obtain the differential value function—any function that differs from the differential value function by a constant vector would be no different. Consequently, algorithms in the literature typically do not seek the differential value function but any function that equals the differential value function up to an additive constant vector.

It is convenient to rule out the possibility of the reward rate of π depending on the start state. In particular, assume that under the policy being evaluated, there is only a set of recurrent states. This is known as the Markov chain being *unichain*. Under the unichain assumption, there exists a unique reward rate $r(\pi)$ that does not depend on the start state:

$$r(\pi) \doteq r(\pi, s), \forall s \in \mathcal{S}.$$

Under the unichain assumption, the *state-value evaluation equation* is

$$v(s) = r_\pi(s) - \bar{r} + P_\pi v(s),\ \forall s \in \mathcal{S}, \tag{3.2}$$

where

$$P_\pi(s, s') \doteq \sum_a \pi(a \mid s) p(s' \mid s, a) \tag{3.3}$$

is the transition matrix under policy π, and

$$r_\pi(s) \doteq \sum_a \pi(a \mid s) \sum_{s', r} p(s', r \mid s, a) r. \tag{3.4}$$

is the expected reward starting from state s under policy π. The state-value evaluation equation has $r(\pi)$ as its unique solution of $\bar{r}$ and any solution of $v : \mathcal{S} \to \mathbb{R}$ satisfies $v = v_\pi + c\mathbf{1}$ for some constant c. Throughout this dissertation, $\mathbf{1}$ is used to denote an all-one vector with dimension implied from the context. Similarly, $\mathbf{0}$ is used to denote an all-zero vector with dimension implied from the context. Similarly, the *action-value evaluation equation*

$$q(s, a) = r(s, a) - \bar{r} + \sum_{s'} p(s' \mid s, a) \sum_{a'} \pi(a' \mid s') q(s', a'), \tag{3.5}$$

for all $s \in \mathcal{S}, a \in \mathcal{A}$, where

$$r(s, a) \doteq \sum_{s', r} p(s', r \mid s, a) r \tag{3.6}$$

is the one step reward, has $r(\pi)$ as its unique solution of $\bar{r}$ and any solution of $q : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ satisfies $q = q_\pi + c\mathbf{1}$ for some constant c.

Tabular *control* learning and planning algorithms seek a policy that achieves the "best reward rate". For an unconstrained continuing MDP, the best reward rate depends on the start state. For example, the MDP may have two disjoint sets of states with no policy that passes from one to the other; in this case, there are effectively two MDPs. A learning algorithm would have no difficulty with such cases—it would optimize for whichever sub-MDP it found itself in—but it is complex to state formally what is meant by an optimal policy. To remove this complexity, it is commonplace to rule out such cases by assuming that the MDP is weakly communicating.

Under the weakly communicating assumption, there exists a unique optimal reward rate r_* that does not depend on the start state.

$$r_* \doteq \sup_{\pi \in \Pi} r(\pi, s),$$

for all $s \in \mathcal{S}$. A control algorithm seeks a policy π such that $r(\pi, s) = r_*$ for all $s \in \mathcal{S}$. A common way to find such a policy is to solve the *action-value optimality equation*:

$$q(s, a) = r(s, a) - \bar{r} + \sum_{s'} p(s' \mid s, a) \max_{a'} q(s', a'), \tag{3.7}$$

for all $s \in \mathcal{S}$ and $a \in \mathcal{A}$. The unique solution of $\bar{r}$ is r_*. The solution set of q has multiple degrees of freedom (Schweitzer and Federgruen 1978) but any greedy policy w.r.t. any solution of q is a deterministic optimal policy (an optimal policy that, at each state, chooses an action w.p. one). Denote the set of deterministic optimal policies as Π_*^D.

3.2 The Convergence of a General Algorithm

This section introduces a new technical result that plays a central role in the convergence proofs of many of the algorithms introduced in the current and the next chapters. Specifically, this result shows the convergence of a general algorithm, which encompasses these algorithms as special cases. The convergence of these algorithms follows as a result of this general convergence result. This general algorithm is called *General RVI Q* because it is a generalization of an algorithm called RVI Q-learning by Abounadi, Bertsekas, and Borkar (2001). The convergence result of General RVI Q is also a generalization of the convergence result of the RVI Q-learning algorithm shown by Abounadi, Bertsekas, and Borkar (2001). Readers who are not intended to read the convergence proofs of algorithms introduced in the current and next chapter may choose to skip this section.

General RVI Q is a stochastic approximation algorithm that finds a solution of the following equation

$$r(i) - \bar{r} + g(q)(i) - q(i) = 0, \forall\, i \in \mathcal{I}, \tag{3.8}$$

where $\mathcal{I} = \{1, \dots, d\}$, $\bar{r} \in \mathbb{R}$ and $q \in \mathbb{R}^d$ are unknown variables to be solved, $r \in \mathbb{R}^d$ is any fixed d-dim vector, $g : \mathbb{R}^d \to \mathbb{R}^d$ is a function satisfying the following assumption.

Assumption 3.1 (Conditions on g)**.**
(i) g is a max-norm non-expansion
(ii) g is a span-norm non-expansion
(iii) $g(x + c\mathbf{1}) = g(x) + c\mathbf{1}$ for any $c \in \mathbb{R}, x \in \mathbb{R}^d$
(iv) $g(cx) = cg(x)$ for any $c \geq 1, x \in \mathbb{R}^d$.

We further make the following conditions on the solution set of (3.8).

Assumption 3.2 (Condition on the solution set of (3.8))**.**
Equation 3.8 has a unique solution of $\bar{r}$ and has at least one solution of q.

Denote the unique solution of $\bar{r}$ as $r_{\#}$. It is clear that if q is a solution of (3.8), then $q + c\mathbf{1}$ is also a solution for any $c \in \mathbb{R}$. Therefore Assumption 3.2 implies that there are an infinite number of solutions of q in (3.8).

I now introduce the General RVI Q algorithm. This algorithm maintains a d-dim vector of estimates $Q \in \mathbb{R}^d$, and updates Q using

$$\begin{aligned} &Q_{n+1}(i) \doteq Q_n(i) \\ &+ \alpha_{\nu_n(i)}\big(r(i) - f(Q_n) + g(Q_n)(i) - Q_n(i) + M_{n+1}(i) + \epsilon_{n+1}(i)\big)\mathbf{I}\{i \in Y_n\}, \end{aligned} \tag{3.9}$$

where $Q_n = [Q_n(1), \ldots, Q_n(d)]^\top \in \mathbb{R}^d$, $\{Y_n\}$ is a random process taking values in the set of non-empty subsets of $\mathcal{I}$, $\nu_n(i) \doteq \sum_{k=0}^{n} \mathbf{I}\{i \in Y_k\}$, where $\mathbf{I}$ is the indicator function (i.e., $\nu_n(i)$ = the number of times the i component was updated up to and including step n), $f : \mathbb{R}^d \to \mathbb{R}$ is a function, $\{\alpha_n\}$ is a sequence of step sizes, M_{n+1} and ϵ_{n+1} are two sequences of noise terms.

Assumption 3.3 (Conditions on f)**.**
(i) f is L-Lipschitz for some $L \in \mathbb{R}$,
(ii) there exists a positive scalar u s.t. $f(x + c\mathbf{1}) = f(x) + cu$ for any $c \in \mathbb{R}$ and $x \in \mathbb{R}^d$ and,
(iii) $f(cx) - f(\mathbf{0}) = c(f(x) - f(\mathbf{0}))$ for any $c \geq 1$ and $x \in \mathbb{R}^d$.

Examples of functions that satisfy the above assumption include $f(x) = a^\top x + b$ for any $a \in \mathbb{R}^d$ such that $a^\top \mathbf{1} > 0$, $b \in \mathbb{R}$, $f(x) = a \cdot \max_i x(i) + b$ for any $a > 0$, $b \in \mathbb{R}$.

The $f(Q_n)$ term in the General RVI Q algorithm is designed to solve for $\bar{r}$ in (3.8). Denote $\mathcal{Q}_\#$ as the set of solutions of q in (3.8) and

$$f(q) = r_\#. \tag{3.10}$$

Lemma 3.1. *Under Assumptions 3.1, 3.2, 3.3, $\mathcal{Q}_\#$ is non-empty and closed.*

Proof. $\mathcal{Q}_\#$ is not empty by Assumption 3.3(ii) and Assumption 3.2.

To show the closeness of $\mathcal{Q}_\#$, first note that the set of solutions of q in (3.8) is closed, because $g(x)-x$ is continuous by Assumption 3.1(i) and the preimage of a continuous function on a closed set is closed. Similarly, the set of solutions of q in (3.10) is closed, because $f(x)$ is continuous by Assumption 3.3(i) and the preimage of a continuous function on a closed set is closed. $\mathcal{Q}_\#$, being the intersection of two closed sets, is therefore closed. □

I will show that Q_n converges to $\mathcal{Q}_\#$. But before that, I need to first finish the presentation of other assumptions. The following assumption will be used to show the stability of General RVI Q.

Assumption 3.4 (Condition on $\mathcal{Q}_\#$)**.** If in (3.8), $r(i) = f(\mathbf{0})$ for all i, $\mathbf{0}$ is the only member of $\mathcal{Q}_\#$.

The following assumption is for the steps-size sequence $\{\alpha_n\}$.

Assumption 3.5 (Step-size conditions)**.**
(i) For $n \geq 0$, $\alpha_n > 0$. $\alpha_{n+1} \leq \alpha_n$ for all n sufficiently large. $\sum_{n=0}^{\infty} \alpha_n = \infty$, and $\sum_{n=0}^{\infty} \alpha_n^2 < \infty$.
(ii) Let $[\cdot]$ denote the integer part of $(\cdot)$, for $x \in (0,1)$,

$$\sup_n \frac{\alpha_{[xn]}}{\alpha_n} < \infty$$

and as $n \to \infty$

$$\frac{\sum_{n=0}^{[ym]} \alpha_n}{\sum_{n=0}^{m} \alpha_n} \to 1 \qquad \text{uniformly in } y \in [x, 1].$$

Assumption 3.6 (Update schedule)**.** The following statements hold:
(i) there exists a deterministic $\Delta > 0$ such that

$$\liminf_{n\to\infty} \frac{\nu_n(i)}{n} \geq \Delta,$$

a.s., for all $i \in \mathcal{I}$.
(ii) for all $x > 0$, let

$$N(n, x) = \min\left\{ m > n : \sum_{k=n}^{m} \alpha_k \geq x \right\},$$

the limit

$$\lim_{n\to\infty} \frac{\sum_{k=\nu_n(i)}^{\nu_{N(n,x)}(i)} \alpha_k}{\sum_{k=\nu_n(i')}^{\nu_{N(n,x)}(i')} \alpha_k} \qquad \text{exists a.s. for all } i, i' \in \mathcal{I}.$$

Remark: Assumptions 3.5 and 3.6 originate from a result showing the convergence of stochastic approximation algorithms (Borkar 1998;2000) and were also required by the convergence theorem of RVI Q-learning. Roughly speaking, Assumptions 3.5 and 3.6 mean that the step-size sequence $\{\alpha_n\}$ decreases appropriately and all states are updated *comparably often* in an *evenly distributed* manner. Assumptions 3.5 can be satisfied if the sequence $\{\alpha_n\}$ decreases to 0 appropriately. The sequence $\{\alpha_n\}$ could be, for example, $1/n$, $1/n \log n$, or $\log n/n$ for $n \geq 2$ (Abounadi, Bertsekas, and Borkar 2001). The first part of Assumption 3.6 requires that the fraction of updates to any element in Q is greater than or equal to any fixed positive number in the limit. Therefore relative update frequency between any two elements can not go infinite. The second part of the assumption is more sophisticated. It puts some restriction on the step-size sequence $\{\alpha_n\}_{n=0}^{\infty}$ and the visitations $\{\nu_n(i)\}, \forall i \in \mathcal{I}$. Certain step-size schedules satisfy the second part with a very flexible update schedule. The next example shows that, with a common step-size sequence $\alpha_n = 1/n$, Assumption 3.6(ii) can be satisfied (Assumption 3.5 can also be satisfied with this step-size sequence), as long as Assumption 3.6(i) is satisfied and $\lim_{n\to\infty} \nu_n(i)/n$ exists for all $i \in \mathcal{I}$, which means the fractions don't change in the limit.

Example 3.1. To see that Assumption 3.6(ii) holds with $\alpha_n = 1/n$, note that

$$\lim_{n\to\infty} \frac{\sum_{k=\nu_n(i)}^{\nu_{N(n,x)}(i)} \alpha_k}{\sum_{k=\nu_n(i')}^{\nu_{N(n,x)}(i')} \alpha_k} = \frac{\sum_{k=1}^{\nu_{N(n,x)}(i)} \alpha_k - \sum_{k=1}^{\nu_n(i)-1} \alpha_k}{\sum_{k=1}^{\nu_{N(n,x)}(i')} \alpha_k - \sum_{k=1}^{\nu_n(i')-1} \alpha_k}$$
$$= \lim_{n\to\infty} \frac{\log\big(\nu_{N(n,x)}(i)\big) + \gamma - \log(\nu_n(i) - 1) - \gamma}{\log\big(\nu_{N(n,x)}(i')\big) + \gamma - \log(\nu_n(i') - 1) - \gamma}.$$

The second equation uses the fact that $\lim_{n\to\infty} \sum_{k=1}^{n} \frac{1}{k} - \log n = \gamma$, where $\gamma \approx 0.57721$ is the Euler's constant. The r.h.s. of the second equation can be further written as follows.

$$\text{r.h.s.} = \lim_{n\to\infty} \frac{\log \frac{\nu_{N(n,x)}(i)}{\nu_n(i)-1}}{\log \frac{\nu_{N(n,x)}(i')}{\nu_n(i')-1}} = \lim_{n\to\infty} \frac{\log \frac{\nu_n(i)-1}{\nu_{N(n,x)}(i)}}{\log \frac{\nu_n(i')-1}{\nu_{N(n,x)}(i')}}$$
$$= \lim_{n\to\infty} \frac{\log \frac{\nu_n(i)}{n} + \log \frac{\nu_n(i)-1}{\nu_n(i)} - \log \frac{\nu_{N(n,x)}(i)}{N(n,x)} - \log \frac{N(n,x)}{n}}{\log \frac{\nu_n(i')}{n} + \log \frac{\nu_n(i')-1}{\nu_n(i')} - \log \frac{\nu_{N(n,x)}(i')}{N(n,x)} - \log \frac{N(n,x)}{n}} \tag{3.11}$$

Note that $\lim_{n\to\infty} \log \frac{\nu_n(i)-1}{\nu_n(i)} = 0$. Also, note that $N(n,x)$ is the minimum integer that satisfies $\sum_{k=1}^{N(n,x)} \frac{1}{k} - \sum_{k=1}^{n} \frac{1}{k} \geq x$ by definition. When $n \to \infty$,

$N(n,x)$ is the minimum integer that satisfies $\log N(n,x) - \log n \geq x$. Thus $N(n,x) \approx ne^x$. Therefore $\log \frac{N(n,x)}{n} \approx \log \frac{ne^x}{n} = x > 0$.

If $\lim_{n\to\infty} \frac{\nu_n(i)}{n}$ exists, $\lim_{n\to\infty} \log \frac{\nu_n(i)}{n} = \lim_{n\to\infty} \log \frac{\nu_{N(n,x)}(i)}{N(n,x)}$. Therefore $\log \frac{N(n,x)}{n} \approx x > 0$ dominates both the numerator and the denominator of (3.11). Therefore (3.11) $= \lim_{n\to\infty} \frac{\log \frac{N(n,x)}{n}}{\log \frac{N(n,x)}{n}} = 1$. Therefore Assumption 3.6(ii) holds.

On the other hand, if $\lim_{n\to\infty} \frac{\nu_n(i)}{n}$ does not exist, $\lim_{n\to\infty} \log \frac{\nu_n(i)}{n} - \log \frac{\nu_{N(n,x)}(i)}{N(n,x)}$ does not necessarily exist. Therefore (3.11) may not exist.

The example is finished.

Let $\mathcal{F}_n \doteq \sigma(x_m, Y_m, M_m, \epsilon_m; m \leq n)$ be an increasing family of σ-fields.

Assumption 3.7 (Conditions on the noise terms)**.** For some deterministic constant $K \geq 0$,
(i) $\mathbb{E}[M_{n+1} \mid \mathcal{F}_n] = 0$, $\mathbb{E}[\|M_{n+1}\|^2 \mid \mathcal{F}_n] \leq K(1 + \|Q_n\|^2)$ a.s., for all $n \geq 0$
(ii) as $n \to \infty$, $\epsilon_n \to \mathbf{0}$ a.s.

Theorem 3.1. *If Assumptions 3.1–3.7 hold, then Q_n converges almost surely to $\mathcal{Q}_\#$ and $f(Q_n)$ converges almost surely to $r_\#$.*

Remark: To prove Theorem 3.1, I will use a result by Theorem 2.5 by Borkar and Meyn (2000). This result is a key result used to show the convergence of RVI Q-learning (Abounadi, Bertsekas, and Borkar 2001) and the convergence of Differential Q-learning (Wan, Naik, Sutton 2021a). However, Borkar and Meyn (2000) only provided a proof sketch of Theorem 2.5 rather than the details of the proof. They suggested that the proof, which concerns a family of asynchronous algorithms, would be similar to their proof concerning a corresponding family of synchronous algorithms. Recently, Huizhen Yu and I tried to complete the details of the proof of Theorem 2.5, but we found that we can not use similar arguments from their synchronous proof to establish the asynchronous result. Huizhen Yu proved a result that is slightly different from Theorem 2.5. Unlike Theorem 2.5, Yu's result does not require that the o.d.e. associated with the update has a unique globally asymptotically stable equilibrium, assumes that there is no delay in updates, and allows the inclusion of an additional noise that decreases to zero almost surely. Note that

Borkar (Section 4, Chapter 7; Theorem 2, Chapter 2; 2009) also presented a similar result without assuming the unique globally asymptotically stable equilibrium. However, just as in Borkar (1998), he didn't provide proof of the stability of the algorithm. Borkar (2009) also suggested, without proof, that including an additional noise in both synchronous and asynchronous updates would not change the convergence of the updates. However, in his comment, the additional noise is bounded while Yu's noise does not have to be bounded. Removing the assumption of a unique globally asymptotically stable equilibrium is needed for my purpose of showing control algorithms' convergence in weakly communicating MDPs. The delay allowed by Theorem 2.5 is not needed for RVI Q-learning or any algorithm presented in this dissertation. The additional noise term (without boundedness condition) will be used to show the convergence of centered algorithms (Section 3.7) and inter-option algorithms (Section 4.5).

I now present Yu's result and use it to show the convergence of General RVI Q. The proof of Yu's result will be shown in our upcoming paper (Wan, Yu, Sutton 2023). The concerned asynchronous stochastic approximation algorithm's update rule is:

$$x_{n+1}(i) = x_n(i) + \alpha_{\nu_n(i)} \left(h_i(x_n) + M_{n+1}(i) + \epsilon_{n+1}(i)\right) \mathbf{I}\{i \in Y_n\}, \ i \in \mathcal{I}, \quad (3.12)$$

where $x_n = [x_n(1), \ldots, x_n(d)]^\top \in \mathbb{R}^d$, $\alpha_n, Y_n, \nu_n, M_{n+1}$, and ϵ_{n+1} are defined just as for General RVI Q (3.8) and satisfy Assumptions 3.5, 3.6, 3.7, $h : \mathbb{R}^d \to \mathbb{R}^d$ satisfies the following conditions.

Assumption 3.8 (Conditions on the function h)**.**

(i) Lipschitz continuity: for some $0 \le L < \infty$, $\|h(x) - h(y)\| \le L\|x - y\|$ for all $x, y \in \mathbb{R}^d$.

(ii) For $c \ge 1$ and functions $h_c(x) \doteq h(cx)/c$, we have $h_c(x) \to h_\infty(x)$ as $c \to \infty$, uniformly on compact subsets of $\mathbb{R}^d$, where h_∞ is a continuous function on $\mathbb{R}^d$. Furthermore, the o.d.e.

$$\dot{x}(t) = h_\infty(x(t)) \quad (3.13)$$

has the origin as its unique globally asymptotically stable equilibrium.

Theorem 3.2. *Under Assumptions 3.5, 3.6, 3.7, 3.8, then the sequence* $\{x_n\}$ *generated by* (3.12) *converges a.s. to a (possibly sample path-dependent) compact, connected, internally chain transitive, invariant set of the o.d.e.* $\dot{x}(t) = h(x(t))$.

Proof of Theorem 3.1

By choosing h such that $h(q)(i) \doteq r(i) - f(q) + g(q)(i) - q(i)$ for all $i \in \mathcal{I}$ and $q \in \mathbb{R}^d$, we see that (3.9) is in the same form of (3.12). We will first show, using a series of lemmas, any compact invariant set of the o.d.e. $\dot{x}(t) = h(x(t))$ is a subset of $\mathcal{Q}_\#$. Then, we will verify conditions required by Theorem 3.2 to show Theorem 3.1.

Define operators T_1, T_2:

$$T_1(Q)(i) \doteq r(i) + g(Q)(i) - r_\#,$$
$$T_2(Q)(i) \doteq r(i) + g(Q)(i) - f(Q) = T_1(Q)(i) + (r_\# - f(Q)).$$

Consider two ordinary differential equations:

$$\dot{y}_t \doteq T_1(y_t) - y_t, \tag{3.14}$$
$$\dot{x}_t \doteq T_2(x_t) - x_t = T_1(x_t) - x_t + (r_\# - f(x_t))\,\mathbf{1}. \tag{3.15}$$

Note that because g is a non-expansion by Assumption 3.1(i) and f is Lipschitz continuous by Assumption 3.3(i), both (3.14) and (3.15) have Lipschitz r.h.s. and thus are well-posed.

Because g is a non-expansion, T_1 is also a non-expansion. Also, (3.14) has an equilibrium by Assumption 3.2. Therefore we have the next lemma, which restates Theorem 3.1 and Lemma 3.2 by Borkar and Soumyanatha (1997).

Lemma 3.2. *Let* $\bar{y}$ *be an equilibrium point of* (3.14). *Then* $\|y_t - \bar{y}\|_\infty$ *is nonincreasing, and* $y_t \to y_\infty$ *for some equilibrium point* y_∞ *of* (3.14) *that may depend on* y_0.

The next three lemmas extend Lemmas 3.2, 3.3 by Abounadi, Bertsekas, and Borkar (2001) to the case where $\mathcal{Q}_\#$ is a set instead of a point and there is a weaker condition on f (Assumption 3.3).

Lemma 3.3. *The set of equilibrium points of* (3.15) *is* $\mathcal{Q}_\#$.

Proof. $\mathcal{Q}_\#$ is the set of fixed points of T_1. Then for any $q \in \mathcal{Q}_\#$, $T_2(q) = T_1(q) + (r_\# - f(q)) = T_1(q) = q$. Thus q is an equilibrium point of (3.15). Conversely, if q is an equilibrium of (3.15), then $q = T_2(q) = T_1(q) + (r_\# - f(q))\mathbf{1}$. But the equation $q = T_1(q) + c\mathbf{1}$ has a solution if and only if $c = 0$ because of Assumption 3.2. Thus $q = T_1(q)$, implying $q \in \mathcal{Q}_\#$. □

Lemma 3.4. *Let* $x_0 = y_0$*, then* $x_t = y_t + z_t\mathbf{1}$*, where* z_t *satisfies the ODE* $\dot{z}_t = -uz_t + (r_\# - f(y_t))$.

Proof. I first show $x_t = y_t + z_t\mathbf{1}$. From (3.14) and (3.15), by the variation of parameters formula,

$$x_t = \exp(-t)x_0 + \int_0^t \exp(\tau - t)T_1(x_\tau)d\tau + \left[\int_0^t \exp(\tau - t)\left(r_\# - f(x_\tau)\right)d\tau\right]\mathbf{1},$$
$$y_t = \exp(-t)y_0 + \int_0^t \exp(\tau - t)T_1(y_\tau)d\tau.$$

Then we have

$$\begin{aligned}\max_i(x_t(i) - y_t(i)) &\le \int_0^t \exp(\tau - t)\max_i(T_1(x_\tau)(i) - T_1(y_\tau)(i))d\tau \\ &\quad + \left[\int_0^t \exp(\tau - t)\left(r_\# - f(x_\tau)\right)d\tau\right], \\ \min_i(x_t(i) - y_t(i)) &\ge \int_0^t \exp(\tau - t)\min_i(T_1(x_\tau)(i) - T_1(y_\tau)(i))d\tau \\ &\quad + \left[\int_0^t \exp(\tau - t)\left(r_\# - f(x_\tau)\right)d\tau\right].\end{aligned}$$

Subtracting, we have

$$sp(x_t - y_t) \le \int_0^t \exp(\tau - t)sp(T_1(x_\tau) - T_1(y_\tau))d\tau,$$

where $sp(x)$ denotes the span of vector x.

Because g is a span-norm non-expansion (Assumption 3.1(ii)), T_1 is also a span-norm non-expansion. Thus,

$$\int_0^t \exp(\tau - t)sp(T_1(x_\tau) - T_1(y_\tau))d\tau \le \int_0^t \exp(\tau - t)sp(x_\tau - y_\tau)d\tau.$$

Recall Gronwall's inequality, which states that for continuous real-valued functions $u(t), v(t)$ defined on $[0, \infty)$ for some positive T, if v is non-negative, and for $c \in \mathbb{R}$,

$$u(t) \leq c + \int_0^t u(s)v(s)ds \ \forall t \in [0, \infty),$$

then

$$u(t) \leq c \exp\left(\int_0^t v(s)ds\right) \ \forall t \in [0, \infty).$$

Applying Gronwall's inequality to

$$sp(x_t - y_t) \leq \int_0^t \exp(\tau - t) sp(x_\tau - y_\tau) d\tau,$$

we have $sp(x_t - y_t) = 0$ for all $t \geq 0$. Therefore $x_t = y_t + z_t\mathbf{1}, t \geq 0$ for some z_t. Also $x_0 = y_0 \implies z_0 = 0$.

Now we show that $\dot{z}_t = -uz_t + (r_\# - f(y_t))$. Note that $f(x_t) = f(y_t + z_t\mathbf{1}) = f(y_t) + uz_t$ where the second equation holds because of Assumption 3.3 (ii). In addition, $T_1(x_t) - T_1(y_t) = T_1(y_t + z_t\mathbf{1}) - T_1(y_t) = T_1(y_t) + z_t\mathbf{1} - T_1(y_t) = z_t\mathbf{1}$, where the second equation holds because of Assumption 3.1(iii). Therefore,

$$\begin{aligned}
&\dot{z}_t\mathbf{1} \\
&= \dot{x}_t - \dot{y}_t \\
&= (T_1(x_t) - x_t + (r_\# - f(x_t))\,\mathbf{1}) - (T_1(y_t) - y_t) \quad \text{(Equations 3.14 and 3.15)} \\
&= -(x_t - y_t) + (T_1(x_t) - T_1(y_t)) + (r_\# - f(x_t))\,\mathbf{1} \\
&= -z_t\mathbf{1} + z_t\mathbf{1} + (r_\# - f(x_t))\,\mathbf{1} \\
&= -uz_t\mathbf{1} + uz_t\mathbf{1} + (r_\# - f(x_t))\,\mathbf{1} \\
&= -uz_t\mathbf{1} + (r_\# - f(y_t))\,\mathbf{1} \\
&\implies \dot{z}_t = -uz_t + (r_\# - f(y_t))\,.
\end{aligned}$$

□

Lemma 3.5. *Let $x_0 = y_0$, then $\lim_{t\to\infty} x_t = y_\infty + (r_\# - f(y_\infty))\mathbf{1}/u$, which is an member of $\mathcal{Q}_\#$.*

Proof. By the variation of parameters formula and noting that $z_0 = x_0 - y_0 = \mathbf{0}$,

$$z_t = \int_0^t \exp(u\tau - ut)(r_\# - f(y_\tau))d\tau.$$

Because $y_t \to y_\infty$ by Lemma 3.2 and f is continuous by Assumption 3.3(i), the above equation can be rewritten as

$$z_t = \int_0^t \exp(u\tau - ut)(r_\# - f(y_\infty))d\tau + \int_0^t \exp(u\tau - ut)o(\tau)d\tau,$$

where $o(\tau)$ is a sequence of scalars that converges to 0 as $\tau \to \infty$. Thererfore, for every $\epsilon > 0$, there exists a $T > 0$ such that $\|o(t)\| < \epsilon$ for all $t > T$. Fix an $\epsilon > 0$, for any $t > T$, we have

$$\begin{aligned}
&\left\|\int_0^t \exp(u\tau - ut)o(\tau)d\tau\right\| \\
&= \left\|\int_0^T \exp(u\tau - ut)o(\tau)d\tau + \int_T^t \exp(u\tau - ut)o(\tau)d\tau\right\| \\
&\leq \int_0^T \exp(u\tau - ut)\|o(\tau)\|d\tau + \int_T^t \exp(u\tau - ut)\|o(\tau)\|d\tau \\
&\leq \epsilon\int_0^T \exp(u\tau - ut)o(\tau)d\tau + \int_T^t \exp(u\tau - ut)d\tau \\
&= \int_0^T \exp(u\tau - ut)o(\tau)d\tau + \epsilon\frac{1 - \exp(uT - ut)}{u}.
\end{aligned}$$

As $t \to \infty$, the r.h.s. converges to ϵ/u. Because the choice of ϵ can be made arbitrarily small, we have $\left\|\int_0^t \exp(u\tau - ut)o(\tau)d\tau\right\| \to 0$ as $t \to \infty$. Using this result and $\lim_{t\to\infty} \int_0^t \exp(u\tau - ut)(r_\# - f(y_\infty))d\tau = (r_\# - f(y_\infty))/u$, we have $z_t \to (r_\# - f(y_\infty))/u$. Combining this result with $x_t = y_t + z_t\mathbf{1}$ (Lemma 3.4) and Lemma 3.2, we have $x_t \to y_\infty + (r_\# - f(y_\infty))\mathbf{1}/u$ as $t \to \infty$. Finally, because the set of equilibrium points T_2 is $\mathcal{Q}_\#$, x_t must converge to one of such points. □

For $\epsilon > 0$, denote by $\mathcal{Q}_\#^\epsilon$ the closed ϵ-neighborhood of $\mathcal{Q}_\#$ w.r.t. $\|\cdot\|_\infty$.

Lemma 3.6. *For the o.d.e.* (3.15), *$\mathcal{Q}_\#$ is Lyapunov stable in the sense that given any $\epsilon > 0$, there exists $\delta > 0$ such that for all initial conditions $x_0 \in \mathcal{Q}_\#^\delta$, $x_t \in \mathcal{Q}_\#^\epsilon$ for all $t \geq 0$.*

Proof. Let $y_0 = x_0$, then $z_0 = 0$, and by Lemma 3.4 we have $\dot{z}_t = -uz_t + (r_\# - f(y_t))$. By variation of parameters and $z_0 = 0$, we have

$$z_t = \int_0^t \exp(u(\tau - t))\,(r_\# - f(y_\tau))\,d\tau.$$

Choose any $q_* \in \mathcal{Q}_\#$,

$$\begin{aligned}
&\|q_* - x_t\|_\infty \\
&= \|q_* - (y_t + z_t u\mathbf{1})\|_\infty \\
&\leq \|q_* - y_t\|_\infty + u|z_t| \\
&\leq \|q_* - y_0\|_\infty + u\int_0^t \exp(u(\tau - t))|r_\# - f(y_\tau)|d\tau && \text{(Lemma 3.2)} \\
&= \|q_* - x_0\|_\infty + u\int_0^t \exp(u(\tau - t))|f(q_*) - f(y_\tau)|d\tau, && (3.16)
\end{aligned}$$

where the last equality holds by (3.10).

Because f is L-Lipschitz by Assumption 3.3(i), we have

$$\begin{aligned}
|f(q_*) - f(y_\tau)| \leq L\|q_* - y_\tau\|_\infty &\leq L\|q_* - y_0\|_\infty && \text{(Lemma 3.2)} \\
&= L\|q_* - x_0\|_\infty.
\end{aligned}$$

Therefore,

$$\begin{aligned}
&\int_0^t \exp(u(\tau - t))|f(q_*) - f(y_\tau)|d\tau \leq \int_0^t \exp(u(\tau - t))L\|q_* - x_0\|_\infty d\tau \\
&= L\|q_* - x_0\|_\infty \int_0^t \exp(u(\tau - t))d\tau \\
&= \frac{L}{u}\|q_* - x_0\|_\infty(1 - \exp(-ut)).
\end{aligned}$$

Substituting the above equation in (3.16), we have

$$\|q_* - x_t\|_\infty \leq (1 + L)\|q_* - x_0\|_\infty.$$

Finally, for any $\epsilon > 0$, choose $\delta = \epsilon/(1 + L)$, and for any $x_0 \in Q_\#^\delta$, choose $q_* \in \mathcal{Q}_\#$ within the δ-neightborhood of x_0, then by the above inequality $\|x_t - q_*\| \leq \epsilon$. Lyapunov stability follows. □

Lemma 3.7. *Any compact invariant set of the o.d.e.* (3.15) *is contained in* $\mathcal{Q}_\#$.

Proof. We use proof by contradiction. Suppose A is a compact invariant set of (3.15) and $A \not\subset \mathcal{Q}_{\#}$. Let $d_{A,\mathcal{Q}_{\#}} \doteq \sup_{x\in A}\inf_{y\in\mathcal{Q}_{\#}} \|x-y\|_\infty$ we have $d_{A,\mathcal{Q}_{\#}} > 0$ by the compactness of A and the closedness of $\mathcal{Q}_{\#}$ by Lemma 3.1, and that $A \not\subset \mathcal{Q}_{\#}$. Let $0 < \epsilon < d_{A,\mathcal{Q}_{\#}}$, and let δ be given by Lemma 3.6 for this ϵ.

Let $\phi(t;x)$ be the solution of the o.d.e. (3.15) with $x(0) = x$. Since $\phi(t;x)$ converges to an element in $\mathcal{Q}_{\#}$ as $t \to \infty$ (Lemma 3.5), there is a time t_x such that $\phi(t_x;x) \in \mathcal{Q}_{\#}^{\delta/2}$. Since h is Lipschitz continuous, $\phi(t;x)$ is continuous in x, so there is an open neighborhood D_x of x such that

$$\phi(t_x;y) \in \mathcal{Q}_{\#}^{\delta}, \qquad \forall\, y \in D_x. \tag{3.17}$$

Now $D_x, x \in A$, form an open cover of the compact set A, so there exist a finite number of points $x^1, x^2, \ldots, x^\ell \in A$ with $A \subset \cup_{i=1}^{\ell} D_{x_i}$. Let $\bar{t} = \max_{1\le i\le \ell} t_{x_i}$. Then by (3.17) and Lemma 3.6,

$$\phi(t;x) \in \mathcal{Q}_{\#}^{\epsilon}, \qquad \forall\, x \in A,\ t \ge \bar{t}. \tag{3.18}$$

With A being invariant for the o.d.e. (3.15), $\{\phi(\bar{t};x) \mid x \in A\} = A$, so (3.18) implies that $A \subset \mathcal{Q}_{\#}^{\epsilon}$, contradicting the fact $d_{A,\mathcal{Q}_{\#}} > \epsilon$. The proof is now complete. □

We now verify assumptions required by Theorem 3.2. First, note that Assumptions 3.5, 3.6, 3.7 are also required by Theorem 3.1 and do not need to be verified. Part (i) of Assumption 3.8 can be easily verified because both f and g are Lipschitz by Assumptions 3.3, 3.1. Thus we only need to verify part (ii) of Assumption 3.8.

Note that for any $c \ge 1$

$$\begin{aligned} h_c(x) &= \left(r - f(cx)\mathbf{1} + g(cx) - cx\right)/c \\ &= \left(r - (f(cx) - f(\mathbf{0}))\mathbf{1} - f(\mathbf{0})\mathbf{1} + g(cx) - cx\right)/c \\ &= \left(r - c(f(x) - f(\mathbf{0}))\mathbf{1} - f(\mathbf{0})\mathbf{1} + cg(x) - cx\right)/c \\ &= r/c - (f(x) - f(\mathbf{0}))\mathbf{1} - f(\mathbf{0})\mathbf{1}/c + g(x) - x, \end{aligned}$$

where the second last equation holds because of Assumption 3.3(iii) and Assumption 3.1(iv). In addition, for every $x \in \mathbb{R}^d$,

$$h_\infty(x) = \lim_{c\to\infty} h_c(x) = f(\mathbf{0})\mathbf{1} - f(x)\mathbf{1} + g(x) - x.$$

The function h_∞ is continuous on $\mathbb{R}^d$ by the non-expansion assumption on g and the Lipschitz assumption on f. In addition, h_c converges to h_∞ as $c \to \infty$, uniformly on compact subsets of $\mathbb{R}^d$ because, for any $x \in \mathbb{R}^d$, and $c \geq 1$,

$$|h_c(x) - h_\infty(x)| = |r/c - f(\mathbf{0})\mathbf{1}/c|,$$

which is independent of the choice of x.

We now show that $\mathbf{0}$ is the unique globally asymptotically stable equilibrium of $\dot{x} = h_\infty(x) = f(\mathbf{0})\mathbf{1} - f(x)\mathbf{1} + g(x) - x$. Note that this is o.d.e. (3.15) with $r(i) = f(\mathbf{0})$. By Assumption 3.4, and Lemma 3.3, $\mathbf{0}$ is the unique equilibrium of this o.d.e. In addition, by Lemma 3.6, $\mathbf{0}$ is also stable. Finally, according to Lemma 3.5, for any $x_0 \in \mathbb{R}^d$, $\lim_{t\to\infty} x_t \in \mathcal{Q}_\#$, which must coincide with $\mathbf{0}$ because $\mathcal{Q}_\# = \{\mathbf{0}\}$. Thus $\mathbf{0}$ is the globally asymptotically stable equilibrium.

All the assumptions required by Theorem 3.1 are verified and the theorem is proved.

3.3 Prediction Algorithms

This section presents the new prediction learning and planning algorithms. Recall that the goal of prediction algorithms is to estimate the reward rate of a given policy and also estimate the given policy's differential value function up to an additive constant. Along with the two algorithms, this section also shows their convergence theories.

The prediction algorithm, called *Differential TD-learning*, updates a table of estimates $V_t : \mathcal{S} \to \mathbb{R}$, $t \geq 1$, as follows:

$$\begin{aligned} V_{t+1}(S_t) &\doteq V_t(S_t) + \alpha_{\nu_t(S_t)}\rho_t\delta_t, \qquad (3.19)\\ V_{t+1}(s) &\doteq V_t(s),\ \forall s \neq S_t, \end{aligned}$$

where $\{\alpha_n\}_{n\geq 1}$ is a step-size sequence satisfying Assumption 3.5, $\nu_t(S_t)$ is the number of times the state S_t was updated up to and including step t, $\{\alpha_n\}_{n\geq 1}$ and $\nu_t(S_t)$ jointly satisfy Assumption 3.6 (with $\mathcal{I} = \mathcal{S}$ and $n = t$), $\rho_t \doteq \pi(A_t \mid S_t)/b(A_t \mid S_t)$ is the importance-sampling ratio, and δ_t is the TD error at time step t:

$$\delta_t \doteq R_{t+1} - \bar{R}_t + V_t(S_{t+1}) - V_t(S_t), \tag{3.20}$$

where $\bar{R}_t$ is a scalar estimate of $r(\pi)$, updated by:

$$\bar{R}_{t+1} \doteq \bar{R}_t + \eta \alpha_{\nu_t(S_t)} \rho_t \delta_t, \tag{3.21}$$

and η is a positive constant. The next assumption is required to guarantee that the behavior policy covers all possible state-action pairs the target policy may incur. It guarantees that importance sampling ratios $\rho_t, t \geq 0$, are bounded.

Assumption 3.9. For all $s \in \mathcal{S}$, $a \in \mathcal{A}$, if $\pi(a \mid s) > 0$, then $b(a \mid s) > 0$.

The pseudocode of Differential TD-learning is shown in Algorithm 1.

Algorithm 1: Differential TD-learning

```
Input: The policy $\pi$ to be evaluated, and $b$ to be used
Algorithm parameters: step-size sequence $\alpha_n$, parameter $\eta$
Initialize $V(s)$, $\forall s$, $\bar{R}$ arbitrarily (e.g., to zero)
$\nu(s) \leftarrow 0 \; \forall s$
Obtain initial $S$
while still time to train do
    $\nu(S) \leftarrow \nu(S) + 1$
    $A \leftarrow$ action given by $b$ for $S$
    Take action $A$, observe $R, S'$
    $\delta \leftarrow R - \bar{R} + V(S') - V(S)$
    $\rho \leftarrow \pi(A \mid S) \,/\, b(A \mid S)$
    $V(S) \leftarrow V(S) + \alpha_{\nu(S)} \rho \delta$
    $\bar{R} \leftarrow \bar{R} + \eta \alpha_{\nu(S)} \rho \delta$
    $S \leftarrow S'$
end
return $V$
```

The theorem to be presented shows that $\bar{R}_t$ converges to $r(\pi)$ and V_t converges to v_∞, which is defined to be the unique solution of v in (3.2) and the following equation.

$$r(\pi) - \bar{R}_0 = \eta \left(\sum v - \sum V_0 \right). \tag{3.22}$$

The uniqueness of v_∞ can be seen as follows. Note that π is unichain, which implies that solutions of v in (3.2) are unique up to an additive constant. This consequence plus (3.22) shows that v_∞ is unique.

Theorem 3.3. *If the target policy π is unichain, Assumptions 3.5, 3.6 (with $\mathcal{I} = \mathcal{S}$), and 3.9 hold, then the Differential TD-learning algorithm* (3.19)–(3.21) *converges, almost surely, $\bar{R}_t$ to $r(\pi)$ and V_t to v_∞.*

Remark: Assumptions 3.5–3.6 used by the above theorem can be satisfied if, for example, the step-size sequence $\{\alpha_n\}$ is $\{1/n\}_{n\geq 1}$ and the behavior policy b visits all *states* an infinite number of times.

The proof of the theorem will be presented at the end of this section.

If Differential TD-learning is applied to a simulated experience generated from a model of the world, then it becomes a planning algorithm, which we call *Differential TD-planning.* Formally, the model is a function $\tilde{p} : \mathcal{S} \times \mathcal{A} \to \Delta(\mathcal{S} \times \mathcal{R})$, analogous to p, that, like p, sums to 1: $\sum_{s',r} \tilde{p}(s', r \mid s, a) = 1$ for all s, a. A model MDP can be thus constructed using $\tilde{p}$ and $\mathcal{S}, \mathcal{A}, \mathcal{R}, d_0$. If a policy π is unichain in the model MDP, then there is a unique reward rate and a unique solution of v in (3.2) and (3.22). Given the model MDP and a target policy π, let $\tilde{r}(\pi)$ and $\tilde{v}_\infty$ be defined in the same way as $r(\pi)$ and v_∞, except for $\tilde{p}$ rather than p.

The simulated transitions are generated as follows: at each planning step n, the agent arbitrarily chooses a state $\tilde{S}_n$ and action $\tilde{A}_n$ and applies $\tilde{p}$ to generate a simulated resulting state and reward $\tilde{S}'_n, \tilde{R}_n \sim \tilde{p}(\cdot, \cdot \mid S_n, A_n)$.

Like Differential TD-learning, Differential TD-planning maintains a table of value estimates $V_n : \mathcal{S} \to \mathbb{R}$ and a reward-rate estimate $\bar{R}_n$. At each planning step n, these estimates are updated by (3.19)–(3.21), just as in Differential TD-learning, except now using $\tilde{S}_n, \tilde{A}_n, \tilde{R}_n, \tilde{S}'_n$ instead of $S_t, A_t, R_{t+1}, S_{t+1}$.

Theorem 3.4. *Under the same assumptions made in Theorem 3.3 (except now for the model MDP corresponding to $\tilde{p}$ rather than p) the Differential TD-planning algorithm converges, almost surely, $\bar{R}_n$ to $\tilde{r}(\pi)$ and V_n to $\tilde{v}_\infty$.*

The proof of Theorem 3.4 is similar to that of Theorem 3.3 and is omitted.

The rest of this section proves Theorem 3.3.

We first transform the Differential TD-learning's update rules (3.19)–(3.21) into an equivalent update rule.

$$\begin{aligned}
&\bar{R}_t - \bar{R}_0 = \eta \sum_{i=0}^{t-1} \sum_s \alpha_{\nu_i(s)} \rho_i \delta_i \\
&= \eta \left(\sum V_t - \sum V_0 \right) \\
&\implies \\
&\bar{R}_t = \eta \sum V_t - \eta \sum V_0 + \bar{R}_0 = f(V_t) \\
&\text{where } f : \mathbb{R}^{|\mathcal{S}|} \to \mathbb{R} \text{ satisfying } \forall v \in \mathbb{R}^{|\mathcal{S}|}, f(v) \doteq \eta \sum v - \eta \sum V_0 + \bar{R}_0.
\end{aligned} \tag{3.23}$$

Substituting $\bar{R}_t$ in (3.19) with (3.22) we have, $\forall s \in \mathcal{S}$:

$$V_{t+1}(S_t) = V_t(S_t) + \alpha_{\nu_t(s)} \rho_t (R_{t+1} - f(V_t) + V_t(S_{t+1}) - V_t(S_t)). \tag{3.24}$$

We first show that (3.24) is a special case of the General RVI Q's update (3.9). To see this point, we hypothesize three $|\mathcal{S}|$-sized random processes $\{A'_t\}, \{S'_t\}, \{R'_t\}$ such that for any t, s, a, $A'_t(s) \sim b(\cdot \mid s), S'_t(s), R'_t(s) \sim p(\cdot, \cdot \mid s, A'_t(s))$. Now consider the stream of experience $\ldots, S_t, A_t, R_{t+1}, S_{t+1}, \ldots$. Then we have $A_t = A'_t(S_t), R_{t+1} = R'_t(S_t, A_t)$ and $S_{t+1} = S'_t(S_t, A_t)$. Equation 3.9 reduces to (3.24) by choosing $i = s, n = t, Q_n = V_t, Y_n = \{S_t\}$,

$$\begin{aligned}
&r(i) = r_\pi(s), \\
&g(v)(i) = \sum_a \pi(a \mid s) \sum_{s'} p(s' \mid s, a) v(s'), \\
&M_{t+1}(i) = \rho_t(s) \left(R'_t(s) - f(V_t) + V_t(S'_t(s)) - V_t(s) \right) \\
&\ - \left(r_\pi(s) - f(V_t) + g(V_t)(s) - V_t(s) \right), \\
&\epsilon_{t+1} = \mathbf{0},
\end{aligned}$$

where $\rho_t(s) \doteq \pi(A'_t(s) \mid s) / b(A'_t(s) \mid s)$.

We now show that the assumptions required by General RVI Q are all satisfied.

1. Assumption 3.1 can be verified for the choice of $g(v)$ easily.

2. Assumption 3.2 is satisfied. Note that (3.8) becomes (3.2) given our definition of r and g. Because the π is assumed to be unichain, $r_\# = r(\pi)$

is the only solution of $\bar{r}$ in (3.2). In addition, (3.2) has an infinite number of solutions of q by the discussion right after (3.2).

3. Assumptions 3.3 holds for f defined in (3.23).

4. Assumption 3.4 is satisfied. Note that $r(i) = r_\pi(s) = f(\mathbf{0})$ implies that $r_\# = r(\pi) = f(\mathbf{0})$. One can thus verify that $\mathbf{0}$ is a solution of q in (3.8) and (3.10), which reduce to (3.2) and (3.22) in the current setting. In addition, (3.2) and (3.22) have a unique solution of v (see the discussion right below (3.22)), implying that $\mathbf{0}$ is the unique solution.

5. Assumptions 3.5–3.6 are assumed in the theorem statement.

6. Assumption 3.7 can be verified by noting that state and reward spaces are finite, ρ_t is bounded by Assumption 3.9, and that $\epsilon_{t+1} = \mathbf{0}$.

Therefore all the conditions of Theorem 3.1 have been verified and we have $f(Q_n) = \bar{R}_t \to r_\#$, which is $r(\pi)$ in the current setting, and $Q_n = V_t$ converges to $\mathcal{Q}_\#$, which is $\{v_\infty\}$ in the current setting.

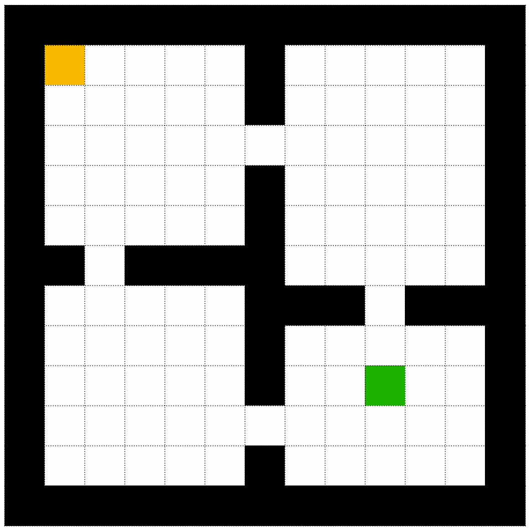


Figure 3.1: A continuing variant of the four-room domain.

3.4 Prediction Experiments

This section presents a set of experiments testing Differential TD-learning in a variant of Sutton, Precup, and Singh's (1999) four-room domain (shown in Figure 3.1). In this domain, the agent starts from the state indicated by the yellow cell. The rewarding state is indicated by the green cell. There are four primitive actions of moving `up`, `down`, `left`, `right`. If the agent takes any action in the rewarding state, it receives a $+1$ reward and moves to the start state. All other rewards are 0. The shortest path to the rewarding state from the start state takes 16 time steps, and there is an extra step moving from the rewarding state to the start state, hence the best possible reward rate for this task is $1/17 \approx 0.0588$.

The target policy π was a greedy policy w.r.t. a solution of q in (3.7). Therefore the target policy is optimal. Note that in this four-room domain, all optimal policies are unichain. Therefore solutions of v in (3.2), under any greedy policy, are only different by some additive constant. In this case, Differential TD-learning guarantees convergence to the differential value function up to some additive constant (Theorem 3.3). The behavior policy was an ϵ-greedy policy—it follows the target policy with probability $1 - \epsilon$ and chooses a random action with probability ϵ. Here ϵ was chosen to be 0.1.

To evaluate the estimated differential value function of Differential TD-learning at time step t, V_t, it is natural to consider the absolute difference between the estimated relative value and the true relative value as a measure of the quality of V_t. This absolute difference is termed *relative value error*. Formally, the relative value error at time step t is defined as follows,

$$\text{Relative Value Error}_t \doteq |(V_t(S_t) - V_t(s_0)) - (v_\pi(S_t) - v_\pi(s_0))|,$$

where s_0 was chosen to be the start state. The error of the estimated reward rate, or the *reward rate error*, measures the distance between $\bar{R}_t$ and the true reward rate $r(\pi)$:

$$\text{Reward Rate Error}_t \doteq |\bar{R}_t - r(\pi)|.$$

The step-size sequence α_t was chosen to be a constant sequence for simplicity. I tested five choices of $\alpha_t : 2^{-x}, x \in \{1, 3, 5, 7, 9\}$ and tested five choices of $\eta : 10^{-x}, x \in \{0, 1, 2, 3, 4\}$. V_0 and $\bar{R}_0$ were set to 0. Each experiment tested one parameter setting and was repeated for 30 times, each of which consists of $500,000$ steps. I recorded the reward rate error and relative value error for each step. Each point in Figure 3.2a or Figure 3.2b denotes one of the two errors, averaged over the last 2000 steps. The shaded region indicates one standard error. The parameter setting was chosen to be the one that resulted in the lowest reward rate error (relative value error) averaged over the last $10,000$ steps. Therefore the reported learning curves show the asymptotic performance of the algorithm rather than its speed of learning.

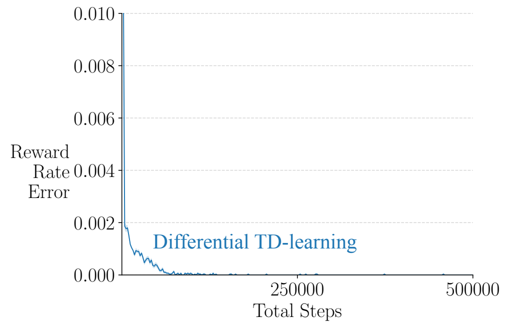
The parameter studies for the reward rate error and relative value error are shown in Figure 3.2c and Figure 3.2d, respectively. Each point in a curve in Figure 3.2c (Figure 3.2d) denotes the average of $500,000$ reward rate errors (relative value errors). The error bars denote one standard error (not visible because the error is too low).

The learning curves show that, asymptotically, the agent achieved a small reward rate and relative value errors. Parameter studies show that the average reward rate and relative value errors are almost zero when both α_t and η were chosen to be large, suggesting that learning is quite fast for these choices of α_t and η. In addition, error bars are not visible, suggesting that the variance of learning is quite low.

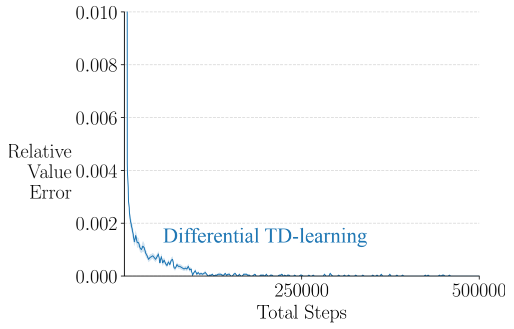
To conclude, our prediction experiments with the continuing four-room domain show that Differential TD-learning indeed learns the reward rate and the differential value function up to some additive constant, in accordance with Theorem 3.3. In addition, the algorithm learned fast and suffered from a low variance in this experiment, for large α_n and η.

3.5 Control Algorithms

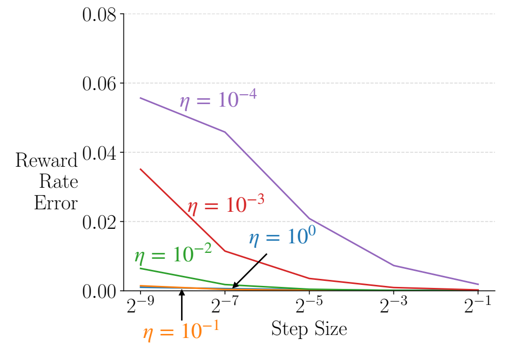
This section presents the new control learning and planning algorithms. Both two algorithms are designed to produce an optimal policy. This section



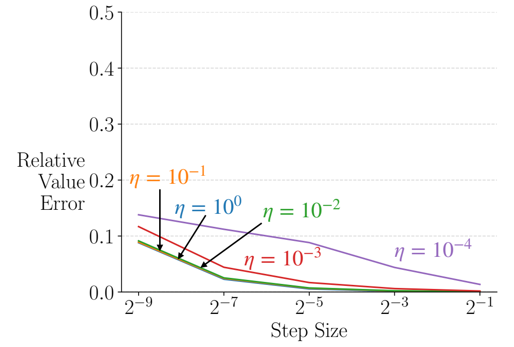
(a) A learning curve of the reward rate error.



(b) A learning curve of the relative value error.



(c) Parameter sensitivity curves of the reward rate error. Larger η and α generally perform better in the tested domain, potentially due to the deterministicity of the tested domain.



(d) Parameter sensitivity curves of the relative value error. Similar to parameter sensitivity curves of the reward rate error, larger η and α generally perform better in the tested domain.

Figure 3.2: Plots showing learning curves and parameter sensitivity curves for Differential TD-learning in the continuing four-room domain.

also presents the convergence results of the two algorithms. In addition, the technical tools used to obtain the convergence results also lead to an improved convergence result for RVI Q-learning (Abounadi, Bertsekas, and Borkar 2001).

The control learning algorithm, *Differential Q-learning*, updates a table of estimates $Q_t : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ as follows:

$$Q_{t+1}(S_t, A_t) \doteq Q_t(S_t, A_t) + \alpha_{\nu_t(s,a)}\delta_t, \tag{3.25}$$
$$Q_{t+1}(s, a) \doteq Q_t(s, a), \ \forall s, a \neq S_t, A_t,$$

where $\{\alpha_n\}_{n\geq 1}$ is a step–size sequence satisfying Assumption 3.5, $\nu_t(s, a)$ is the number of times state-action pair (s, a) was updated up to and include time step t, $\{\alpha_n\}_{n\geq 1}$ and $\nu_t(s,a)$ jointly satisfy Assumption 3.6 (with $\mathcal{I} = \mathcal{S} \times \mathcal{A}$) and δ_t, the temporal-difference (TD) error at time step t, is:

$$\delta_t \doteq R_{t+1} - \bar{R}_t + \max_a Q_t(S_{t+1}, a) - Q_t(S_t, A_t), \tag{3.26}$$

where $\bar{R}_t$ is a scalar estimate of r_*, updated by:

$$\bar{R}_{t+1} \doteq \bar{R}_t + \eta \alpha_{\nu_t(s,a)} \delta_t, \tag{3.27}$$

and η is a positive constant.

Remark: The idea of updating the reward rate estimate using the TD error has been used in the R-learning algorithm by Schwartz (1993) and a variant of R-learning by Singh (1994). R-learning and Singh's (1994) variant share the same update rules with Differential Q-learning. However, in R-learning, (3.27) is only performed when A_t is greedy ($A_t \in \operatorname{argmax} Q(S_t, \cdot)$). This could potentially be inefficient if the greedy action is rarely chosen, which would happen if the behavior policy is not close to an optimal policy (e.g., a random policy). Singh's (1994) variant addressed this issue by performing (3.27) for every time step, just like our Differential Q-learning. His algorithm grounds the value of a fixed state-action pair to zero, while our algorithm does not. Singh (1994) argued that a potential disadvantage of not grounding the value is that action values can be too large. This does not seem to be a problem for our Differential Q-learning, because the set that action-value estimates converge to is bounded by a result shown later (Theorem 3.5). Finally, neither R-learning nor Singh's variant has been shown to be convergent.

We now transform the Differential Q-learning's update rules (3.25)—(3.27) into an *equivalent* update rule, from which it is easier to present further results. At each step, the increment to $\bar{R}_t$ is η times the increment to Q_t and $\sum Q_t$. Therefore, the cumulative increment can be written

$$\bar{R}_t - \bar{R}_0 = \eta \sum_{i=0}^{t-1} \alpha_{\nu_i(S_i,A_i)} \delta_i = \eta \left(\sum Q_t - \sum Q_0 \right)$$

$$\implies \bar{R}_t = \eta \sum Q_t - \eta \sum Q_0 + \bar{R}_0 = f(Q_t),$$

where $f : \mathbb{R}^{|\mathcal{S}|\times|\mathcal{A}|} \to \mathbb{R}$ satisfying

$$f(q) \doteq \eta \sum q - \eta \sum Q_0 + \bar{R}_0, \forall q \in \mathbb{R}^{|\mathcal{S}|\times|\mathcal{A}|}. \tag{3.28}$$

```
Algorithm 2: Differential Q-learning
   Input: The behavior policy $b$
   Algorithm parameters: step-size sequence $\alpha_n$, parameter $\eta$
1  Initialize $Q(s, a)$ $\forall s, a$; $\bar{R}$ arbitrarily (e.g., to zero)
2  $\nu(s, a) \leftarrow 0$ $\forall s, a$
3  Obtain initial $S$
4  while still time to train do
5      $\nu(S, A) \leftarrow \nu(S, A) + 1$
6      $A \leftarrow$ action given by $b$ for $S$
7      Take action $A$, observe $R, S'$
8      $\delta \leftarrow R - \bar{R} + \max_a Q(S', a) - Q(S, A)$
9      $Q(S, A) \leftarrow Q(S, A) + \alpha_{\nu(S,A)}\delta$
10     $\bar{R} \leftarrow \bar{R} + \eta\alpha_{\nu(S,A)}\delta$
11     $S \leftarrow S'$
12 end
13 return $Q$
```

Now substituting $\bar{R}_t$ in (3.25) with $f(Q_t)$, we have $\forall s \in \mathcal{S}, a \in \mathcal{A}$:

$$\begin{aligned} Q_{t+1}(S_t, A_t) &= Q_t(S_t, A_t) \\ &\quad + \alpha_{\nu_t(S_t,A_t)} \Big(R_{t+1} - f(Q_t) + \max_{a'} Q_t(S_{t+1}, a') - Q_t(S_t, A_t) \Big) \\ Q_{t+1}(s, a) &= Q_t(s, a), \ \forall s, a \neq S_t, A_t. \end{aligned} \tag{3.29}$$

The above update rule is in the same form as it of RVI Q-learning by Abounadi, Bertsekas, and Borkar (2001). In RVI Q-learning, this f function is called the reference function. However, the f function defined in (3.28) violates an assumption required by RVI Q-learning. This assumption consists of three parts: 1) f is L-Lipschitz for some $L \in \mathbb{R}$, 2) $f(\mathbf{1}) = 1$ and $f(x+c\mathbf{1}) = f(x)+c$ and, 3) $f(cx) = cf(x)$.

We now show the convergence of (3.29) to a set under Assumption 3.3, which subsumes the above assumption and the case defined in (3.28) as special cases. This result, therefore, immediately implies the convergence of Differential Q-learning and RVI Q-learning.

The set that (3.29) will be shown to converge to is the set of solutions of

(3.7) and

$$f(q) = r_*. \tag{3.30}$$

Denote this set as $\mathcal{Q}_\infty$. The properties of the set are stated formally in the following theorem.

Theorem 3.5. *If the MDP is weakly communicating and Assumption 3.3 holds, $\mathcal{Q}_\infty$ is non-empty, compact, connected, and possibly non-convex.*

This theorem is built upon a result by Schweitzer and Federgruen (1978), who showed that the set of solutions of q in (3.7) is non-empty, closed, unbounded, connected, and possibly non-convex.

The proof of this theorem will not be presented. However, I will prove in the next chapter a more general theorem that subsumes this theorem as a special case. The presentation and the proof of this general theorem must wait until the next chapter because it involves Semi-Markov Decision Processes, which generalize MDPs and will be required in the next chapter, but not in the current one.

The next theorem establishes the convergence of (3.29).

Theorem 3.6. *If the MDP is weakly communicating, Assumptions 3.3, 3.5, 3.6 (with $\mathcal{I} = \mathcal{S} \times \mathcal{A}$) hold, then* (3.29) *converges, almost surely, 1) $f(Q_t)$ to r_*, 2) Q_t to a sample-path dependent compact connected subset of $\mathcal{Q}_\infty$, and 3) almost surely, $\pi_t \in \Pi_*^D$ for sufficiently large t.*

The proof of the theorem will be presented at the end of this section.

Remark: Assumptions 3.5–3.6 used by the above theorem are also used in the convergence theorem of Differential TD-learning (Theorem 3.3). Here, for Differential Q-learning and RVI Q-learning, these assumptions can be satisfied if, for example, the step-size sequence $\{\alpha_n\}$ is $\{1/n\}_{n \geq 1}$ and the behavior policy b eventually visits all *state-action pairs* an infinite number of times. Note that if the MDP has transient states, these states can not be visited for an infinite number of times no matter what policy the agent follows. Therefore action values associated with these states can not be estimated accurately by any learning algorithm that uses only a single stream of experience. Nevertheless, Q_t still converges to a sample-path dependent compact connected set (but not necessarily a subset of $\mathcal{Q}_\infty$), and, because actions chosen in transient states do not influence the reward rate of a policy, the other two conclusions of the theorem still hold.

Algorithm 3: RVI Q-learning (Abounadi et al. 2001)

```
Input: The policy $b$ to be used (e.g., $\epsilon$-greedy), a reference function
       $f : \mathbb{R}^{|\mathcal{S}|\times|\mathcal{A}|} \to \mathbb{R}$
Algorithm parameters: step-size sequence $\alpha_n$, parameter $\eta$
Initialize $Q(s, a)$, $\forall s, a$ arbitrarily (e.g., to zero)
$\nu(s, a) \leftarrow 0\ \forall s, a$
Obtain initial $S$
while still time to train do
    $\nu(S, A) \leftarrow \nu(S, A) + 1$
    $A \leftarrow$ action given by $b$ for $S$
    Take action $A$, observe $R, S'$
    $\delta \leftarrow R - f(Q) + \max_a Q(S', a) - Q(S, A)$
    $Q(S, A) \leftarrow Q(S, A) + \alpha_{\nu(S,A)}\delta$
    $S \leftarrow S'$
end
return $Q$
```

The planning version of Differential Q-learning, called *Differential Q-planning*, uses simulated transitions generated just as in Differential TD-planning. Differential Q-planning maintains a table of value estimate $Q_n : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ and a reward rate estimate $\bar{R}_n$ and updates them just as in Differential Q-learning (3.25)–(3.27) (or equivalently, (3.29)) using $\tilde{S}_n, \tilde{A}_n, \tilde{R}_n, \tilde{S}'_n$ instead of $S_t, A_t, R_{t+1}, S_{t+1}$.

Just like Differential Q-planning, the planning version of RVI Q-learning operates following (3.29) using simulated transitions.

The next theorem shows the convergence of (3.29). Let $\tilde{r}_*$, $\tilde{\mathcal{Q}}_\infty$, and $\tilde{\Pi}^D_*$ be defined in the same way as r_*, $\mathcal{Q}_\infty$, and Π^D_*, except now for $\tilde{p}$ rather than p.

Theorem 3.7. *If the model MDP is weakly communicating and Assumptions 3.3, 3.5, 3.6 hold (with $\mathcal{I} = \mathcal{S}\times\mathcal{A}$), then* (3.29) *(using $\tilde{S}_n, \tilde{A}_n, \tilde{R}_n, \tilde{S}'_n$ instead of*

$S_t, A_t, R_{t+1}, S_{t+1}$) converges, almost surely, 1) $f(Q_n)$ to $\tilde{r}_*$, 2) Q_n to a sample-path dependent compact connected subset of $\tilde{\mathcal{Q}}_\infty$, and 3) almost surely, $\pi_n \in \tilde{\Pi}_*^D$ for a sufficiently large n.

The proof of the theorem is omitted as it is essentially the same as the proof of Theorem 3.6.

As promised earlier, the rest of this section provides the proof Theorem 3.6.

We first show that (3.29) is a special case of the General RVI Q's update (3.9). To see this point, we hypothesize two $|\mathcal{S} \times \mathcal{A}|$-sized random processes $\{S'_t\}, \{R'_t\}$ such that for any t, s, a, $S'_t(s, a), R'_t(s, a) \sim p(\cdot, \cdot \mid s, a)$. Now consider the stream of experience $\ldots, S_t, A_t, R_{t+1}, S_{t+1}, \ldots$. Then we have $R_{t+1} = R'_t(S_t, A_t)$ and $S_{t+1} = S'_t(S_t, A_t)$. Equation 3.9 reduces to (3.29) by choosing $i = (s, a), n = t, Y_n = \{(S_t, A_t)\}, r(i) = r(s, a), g(Q_n)(i) = \sum_{s'} p(s' \mid s, a) \max_{a'} Q_t(s', a')$, $M_{n+1}(i) = R'_t(s, a) - r(s, a) + \max_{a'} Q_t(S'_t(s, a), a') - g(Q_t)(s, a), \epsilon_{n+1} = \mathbf{0}$.

We now show that the assumptions required by General RVI Q are all satisfied.

1. Assumption 3.1 can be verified for $g(q)(s, a) = \sum_{s'} p(s' \mid s, a) \max_{a'} q(s', a')$ easily.

2. Assumption 3.2 is satisfied. Note that (3.8) becomes (3.7) given our definition of r and g. Because the MDP $\mathcal{M}$ is weakly communicating, $r_\# = r_*$ is the only solution of $\bar{r}$ in (3.7). Furthermore, there exists a solution of q in (3.7) by the discussion right after (3.7).

3. Assumptions 3.3–3.6 are assumed in the theorem statement.

4. Assumption 3.4 is satisfied by noting that the MDP is weakly communicating and applying Corollary 4.1, which will be presented and proved in the next chapter.

5. Assumption 3.7 can be verified by noting that state and reward spaces are finite and that $\epsilon_{n+1} = \mathbf{0}$.

Therefore all the conditions of Theorem 3.1 have been verified and we have a.s., Q_t converges to $\mathcal{Q}_\#$ and $f(Q_t) = \bar{R}_t$ converges to $r_\#$, which are $\mathcal{Q}_\infty$ and r_*, respectively, in the current setting. The rest of the proof follows Lemma 4.3, which will be presented and proved in the next chapter.

3.6 Control Experiments

This section presents two sets of experiments testing Differential Q-learning.

The first set of experiments tests Differential Q-learning in the four-room domain (shown in Figure 3.1) with RVI Q-learning and Gosavi's (2004) algorithm as two baselines.

Gosavi's (2004) algorithm estimates the reward rate $\bar{R}_t$ by tracking a weighted average of observed rewards when greedy actions are taken. If the action executed is a greedy choice at time step t, then $\bar{R}_{t+1}$ is updated:

$$\bar{R}_{t+1} \doteq \bar{R}_t + \beta_t(\hat{R}_t - \bar{R}_t),$$

where $\{\beta_t\}$ is a step-size sequence. Otherwise $\bar{R}_{t+1} = \bar{R}_t$. The action-value function is updated with (3.25) with δ_t as defined in (3.27). The pseudocode of his algorithm can be found in Algorithm 4 for readers' convenience.

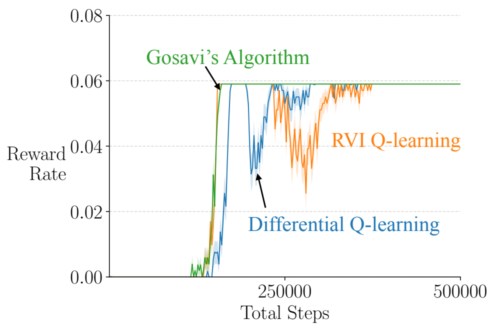
Remark: The convergence result of Gosavi's proposed algorithm is summarized in his Theorem 2. In the proof of the theorem, he used Borkar's (1997) two-time scale stochastic approximation result to prove the convergence of the proposed algorithm. Specifically, he argued that his algorithm is a special case of the general class of algorithms considered by Borkar (1997). As Gosavi quotes, "Note that the Eqs. (48) and (49) for SMDPs form a special case of the general class of algorithms (29) and (30) analyzed using the lemma given in Section 5.1.1." However, a closer look at these equations shows that equation (49) is not a special case of equation (30). Note that because ρ^k is a scalar, y^k only has one element, and thus the f function in equation (30) does not vary across different state-action pairs. However, this is not true for the f function in equation (49). It appears to me that there is no straightforward way to fix this issue.

Algorithm 4: Gosavi's (2004) Algorithm for MDPs

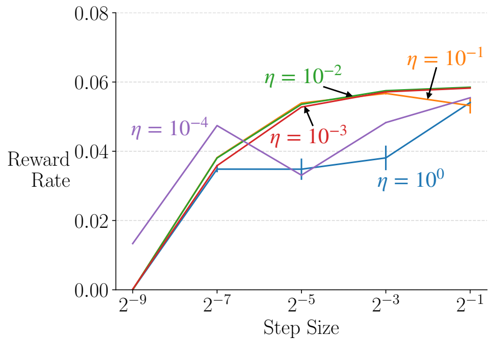
```
    Input: Behavioral policy $b$
    Algorithm parameters: step-size sequences $\alpha_n, \beta_n$
 1  Initialize $Q(s,a)$, $\forall\, s \in \mathcal{S}, o \in \mathcal{A}, \bar{R}$ arbitrarily (e.g., to zero)
 2  $\nu(s,a) \leftarrow 0\ \forall s, a$
 3  $N \leftarrow 0$
 4  Obtain initial $S$
 5  while still time to train do
 6      $A \leftarrow$ action given by $b$ for $S$
 7      $\nu(S,A) \leftarrow \nu(S,A) + 1$
 8      Take action $A$, observe $R, S'$
 9      $\delta \leftarrow R - \bar{R} + \max_a Q(S',a) - Q(S,A)$
10      $Q(S,A) \leftarrow Q(S,A) + \alpha_{\nu(S,A)}\delta$
11      if $A \in \operatorname{argmax} Q(S,\cdot)$ then
12          $\bar{R} \leftarrow \bar{R} + \beta_N(R - \bar{R})$
13      end
14      $S \leftarrow S'$
15      $N \leftarrow N + 1$
16  end
17  return $Q$
```

For all three tested algorithms, the agent used an ϵ-greedy policy with $\epsilon = 0.1$. For Differential Q-learning, the tested choices of parameters α_n and η were the same as those used for Differential TD-learning in the prediction experiments (Section 3.4). For RVI Q-learning, the tested choices of the step-size sequence α_t were the same as those for Differential Q-learning. Tested reference functions include the values of 26 particular state-action pairs, the max of all action values, the min of all action values, the mean of all action values, and the max of action values per state averaged over states. For Gosavi's algorithm, the choices of both two step sizes were the same as the choices of α_t for Differential Q-learning.

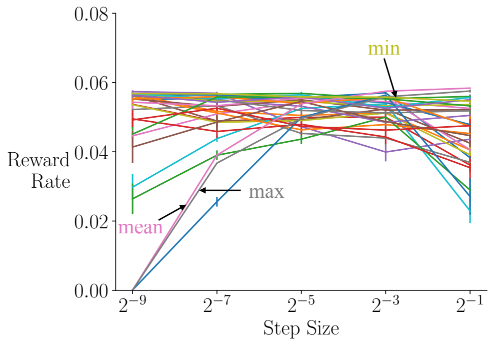
For control algorithms, it is natural to consider the reward rate achieved by the learned policy. To estimate the reward rate, for every 2000 steps, I evaluated the reward rate achieved by the learned policy, which was just a greedy policy w.r.t. the action-value estimates, by following the policy for 2000 evaluation steps and computing the average reward over these evaluation steps. No updates to parameters were performed during evaluations.



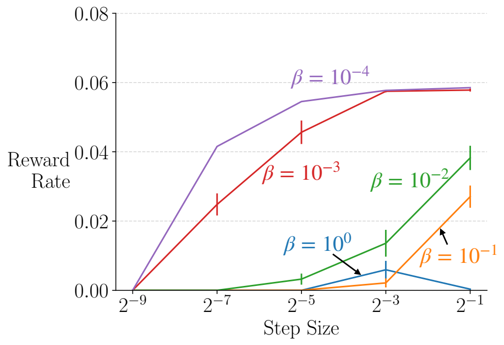
(a) Learning curves for the three algorithms. It can be seen that all three algorithms can achieve an optimal reward rate eventually in the tested domain.



(b) Reward rate for Differential Q-learning. The algorithm is sensitive to the first step size and is relatively less sensitive to its second step size η. Moreover, intermediate values of η perform the best.



(c) The reward rate achieved by RVI Q-learning is sensitive to its choice of the reference state-action pairs/reference functions. For small step sizes, some choices of the reference state-action pairs can achieve a high reward rate quickly.



(d) The reward rate of Gosavi's algorithm is sensitive to both of the two step sizes. In order to achieve a high reward rate quickly in the tested domain, the first step size should be large while the second step size should be small.

Figure 3.3: Plots showing reward rate achieved by policies learned by Differential Q-learning, RVI Q-learning, and Gosavi's algorithm.

Learning curves of the reward rate during evaluations of three tested algorithms are shown in Figure 3.3a. For each algorithm, each point in the algorithm's curve denotes the reward rate obtained in an evaluation. The shading region denotes one standard error. For each curve, the corresponding parameter setting is the one that resulted in the highest reward rate averaged over the last five evaluations, which happened in the last $10,000$ steps. The parameter studies for the three tested algorithms in terms of the reward rate achieved by their learned policies are shown in the other three subfigures in Figure 3.3. Each point in a curve in these three subfigures denotes the reward rate averaged over all $500,000/2000 = 250$ evaluations.

Just as I did for prediction experiments, for control experiments, I also used the relative value and reward rate errors. Given an estimated differential action-value function Q, the relative value error at time t is defined as follows,

$$\text{Relative Value Error}_t \doteq |(Q_t(S_t, A_t) - Q_t(s_0, a_0)) - (q_*(S_t, A_t) - q_*(s_0, a_0))|,$$

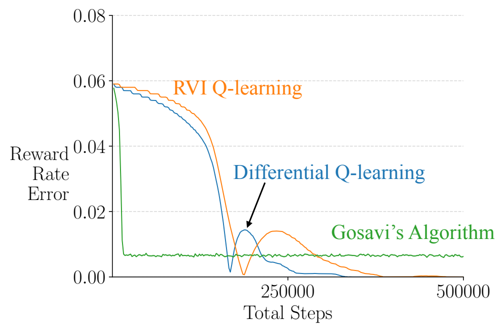
where (s_0, a_0) is a fixed state-action pair. I chose s_0 to be the state indicated by the yellow cell and chose a_0 to be the action `up` in this experiment. In the relative value error, q_* is any solution of (3.7). Note that this four-room domain is a bit special. The MDP is communicating, but all solutions of q in (3.7) are only different by a constant vector, just as in unichain MDPs. Thus choosing any q_* would be no different.

For the reward rate estimate $\bar{R}_t$,

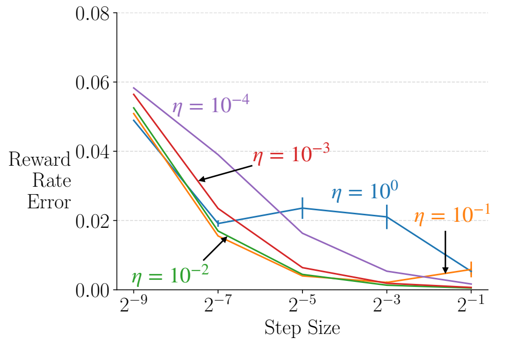
$$\text{Reward Rate Error}_t \doteq |\bar{R}_t - r_*|.$$

Learning curves of the relative value and reward rate errors of the three tested algorithms are shown in Figure 3.4a and Figure 3.5a. Each point in a curve denotes the error averaged over the past 2000 steps. The shading region denotes one standard error. The reported parameter settings were chosen in the same way as in the prediction experiments. The parameter studies for the reward rate error and relative value error are shown in the corresponding subfigures in Figure 3.4 and Figure 3.5, respectively.

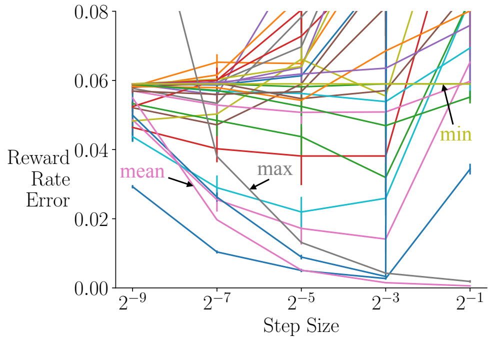
The reward rate learning curves (Figure 3.3a) show that, asymptotically, the learned policies by each of the three tested algorithms achieved the optimal reward rate. Parameter studies in terms of the achieved reward rate (Figure 3.3b–3.3d) show that in the tested problem 1) Differential Q-learning learned faster when α_n was large and η was some intermediate value, 2) the learning rate of Differential Q-learning varies less significantly across its second parameter than Gosavi's algorithm, 3) the learning rate of RVI Q-learning varies across different choices of the reference function significantly.



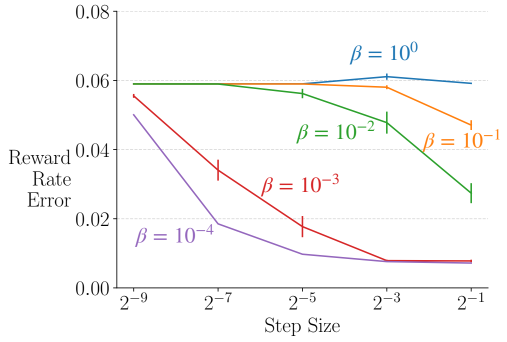
(a) Learning curves for the three algorithms. It can be seen that Differential Q-learning and RVI Q-learning achieved a zero reward rate error eventually in the tested domain while Gosavi's algorithm failed to do so.



(b) Reward rate error for Differential Q-learning. Intermediate values of η perform the best.



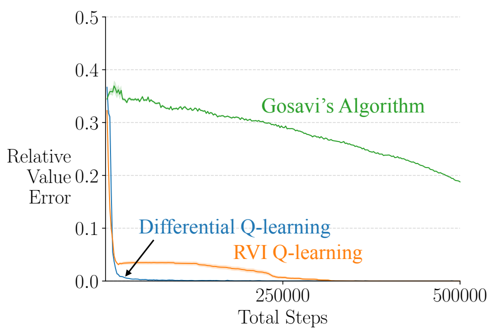
(c) For RVI Q-learning, the reward rate error is high for most of the tested choices of reference functions.



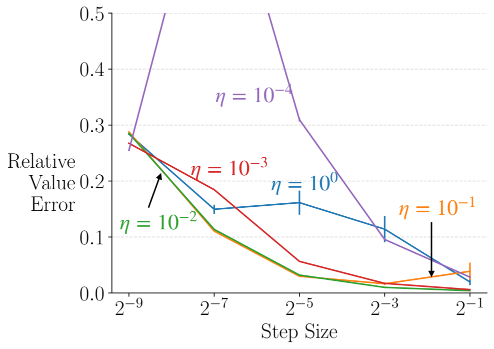
(d) The relative value error of Gosavi's algorithm. The algorithm's reward rate error is again, sensitive to the choice of the two step sizes and the algorithm's best step sizes still can not achieve a near-zero reward rate error while both of the other two algorithms with their best parameter settings can.

Figure 3.4: Plots showing reward rate error for Differential Q-learning, RVI Q-learning, and Gosavi's algorithm.

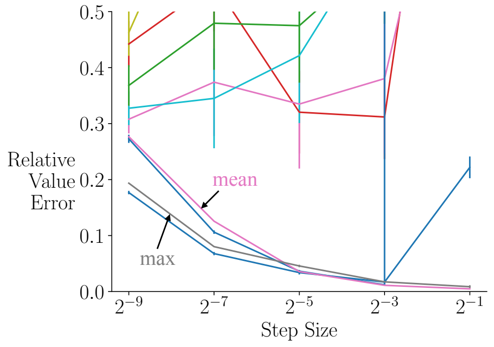
The reward rate error learning curves (Figure 3.4a) show that, asymptotically, the reward rate estimate in Differential Q-learning and the output of the f function both approached the optimal reward rate r_* asymptotically. On the contrary, the reward rate estimate of Gosavi's algorithm failed to converge to r_*. Note that the parameter setting minimizes the error at the end of train-



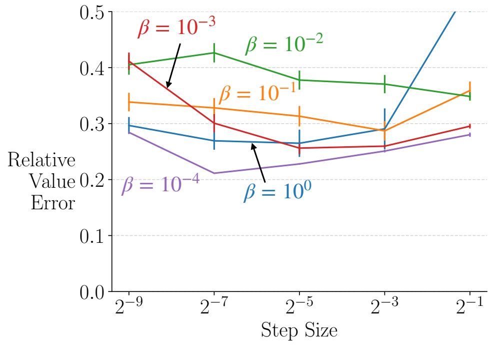
(a) Learning curves for the three algorithms. It can be seen that Differential Q-learning and RVI Q-learning can achieve a zero relative value error eventually in the tested domain while Gosavi's algorithm fails to do so.



(b) Relative value error for Differential Q-learning. Again, intermediate values of η perform the best.



(c) The relative value error is high for most of the tested choices of reference functions.



(d) The relative value error of Gosavi's algorithm is generally higher compared with the two other algorithms.

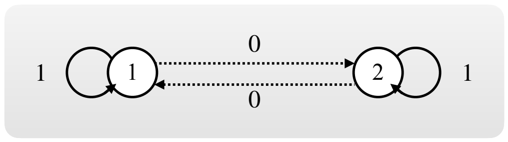
Figure 3.5: Plots showing relative value error for Differential Q-learning, RVI Q-learning, and Gosavi's algorithm.

ing. Thus other parameter settings resulted in even higher asymptotic errors. Parameter studies in terms of the reward rate error (Figure 3.4b–3.4d) show that the learning rate of Differential Q-learning was less sensitive to its choice of the parameters than RVI Q-learning and Gosavi's algorithm.

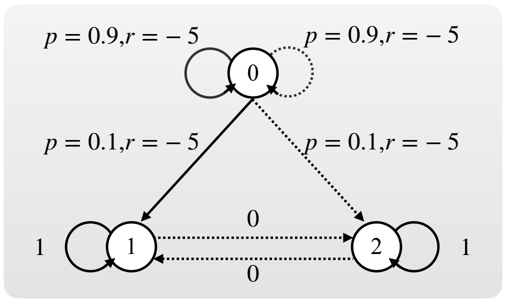
The above observations for the reward rate error carry over to the relative value error learning curves and parameter studies (Figure 3.5).

To conclude, the first set of control experiments suggests several points. First, the learned policy of Differential Q-learning achieved the optimal reward rate, and the estimated action values and the estimated reward rate indeed converged to desired points, in accordance with Theorem 3.6. Second, Differential Q-learning was less sensitive to its second parameter η compared with the two baseline algorithms w.r.t. their second parameters (reference function for RVI Q-learning and β_n for Gosavi's algorithm). Third, for Gosavi's algorithm, the estimated action values and the estimated reward rate failed to converge to the desired points, which empirically validates my arguments about the incorrectness of his convergence proof at the beginning of this section.

In the first set of experiments, the MDP (four-room domain) is communicating but is somewhat special. It has a unique solution of q up to an additive constant in the optimality equation (3.7). Thus the solutions of Differential Q-learning and RVI Q-learning are unique.



(a) A communicating MDP. States 1 and 2 are in the same communicating class. For each of the two states, taking action `solid` stays at the same state and receives a reward of one, and taking action `dashed` moves to the other state and receives a reward of zero. The initial state of the MDP is state 1.



(b) A weakly communicating MDP constructed by adding one more state (State 0) to the MDP shown on the left panel. State 0 is transient. In this state, taking both `solid` and `dashed` actions stays at the transient state with probability 0.9. The MDP moves to state 1 with probability 0.1 given action `solid` and moves to state 2 with probability 0.1 given action `dashed`. The reward starting from state 0 is always -5. The initial state of the MDP is state 0.

Figure 3.6: Tested MDPs for verifying the convergence of Differential Q-learning and RVI Q-learning when the solution set has more than one degree of freedom.

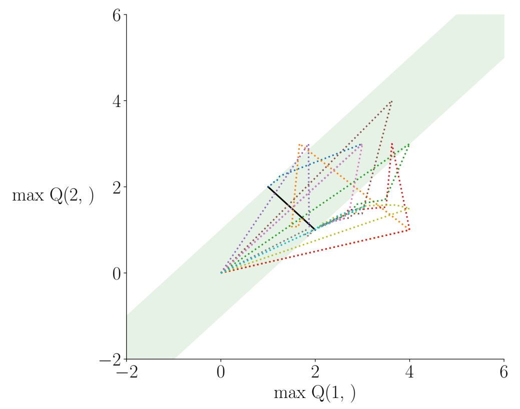
My second set of experiments was to empirically verify the convergence results of Differential Q-learning and RVI Q-learning (Theorem 3.6) when the solution set of q in (3.7) has *more than one degree of freedom*.

For this second set of experiments, the experiment setting was chosen to satisfy assumptions required by Theorem 3.6. Notably, I chose the step-size sequence $\alpha_n = 1/n$ instead of a constant step-size sequence. While the constant step size sequence is commonly used in practical algorithms, it is not allowed by the step-size assumption required by Theorem 3.6. The $1/n$ step size sequence is, however, allowed given that the behavior policy is also randomized (see Example 3.1). The test domains were two weakly communicating MDPs. The first MDP (shown in Figure 3.6a) is not only a weakly communicating MDP but also a communicating MDP, because it does not have transient states. The second MDP is the first communicating MDP plus a transient state. I chose the behavior policy to be a randomized one, which chooses action `solid` with probability 0.8 and action `dashed` with probability 0.2 for all states so that the required assumptions Assumption 3.5 and Assumption 3.6 are satisfied. Therefore all assumptions required by Theorem 3.6 are satisfied.

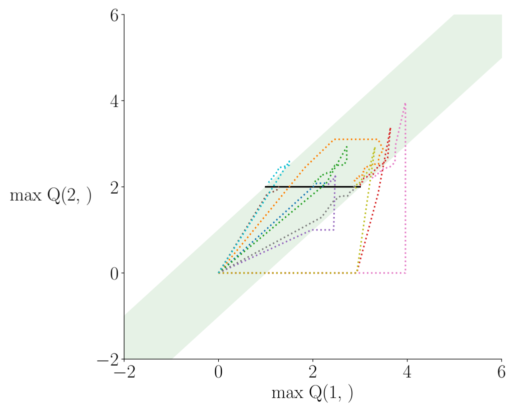
Experiment settings that are unrelated to the required assumptions of Theorem 3.6 were chosen as follows. I tested Differential Q-learning with initial action values 0, initial reward rate estimate -3, and $\eta = 1$, and tested RVI Q-learning with action values initialized to 0 and $f(q) = q(1, \texttt{dashed})$. I performed 10 runs for each algorithm. Each run started from state 1 and lasted for 1000 steps. For every 10 steps, I recorded the estimated action values.

Figure 3.7a and Figure 3.7b show, in the first MDP, the dynamics of the higher action value at each state for Differential Q-learning and RVI Q-learning, respectively. It can be seen that for both algorithms, 1) for each run, the estimated action-value function converged to a point in the solution set (the black line segments), and 2) for different runs, the estimated action values generally converged to different points in the solution set.

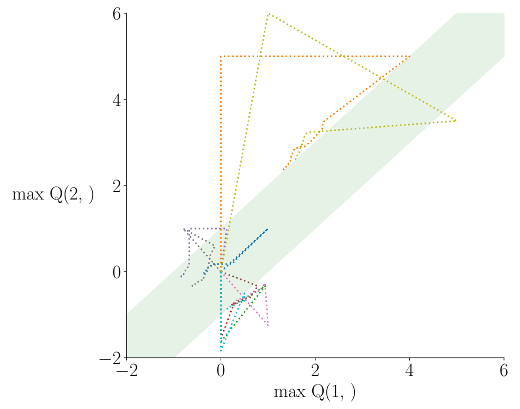
The dynamics of the estimated action values of Differential Q-learning and RVI Q-learning in the second MDP are shown in Figure 3.7c and Figure 3.7d, respectively. This time, the solution set of Differential Q-learning depends on the action values associated with the transient states when entering the



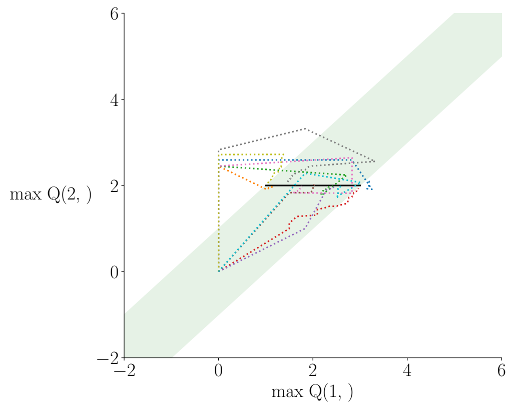
(a) Differential Q-learning in the communicating MDP (Figure 3.6a).



(b) RVI Q-learning in the communicating MDP (Figure 3.6a).



(c) Differential Q-learning in the weakly communicating MDP (Figure 3.6b). Note that, I didn't draw a black line segment in this plot, because the set the algorithm converges to varies across different trajectories. Specifically, it depends on the estimated action values of transient states when the agent first enters the communicating class.



(d) RVI Q-learning in the weakly communicating MDP (Figure 3.6b).

Figure 3.7: Dynamics of the estimated values produced by Differential Q-learning and RVI Q-learning in the two MDPs shown in Figure 3.6. The black line segment in each plot marks the set of points that the corresponding algorithm should converge to. The green regions denote the solution set of the action-value optimality equation (3.7).

communicating class. Nevertheless, the estimated action-value function in all runs converged to the region within the green region, which corresponds to the solution set of the optimality equation. On the other hand, the solution set of RVI Q-learning with the choice of the reference function $f(q) = q(1, \texttt{dashed})$

does not depend on the action values associated with states in the communicating class when entering the class. Therefore the solution set did not vary across different runs.

In conclusion, the second set of experiments empirically verified Theorem 3.6 for communicating and weakly communicating MDPs.

3.7 Centered Algorithms

This section introduces algorithms that can obtain the differential value function. Recall that all average-reward algorithms, including the ones presented here, converge to an *uncentered* differential value function, in other words, the actual differential value function plus some unknown offset that depends on the algorithm itself and design choices such as initial values or reference states. This section introduces a simple technique to compute the *offset* in the value estimates for both on- and off-policy learning and sample-based planning. Once the offset is computed, it can simply be subtracted from the value estimates to obtain the estimate of the actual (centered) differential value function.

First, we show how the offset can be eliminated for Differential TD-learning. For this purpose, I introduce, in addition to the first estimator (3.19)–(3.21), a second estimator for which the rewards are the value estimates of the first estimator. The second estimator maintains an estimate of the scalar offset $\bar{V}_t$, an auxiliary table of estimates $W_t(s), \forall s \in \mathcal{S}$, and uses the following update rules:

$$W_{t+1}(S_t) \doteq W_t(S_t) + \beta_{\nu_t(S_t)}\rho_t\Delta_t, \tag{3.31}$$

$$W_{t+1}(s) \doteq W_t(s), \ \forall s \neq S_t,$$

where $\{\beta_n\}$ is a step-size sequence, Δ_t is the TD error of the second estimator at time steps t:

$$\Delta_t \doteq V_t(S_t) - \bar{V}_t + W_t(S_{t+1}) - W_t(S_t), \tag{3.32}$$

and

$$\bar{V}_{t+1} \doteq \bar{V}_t + \kappa\beta_{\nu_t(S_t)}\rho_t\Delta_t, \tag{3.33}$$

and κ is a positive constant. We call (3.19)–(3.21) plus (3.31)–(3.33) *Centered Differential TD-learning*. See Algorithm 5 for the pseudo-code of Centered Differential TD-learning.

Before presenting the convergence theorem, I briefly give the intuition on why this technique can successfully compute the offset. By Theorem 3.3, $\bar{R}_t$ converges to $r(\pi)$ and V_t converges to some v_∞ almost surely, where $v_\infty(s) = v_\pi(s) + c, \forall s \in \mathcal{S}$ for some offset $c \in \mathbb{R}$. Lemma 3.8 (to be presented shortly) shows that $\sum_s d_\pi(s) v_\pi(s) = 0$, where d_π is the unique limiting state distribution following policy π (the uniqueness follows the unichain assumption), which implies $\sum_s d_\pi(s) v_\infty(s) = c$. As V_t converges to v_∞, $\sum_s d_\pi(s) V_t(s)$ converges to c. Now note that $\sum_s d_\pi(s) V_t(s)$ and $r(\pi) = \sum_s d_\pi(s) r_\pi(s)$ are of the same form. Therefore the estimation of $\sum_s d_\pi(s) V_t(s)$ can be similar to it of $r(\pi)$, using V_t as the reward. This leads to the second estimator: (3.31)–(3.33). The next theorem shows that Centered Differential TD-learning converges.

Algorithm 5: Centered Differential TD-learning

```
   Input: The policy $\pi$ to be evaluated, and $b$ to be used
   Algorithm parameters: step-size sequences $\alpha_n, \beta_n$, parameters $\eta, \kappa$
1  Initialize $V(s), W(s),\ \forall s,\ \bar{R}, \bar{V}$ arbitrarily (e.g., to zero)
2  $\nu(s) \leftarrow 0,\ \forall s$
3  Obtain initial $S$
4  while still time to train do
5      $\nu(S) \leftarrow \nu(S) + 1$
6      $A \leftarrow$ action given by $b$ for $S$
7      Take action $A$, observe $R, S'$
8      $\delta \leftarrow R - \bar{R} + V(S') - V(S)$
9      $\rho \leftarrow \pi(A \mid S) / b(A \mid S)$
10     $V(S) \leftarrow V(S) + \alpha_{\nu(S)} \rho \delta$
11     $\bar{R} \leftarrow \bar{R} + \eta \alpha_{\nu(S)} \rho \delta$
12     $\Delta \leftarrow V(S) - \bar{V} + W(S') - W(S)$
13     $W(S) \leftarrow W(S) + \beta_{\nu(S)} \Delta$
14     $\bar{V} \leftarrow \bar{V} + \kappa \beta_{\nu(S)} \Delta$
15     $S \leftarrow S'$
16 end
17 return $V - \bar{V}\mathbf{1}$.
```

Theorem 3.8. *If the target policy π is unichain, Assumptions 3.5, 3.6 hold for both α_n and β_n (with $\mathcal{I} = \mathcal{S}$), and Assumption 3.9 holds, then Centered*

Differential TD-learning converges, almost surely, $V_t(s) - \bar{V}_t$ to $v_\pi(s)$ for all s.

The proof of the theorem will be presented at the end of this section.

The planning version of Centered Differential TD-learning is called *Centered Differential TD-planning.* Centered Differential TD-planning estimates, in the *model* MDP, the differential value function $\tilde{v}_\pi$ of the target policy π. It uses simulated experience $\ldots, \tilde{S}_n, \tilde{A}_n, \tilde{R}_n, \tilde{S}'_n, \ldots$ just as in Differential TD-planning. In addition, just like Differential TD-planning, Centered Differential TD-planning maintains V_n and $\bar{R}_n$. Centered Differential TD-planning also maintains an auxiliary table of estimates $W_n(s,a), \forall s \in \mathcal{S}, a \in \mathcal{A}$ and an offset estimate $\bar{V}_n$, and updates them just as in Centered Differential TD-learning, using $\tilde{S}_n, \tilde{A}_n, \tilde{R}_n, \tilde{S}'_n$ instead of $S_t, A_t, R_{t+1}, S_{t+1}$.

Theorem 3.9. *Under the same assumptions made in Theorem 3.8 (except now for the model MDP corresponding to $\tilde{p}$ rather than $\hat{p}$), Centered Differential TD-planning converges, almost surely, $V_n(s) - \bar{V}_n$ to $\tilde{v}_\pi(s)$ for all s.*

The proof of the theorem is very similar to it of Theorem 3.8 and is therefore omitted.

Applying the centering technique to the Differential Q-learning algorithm results in *Centered Differential Q-learning.* This centered algorithm maintains, in addition to the first estimator (Equations 3.25–3.27), a second estimator in which the reward is the value estimate of the first estimator. The second estimator maintains a scalar offset estimate $\bar{Q}_t$, an auxiliary table of estimates $W_t(s,a), \forall s \in \mathcal{S}, a \in \mathcal{A}$, and uses the following update rules:

$$W_{t+1}(S_t, A_t) \doteq W_t(S_t, A_t) + \beta_{\nu_t(S_t,A_t)} \rho_t \Delta_t, \quad \text{and}$$

$$W_{t+1}(s,a) \doteq W_t(s,a), \forall s \neq S_t, a \neq A_t, \tag{3.34}$$

$$\bar{V}_{t+1} \doteq \bar{V}_t + \kappa \beta_{\nu_t(S_t,A_t)} \rho_t \Delta_t, \tag{3.35}$$

where

$$\Delta_t \doteq Q_t(S_t, A_t) - \bar{Q}_t + W_t(S_{t+1}, \operatorname*{argmax}_{a'} Q_t(S_{t+1}, a')) - W_t(S_t, A_t), \tag{3.36}$$

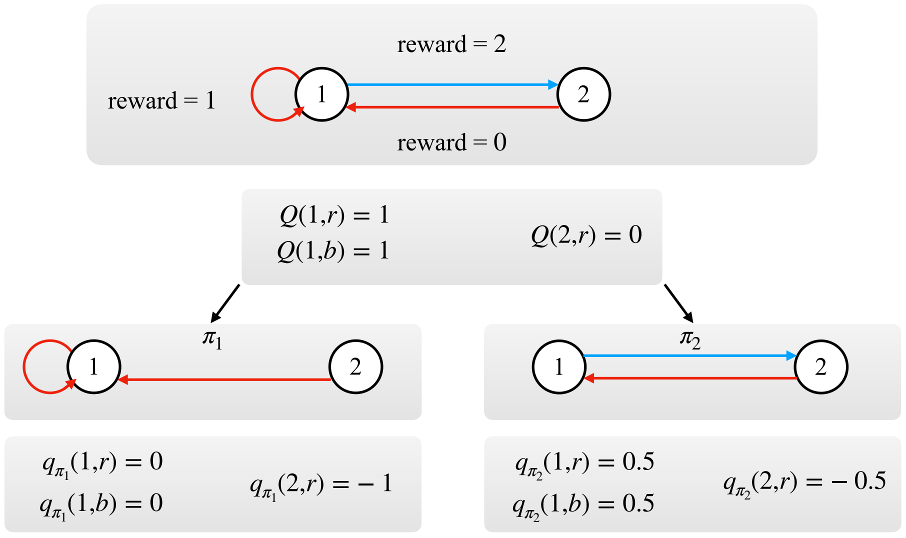


Figure 3.8: Example showing that there can be two different greedy policies w.r.t. a solution of the action-value optimality equation and the action-value functions of the two policies are different. *First row:* The example MDP is a unichain MDP. There are two states marked by two circles respectively. There are two actions *red (r)* and *blue (b)*. *Second row:* a solution of the action-value optimality equation. Note that the MDP is unichain and thus all solutions of the equation are different by a constant. *Third row:* two different greedy policies w.r.t. the solution. *Fourth row:* the two different policies correspond to different differential action-value functions.

is the TD error of the second estimator, $\rho_t \doteq \pi(A_t \mid S_t)/b(A_t \mid S_t)$ is the importance sampling ratio, $\{\beta_n\}_{n\geq 1}$ is a step-size sequence, and κ is a positive constant. I call (3.25)–(3.27) plus (3.34)–(3.36) Centered Differential Q-learning. See Algorithm 6 for the pseudo-code of Centered Differential Q-learning.

I now present a convergence theorem for Centered Differential Q-learning. Unlike the previous theorems, this theorem requires that the optimal policy is unique. The reason is, if there are multiple optimal policies all achieving the optimal average reward, the greedy policy w.r.t. Q_t will jump between these optimal policies even in the limit so the second estimator can not evaluate any particular optimal policy. In addition, unlike the discounted case, where different optimal policies all correspond to the same unique optimal value function, in the average reward case, in general, optimal policies correspond to different differential value functions. An example illustrating this point

Algorithm 6: Centered Differential Q-learning

```
   Input: The behavior policy $b$
   Algorithm parameters: step-size sequences $\alpha_n, \beta_n$, parameters $\eta, \kappa$
 1 Initialize $Q(s,a), W(s,a),\ \forall s,a; \bar{R}, \bar{Q}$ arbitrarily (e.g., to zero)
 2 $\nu(s,a) \leftarrow 0,\ \forall s,a$
 3 Obtain initial $S$
 4 while still time to train do
 5     $A \leftarrow$ action given by $b$ for $S$
 6     $\nu(S,A) \leftarrow \nu(S,A) + 1$
 7     Take action $A$, observe $R, S'$
 8     $\delta \leftarrow R - \bar{R} + \max_a Q(S',a) - Q(S,A)$
 9     $Q(S,A) \leftarrow Q(S,A) + \alpha_{\nu(S,A)}\delta$
10     $\bar{R} \leftarrow \bar{R} + \eta\alpha_{\nu(S,A)}\delta$
11     $\Delta \leftarrow Q(S,A) - \bar{Q} + W(S', \operatorname{argmax}_a Q(S',a)) - W(S,A)$
12     $W(S,A) \leftarrow W(S,A) + \beta_{\nu(S,A)}\Delta$
13     $\bar{Q} \leftarrow \bar{Q} + \kappa\beta_{\nu(S,A)}\Delta$
14     $S \leftarrow S'$
15 end
16 return $Q - \bar{Q}\mathbf{1}$.
```

is shown in Figure 3.8. Therefore, in order to use the second estimator to evaluate some policy derived from Q_t, that policy must converge as $t \to \infty$.

In practice, our algorithm can still deal with problems with multiple optimal policies. This can be achieved by choosing a small threshold $\epsilon > 0$, and then replacing the $\operatorname{argmax}_a Q(s,a)$ in our algorithms with the first action $\tilde{a}$ (assume that actions are ranked) satisfying $Q(s,\tilde{a}) >= \max_a Q(s,a) - \epsilon$. The resulting policy of the algorithm will converge to an optimal policy if ϵ is sufficiently small.

Theorem 3.10. *If the MDP is weakly communicating and Assumptions 3.5, 3.6 hold for both α_n and β_n (with $\mathcal{I} = \mathcal{S} \times \mathcal{A}$), and the optimal policy is unique (denote it as q_{π_*}), then the Centered Differential Q-learning algorithm converges, almost surely, $Q_t(s,a) - \bar{Q}_t$ to $q_{\pi_*}(s,a)$ for all $s \in \mathcal{S}, a \in \mathcal{A}$.*

The proof of the theorem will be presented at the end of this section.

The planning version of Centered Differential Q-learning is called *Centered Differential Q-planning*. Centered Differential Q-planning estimates the differential action-value function of the optimal policy (assumed to be unique just as in the learning case) in the model MDP. Denote $\tilde{\pi}_*$ as the unique optimal policy and $\tilde{q}_{\tilde{\pi}_*}$ as the differential action-value function of this policy. Centered Differential Q-planning estimates these two quantities using simulated experience just as in Centered Differential TD-planning.

Theorem 3.11. *Under the same assumptions made in Theorem 3.10 (except now for the model MDP corresponding to $\tilde{p}$ rather than p and that $\tilde{p}$ can be weakly communicating rather than communicating), Centered Differential Q-planning converges, almost surely, $Q_n(s,a) - \bar{Q}_n$ to $\tilde{q}_{\tilde{\pi}_*}(s,a)$ for all s, a.*

The proof of Theorem 3.11 is quite similar to it of Theorem 3.10 and is therefore omitted.

Proof of Theorem 3.8

The following lemma will be used.

Lemma 3.8. *Given a policy $\pi \in \Pi$, assume that the induced Markov chain under π is unichain. Let d_π be the unique stationary distribution following policy π. Then*

(i) $(v, \bar{r}) = (v_\pi, r(\pi))$ *is the unique solution of* (3.2) *and*

$$\sum_s d_\pi(s) v(s) = 0, \tag{3.37}$$

and

(ii) *if* $v = v_\pi + c\boldsymbol{1}$ *then* $c = \sum_s d_\pi(s) v(s)$.

Proof. Recall the transition matrix P_π defined in (3.3). Define P_π^∞ as the Cesaro limit of the sequence $\{P_\pi^i\}_{i=1}^\infty$:

$$P_\pi^\infty \doteq \lim_{n \to \infty} \frac{1}{n} \sum_{i=0}^{n-1} P_\pi^i.$$

Because $\mathcal{S}$ is finite, the Cesaro limit exists and P_π^∞ is a stochastic matrix (has row sums equal to 1). Because the Markov chain induced by π is unichain, all rows of P_π^∞ are identical and are all equal to $d_\pi^\top$. Then the average-reward rate following π can be written as

$$r(\pi) = d_\pi^\top r_\pi. \tag{3.38}$$

The differential value function following policy π can be written as

$$v_\pi(s) = \lim_{N\to\infty} \frac{1}{N} \sum_{k=0}^{N-1} \sum_{t=0}^{k} P_\pi^t (r_\pi - r(\pi))(s),$$

or $v_\pi = H_{P_\pi} r_\pi$ in vector form, where $H_{P_\pi} \doteq \lim_{N\to\infty} \frac{1}{N} \sum_{k=0}^{N-1} \sum_{t=0}^{k} (P_\pi^t - P_\pi^\infty)$.

The differential value function v_π satisfies (3.2) due to Theorem 8.2.6 (a) by Puterman (1994).

To see that v_π satisfies the equation (3.37), let's apply Equation A.18 in Appendix A by Puterman (1994), which states that $P_\pi^\infty H_{P_\pi}$ is an all-zero matrix. Therefore we have $d_\pi^\top H_{P_\pi} = \mathbf{0}^\top$ because all rows of P_π^∞ are $d_\pi^\top$. Because $v_\pi = H_{P_\pi} r_\pi$, we have $d_\pi^\top v_\pi = d_\pi^\top H_{P_\pi} r_\pi = 0$.

To verify that v_π is the unique solution of (3.2) and (3.37), suppose there exists another vector $v' \neq v_\pi$ satisfying (3.2) and (3.37), then $v' = v_\pi + c\mathbf{1}$ for some $c \neq 0$ (any two solutions of (3.2) differ by a constant). Substituting this into (3.37), we have

$$d_\pi^\top v' = d_\pi^\top (v_\pi + c\mathbf{1}) = d_\pi^\top v_\pi + c d_\pi^\top \mathbf{1} = c$$

To satisfy (3.37), we must have $c = 0$. Therefore, v_π is the unique solution of (3.2) and (3.37).

To prove the second part, consider $v = v_\pi + c\mathbf{1}$, then we have $\sum_s d_\pi(s) v(s) = \sum_s d_\pi(s)(v_\pi + c\mathbf{1})(s) = c$. □

Because (3.19)–(3.21) are independent of (3.31)–(3.33), we have $V_t \to v_\infty$ and $\bar{R}_t \to r(\pi)$ according to Theorem 3.3.

Similar as the proof of Theorem 3.3, we can combine (3.31)–(3.33) and obtain a single update rule:

$$W_{t+1}(S_t) = W_t(S_t) + \beta_{\nu_t(S_t)} \rho_t \Big(V_t(S_t) - f(W_t) + W_t(S_{t+1}) - W_t(S_t) \Big), \tag{3.39}$$

$$\text{where } f : \mathbb{R}^{|\mathcal{S}|} \to \mathbb{R} \text{ satisfying } f(w) \doteq \eta \sum w - \kappa \sum W_0 + \bar{V}_0, \forall w \in \mathbb{R}^{|\mathcal{S}|}. \tag{3.40}$$

We now show that (3.39) is a special case of the General RVI Q's update (3.9). To see this point, recall A_t', S_t', R_t', Y_t defined for Differential TD-learning, right below (3.24). Equation 3.9 reduces to (3.39) by choosing $\mathcal{I} = \mathcal{S}, n = t, Q_n = W_t$, and

$$
\begin{aligned}
&r(i) = v_\infty(s), \\
&g(W_t)(i) = \sum_a \pi(a \mid s) \sum_{s'} p(s' \mid s, a) W_t(s'), \\
&M_{n+1}(i) = \rho_t(s)\Big(v_\infty(s) - f(W_t) + W_t(S'_t(s)) - W_t(s)\Big) \\
&- \left(v_\infty(S_t) - f(W_t) + g(W_t)(s) - W_t(s)\right), \\
&\epsilon_{n+1}(i) = \rho_t(s)V_t(s) - \rho_t(s)v_\infty(s),
\end{aligned}
$$

where $\rho_t(s) \doteq \pi(A'_t(s) \mid s)/b_t(A'_t(s) \mid s)$.

We now show that the assumptions required by General RVI Q are all satisfied.

1. Assumption 3.1 can be verified for the choice of $g(w)$ easily.

2. Assumptions 3.2, 3.3 can be verified the same way as in the proof of Differential TD-learning by noting that (3.8) reduces to the state-value evaluation equation (3.2) given our definition of r and g.

3. Assumption 3.3 holds for f defined in (3.40).

4. Assumptions 3.5–3.6 are assumed in the theorem statement.

5. Assumption 3.7 can be verified by noting that state and reward spaces are finite, $\rho_t(s)\ \forall s \in \mathcal{S}$ is bounded by Assumption 3.9, and that $\epsilon_{t+1} \to \mathbf{0}$ as $t \to \infty$ because $V_t \to v_\infty$.

Therefore all the conditions of Theorem 3.1 have been verified and we have $f(Q_n)$, which is $\bar{V}_t$ in the current setting, converges to $r_\#$, which is $d_\pi^\top v_\infty$, the solution of $\bar{r}$ in (3.8) given our choice of r, g. Because $V_t \to v_\infty$ and $\bar{V}_t \to d_\pi^\top v_\infty$ by Theorem 3.3, $V_t - \bar{V}_t\mathbf{1} \to v_\infty - d_\pi^\top v_\infty \mathbf{1} = v_\pi$ by Lemma 3.8 (ii).

Proof of Theorem 3.10

Because there is only one optimal policy, the solution set of q in the action-value optimality equation (3.7) only has one degree of freedom (i.e., every pair of solutions are different by a constant vector). Therefore the set $\mathcal{Q}_\infty$ only contains one element. Denote this element as q_∞.

Because (3.25)–(3.27) do not depend on (3.34)–(3.36), we have $Q_t \to q_\infty$ and $\bar{R}_t \to r_*$ a.s., according to Theorem 3.6.

Similar as the proof of Theorem 3.6, we can combine (3.34)–(3.36) and obtain a single update rule:

$$W_{t+1}(S_t, A_t) = W_t(S_t, A_t) + \beta_{\nu_t(S_t,A_t)}$$
$$\left(Q_t(S_t, A_t) - f(W_t) + W_t(S_{t+1}, \operatorname*{argmax}_a Q_t(S_{t+1}, a) - W_t(S_t, A_t)\right), \quad (3.41)$$

where $f : \mathbb{R}^{|\mathcal{S}|\times|\mathcal{A}|} \to \mathbb{R}$ satisfying $\forall w \in \mathbb{R}^{|\mathcal{S}|\times|\mathcal{A}|}$,

$$f(w) \doteq \eta \sum w - \kappa \sum W_0 + \bar{Q}_0. \quad (3.42)$$

We show that (3.42) is a special case of the General RVI Q's update (3.9). To see this point, recall $\{S'_t\}, \{R'_t\}, \{Y_t\}$ defined in the proof of Theorem 3.6. Equation 3.9 reduces to (3.42) by choosing $i = (s, a), n = t$,

$$r(i) = q_\infty(s, a),$$
$$g(w)(i) = \sum_{s'} p(s' \mid s, a) w(s', \pi_*(s')),$$
$$M_{n+1}(i) = W_t(S'_t(s, a), \pi_*(S'_t(s, a)))) - g(Q_t)(s, a),$$
$$\epsilon_{n+1}(i) = Q_t(s, a) - q_\infty(s, a)$$
$$+ W_t(S'_t(s, a), \operatorname*{argmax}_{a'} Q_t(S'_t(s, a), a')) - W_t(S'_t(s, a), \pi_*(S'_t(s, a)))).$$

We now show that the assumptions required by General RVI Q are all satisfied.

1. Assumption 3.1 can be verified for $g(q)(s, a) = \sum_{s',r} p(s', r \mid s, a) \max_{a'} q(s', a')$ easily.

2. Assumptions 3.3 holds for f defined in (3.42).

3. Assumptions 3.3–3.6 are assumed in the theorem statement.

4. Assumption 3.7 can be verified by noting that state and reward spaces are finite and that $\epsilon_{t+1} \to \mathbf{0}$ a.s. because $Q_t \to q_\infty$ a.s. and $\pi_t \in \Pi^D_*$ for sufficiently large t by Theorem 3.6.

In order to verify Assumption 3.2, first note that, given our choices of r and g, (3.8) reduces to

$$w(s,a) = q_\infty(s,a) - \bar{r} + \sum_{s',r} p(s', r \mid s, a) w(s', \pi_*(s')).$$

The above equation becomes the action-value evaluation equation (3.5) of the policy π_* in an MDP $\hat{\mathcal{M}}$ that is the same as the original one, except that the reward setting is replaced by q_∞.

The Markov chain induced by π_* in $\hat{\mathcal{M}}$ must be unichain. The reason is stated as follows. Suppose π_* is not unichain in $\hat{\mathcal{M}}$, then it is not unichain in $\mathcal{M}$ and induces more than one recurrent class. All of these recurrent classes share the same reward rate, which is the optimal one. Also, note that there exists a path from one class to the other class because $\mathcal{M}$ is weakly communicating. Now, we can construct another optimal policy, which moves from states in one recurrent class to states in the other recurrent class and stays in the latter chain. This contradicts our assumption of a single optimal policy in $\mathcal{M}$.

Because π_* is *unichain* in $\hat{\mathcal{M}}$, the solution of $\bar{r}$ in (3.5) for $\hat{\mathcal{M}}$ is uniquely $d_{\pi_*}^\top q_\infty$. The solution of q exists in (3.5) for any policy. Therefore Assumption 3.2 is verified. Assumption 3.4 can be easily verified by noting that $\mathbf{0}$ is a member of $\mathcal{Q}_\#$, and $\mathcal{Q}_\#$ only has one member by noting that all solutions of q in (3.5) are only different by a constant, the solution of $\bar{r}$ is unique in (3.5), and Assumption 3.3(ii).

Therefore Theorem 3.1 applies and we conclude that $\bar{Q}_t$ converges to $d_{\pi_*}^\top q_\infty$. Combining this result with $Q_t \to q_\infty$ (Theorem 3.3) and Lemma 3.8, we conclude that $Q_t - \bar{Q}_t \mathbf{1} \to q_\infty - d_{\pi_*}^\top q_\infty \mathbf{1} = q_*$ a.s.

3.8 Centering Experiments

This section presents two sets of experiments testing if Centered Differential TD-learning and Centered Differential Q-learning indeed learn the actual differential value function. The baseline algorithms are their uncentered versions introduced in Section 3.3 and Section 3.5. The test domain is still the continuing four-room domain (Figure 3.1).

The first set of experiments is for Centered Differential TD-learning and its uncentered version. The experiment setup is the same as in Section 3.4. For both algorithms, the parameters of the second system of updates were chosen to be the same as those used in the first system. That is, $\alpha_n = \beta_n$ and $\eta = \kappa$.

Because the estimated value function is expected to converge to the actual differential value function, rather than the differential value function plus some constant vector, the relative value errors introduced in the above two sections are not appropriate to measure the progress of learning. To measure the learning progress, I used the *value error*, defined as follows,

$$\text{Value Error}_t \doteq |V_t(S_t) - v_\pi(S_t)|$$

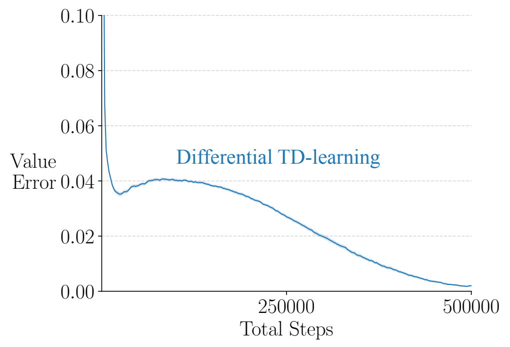
for Differential TD-learning, and

$$\text{Value Error}_t \doteq |V_t(S_t) - \bar{V}_t\mathbf{1} - v_\pi(S_t)|$$

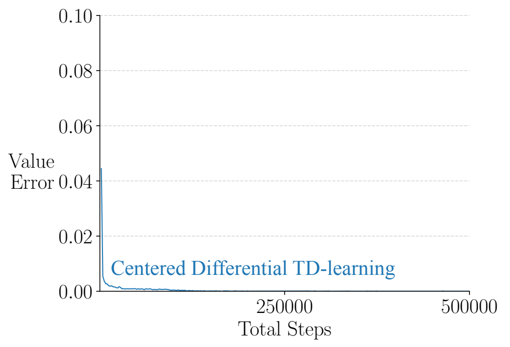
for Centered Differential TD-learning (note that $V_t(S_t) - \bar{V}_t\mathbf{1}$ is the output of the centered algorithm). In both definitions of value errors, π is the target policy (the policy being evaluated).

Figure 3.9a and Figure 3.9b show the value error learning curves of Differential TD-learning and Centered Differential TD-learning, with their parameter settings chosen to minimize the value error over the last $10,000$ steps. The value error of Centered Differential TD-learning diminished to zero eventually, in accordance with Theorem 3.8. It might seem to be surprising that the asymptotic value error of Differential TD-learning also is also close to zero. In fact, this is because the parameter setting was chosen to be the one that resulted in the lowest value error asymptotically and Differential TD-learning with some parameter setting resulted in a near-zero value error.

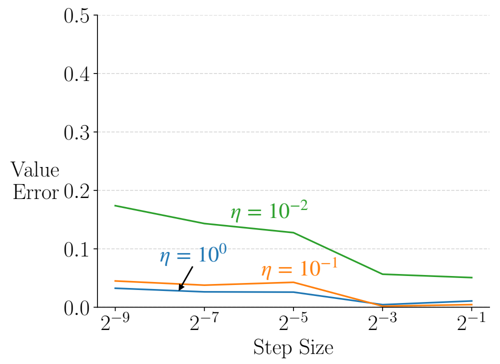
To better understand the difference between the two algorithms, I show parameter studies for Differential TD-learning and Centered Differential TD-learning in Figure 3.9c and Figure 3.9d, respectively. Each point in a curve is the value error average of the last $10,000$ steps and thus reflecting the algorithms' asymptotic performance. It can be seen that Centered Differential TD-learning achieved a close-to-zero value error for a wide range choice of



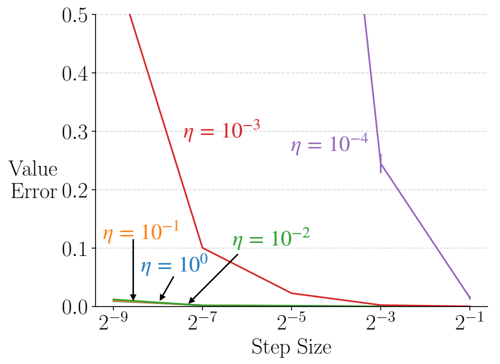
(a) The best learning curve for Differential TD-learning. The parameter settings were chosen in the same way as in Section 3.4.



(b) The best learning curve for Centered Differential TD-learning. The parameter settings were chosen in the same way as in Section 3.4.



(c) Parameter sensitivity study of Differential TD-learning; the curves corresponding to $\eta = 10^{-3}$ and $\eta = 10^{-4}$ have errors higher than 0.5 and are therefore not visible. Each point is the average value error over the last 10,000 steps. Error bars denote one standard error.

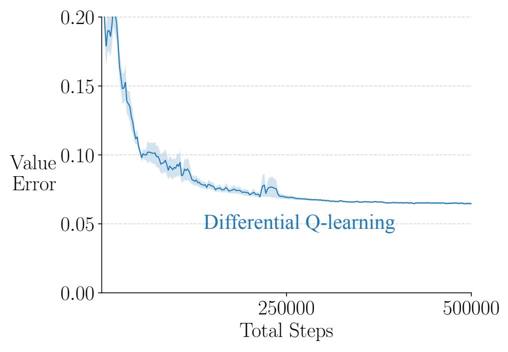


(d) Parameter sensitivity of Centered Differential TD-learning. Each point is the average value error over the last 10,000 steps. Error bars denote one standard error.

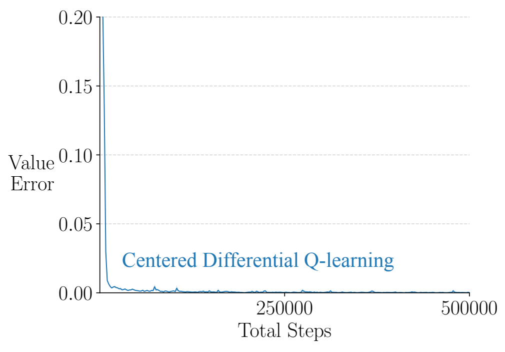
Figure 3.9: Plots showing learning curves and parameter sensitivity curves for Centered Differential TD-learning.

ηs. Surprisingly, Differential TD-learning also achieved a near-zero value error when $\eta = 1$ or 0.1. To see why this happened, let's compute the value error of the final solution of Differential TD-learning, given that $V_0 = 0$ and $\bar{R}_0 = 0$ in this domain. Note that because (3.22) is satisfied, $r(\pi) = \eta \sum_s v_\infty(s) \implies \sum_s v_\infty(s) = r(\pi)/\eta$. In this problem, $r(\pi) = 1/17$, therefore $\sum_s v_\infty(s) = \frac{1}{17\eta}$. Using this equation and a series of derivations, we can then obtain $\sum_s d_\pi(s) v_\infty(s) \approx \frac{1}{1768\eta} - 0.0136$. We also know that $\sum_s d_\pi(s) v_\pi(s) = 0$

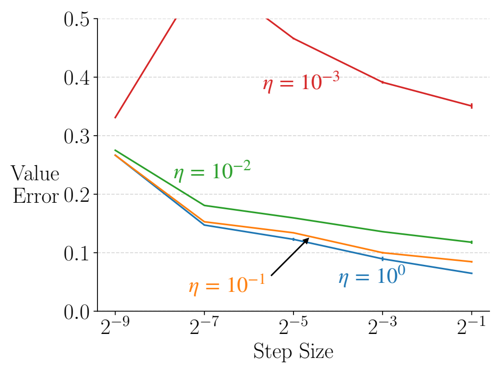
and the average value error over the last 10,000 steps should be approximately $\sum_s d_\pi(s)|v_\pi(s) - v_\infty(s)| = |\sum_s d_\pi(s)(v_\pi(s) - v_\infty(s))| = |\sum_s d_\pi(s)v_\infty(s)| = |\frac{1}{1768\eta} - 0.0136|$. Here the first equality holds because v_π and v_∞ are just different by a constant and d_π is non-negative. Choosing $\eta = 1$, we have the average value error ≈ 0.013. Choosing $\eta = 0.1$, we have the error ≈ 0.008. Choosing $\eta = 0.01$, we have the error ≈ 0.043. These values roughly mirror the value errors shown in Figure 3.9c with step size 2^{-1} or 2^{-3}.



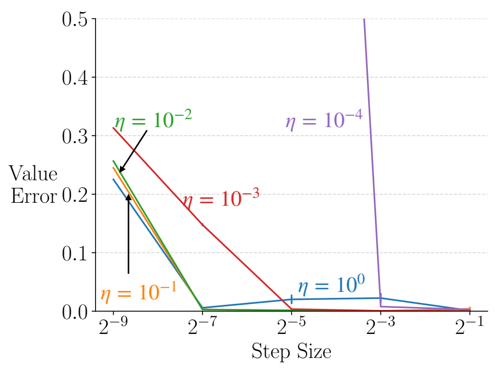
(a) A learning curve for Differential Q-learning. The parameter settings were chosen in the same way as in Section 3.6.



(b) A learning curve for Centered Differential Q-learning. The parameter settings were chosen in the same way as in Section 3.6.



(c) Parameter sensitivity of Differential Q-learning. Each point is the average value error over the last 10,000 steps. Error bars denote one standard error.



(d) Parameter sensitivity of Centered Differential Q-learning. Each point is the average value error over the last 10,000 steps. Error bars denote one standard error. each point is the average value error over the last 10,000 steps. Error bars denote one standard error.

Figure 3.10: Plots showing learning curves and parameter sensitivity curves for Centered Differential Q-learning.

The experiment setup testing Differential Q-learning and Centered Differential Q-learning is the same as in Section 3.6. The metric measuring the progress of value learning for the control algorithms is different and is tricky to define. Note that in this domain, there is more than one optimal policy and thus the assumption on the uniqueness of the optimal policy made in Theorem 3.10 is not satisfied. However, this domain is somewhat special because the differential action-value function of any greedy policy w.r.t. any solution of the action-value optimality equation, q_*, does not depend on the choice of the greedy policy and the choice of the solution. To see this point, note that any solution q of the action-value optimality equation in this domain can be expressed as $q(s,a) = \sum_{s',r} p(s', r \mid s, a) v(s') + c, \ \forall s, a$, where $v(s) \doteq -\frac{1}{17} \times$ number of steps to reach the green cell from s, and c is some constant. Therefore any greedy policy w.r.t. q takes an action that results in a state one step closer to the green cell. Finally, the differential action-value function of any greedy policy q_* satisfies $q_*(s,a) = q(s,a) - \frac{1}{17}(-\frac{0}{17} + c - \frac{1}{17} + c - \ldots - \frac{16}{17} + c) = q(s,a) - c - \frac{8}{17} = \sum_{s',r} p(s', r \mid s, a) v(s') - \frac{8}{17}$. Thus, q_* does not depend on the choice of the greedy policy or the choice of the solution of (3.7). Therefore, even if the theory does not guarantee convergence in this case, as long as $Q_t - \bar{Q}_t \mathbf{1}$ converges, it must converge to q_*.

Based on the above discussion, we can now define the error metric. Just like what I did for the two centered prediction algorithms, for Differential Q-learning, the value error is defined as

$$\text{Value Error}_t \doteq |Q_t(S_t, A_t) - q_*(S_t, A_t)|,$$

and for Centered Differential Q-learning, we have

$$\text{Value Error}_t \doteq |Q_t(S_t, A_t) - \bar{Q}_t \mathbf{1} - q_*(S_t, A_t)|.$$

Figure 3.10a and Figure 3.10b show the value error learning curves of Differential Q-learning and Centered Differential Q-learning, with the parameter settings chosen in the same way as in Section 3.6. Each point is the average of value errors in the 2000 previous steps. The value error in Centered Differential Q-learning diminished to zero eventually while the value error in Differential Q-learning failed to do so.

Parameter studies for Differential Q-learning and Centered Differential Q-learning are shown in Figure 3.10c and Figure 3.10d, respectively. The parameter setting used to generate learning curves were those resulting in the lowest value error over the last $10,000$ steps. Centered Differential Q-learning achieved near zero value error asymptotically for a wide range of parameter settings while Differential Q-learning failed to achieve a near zero value error, regardless of the choice of parameter setting. These results show that even if Theorem 3.10 does not guarantee convergence in this domain, Centered Differential Q-learning still converged to the desired point.

3.9 Summary

This chapter introduced a family of tabular algorithms for the average-reward formulation as well as the algorithms' convergence theories.

The average-reward sub-problems addressed by these algorithms, according to the taxonomy provided in Section 1.3, include all tabular sub-problems except for model-based learning problems, on-policy model-free learning problems, and problems that involve options.

The most distinctive feature of algorithms introduced in this chapter is that these algorithms all estimate the reward rate using the TD error while most previous approaches either do not maintain an estimate of the reward rate or update the reward rate estimate using the conventional reward error. The experiments testing Differential Q-learning, while limited in its complexity, showed that applying the TD error to update the reward rate estimate may result in an algorithm that is less sensitive to hyper-parameters.

The family of algorithms can be divided into centered ones and uncentered ones, depending on whether the estimated differential value function has a constant offset. The centered algorithms were derived by applying the centering technique to uncentered algorithms. In the average-reward literature, it is common for algorithms to be uncentered, because this offset does not change the greedy policies derived from these estimated differential value functions. I conjecture that removing offsets may have benefits such as achieving faster learning. However, these potential benefits were not explored in this chapter.

The key to the convergence theories of the algorithms is the convergence theory of General RVI Q, which encompasses all of the algorithms introduced in this chapter as special cases. In addition, by applying this convergence theory to RVI Q-learning (Abounadi et al. 2001), which was originally proved to converge in unichain MDPs, this chapter established its convergence in the more general weakly communicating MDPs. The convergence theory of General RVI Q extends that of RVI Q-learning by Abounadi et al. (2001) in several ways, as discussed in Section 3.2. This convergence theory will continue to be the core technical tool to show the convergence of algorithms introduced in the next chapter.

Chapter 4

Temporal Abstraction with Options

This chapter presents the second area of contributions of this dissertation: an extension of the family of average-reward algorithms introduced in the previous chapter from primitive actions to *options*, which are temporally abstracted courses of action. The idea of options was originally introduced by Sutton, Precup, and Singh (1999) as a way to achieve temporal abstraction. Intuitively, if actions are likened to muscle twitches, options can be thought of as more complex behaviors, such as rinsing a plate, which is composed of a sequence of muscle twitches. Using options instead of actions alone can benefit learning and planning in at least three ways. Firstly, planning with a model that predicts the outcomes of options can be considerably faster. Secondly, exploration can be more effective when following options. Finally, human experts may impart prior knowledge to the agent by appropriately specifying options. The paper by Sutton, Precup, and Singh (1999) further explores options algorithms and theories concerning the *discounted* formulation. The algorithms and theories introduced in this chapter can be considered as an extension of theirs to the *average-reward* formulation.

Algorithms and theories introduced in this chapter work with a predetermined set of options. These options can either be defined by experts or learned to achieve goals specified by humans. However, when such options are not available, the agent must undertake the task of discovering them. Significant progress has been made to address this option discovery problem over the past two decades. Readers interested in exploring this topic further may refer to Pateria et al.'s (2021) comprehensive survey.

To present the options algorithms and theories, this chapter first introduces some background knowledge about Semi-MDPs (SMDPs), which are tightly related to options. This chapter then presents a new SMDP theory, which is used by theories of many algorithms introduced in the current chapter. This chapter sets up the tabular off-policy learning and planning problems with options in average-reward MDPs. For both off-policy learning and planning problems, this chapter introduces inter- and intra-option algorithms, proves their convergence, and demonstrates their empirical performance using experiments. Because inter-option planning algorithms require using an option model, this chapter also introduces convergent intra-option learning and planning algorithms to obtain the option model. Finally, this chapter demonstrates that by interrupting the agent's behavior and switching from the current option to a new one, the agent can achieve a behavior that results in a higher reward rate.

4.1 SMDP Preliminaries

This section introduces basic definitions and results in average-reward SMDPs, which will be useful throughout this chapter.

SMDPs extend MDPs by adding the holding time (length) of each action. A finite SMDP is defined by the tuple $(\mathcal{S}, \mathcal{A}, \mathcal{R}, \mathcal{L}, d_0, p)$, where $\mathcal{S}$ is a finite set of states, $\mathcal{A}$ is a finite set of actions, $\mathcal{R}$ is a countably infinite set of rewards, $\mathcal{L}$ is a countably infinite set of time steps, $d_0 \in \Delta(\mathcal{S})$ is the initial state distribution, and $p : \mathcal{S} \times \mathcal{A} \to \Delta(\mathcal{S} \times \mathcal{R} \times \mathcal{L})$ is the transition function of the SMDP with $p(s', r, l \mid s, a)$ defined as the probability of transitioning to state $s' \in \mathcal{S}$ using $l \in \mathcal{L}$ time steps and observes a reward $r \in \mathcal{R}$ when taking action $a \in \mathcal{A}$ from state $s \in \mathcal{S}$.

An SMDP and a policy $\pi \in \Pi$ induce a sequence of tuples of random variables $(S_n, A_n, R_{n+1}, L_{n+1})$ where the transition probability is governed by p and π. That is, $\Pr(S_{n+1} = s', R_{n+1} = r, L_{n+1} = l \mid S_n = s, A_n = a) = p(s', r, l \mid$

$s, a)$ and $\Pr(A_n = a \mid S_n = s) = \pi(a \mid s)$. Let $T_n = \sum_{i=1}^{n} L_n$ be the time step at which the n-th transition is finished. Let $N_t = \max_n \mathbf{I}\{T_n \leq t\}$ be a random variable denoting the number of transitions up to time t. The *reward rate* of a policy $\pi \in \Pi$ given a starting state s can be defined as

$$r(\pi, s) \doteq \lim_{t \to \infty} \frac{\mathbb{E}_\pi \left[\sum_{i=1}^{N_t} R_i \mid S_0 = s \right]}{t}, \tag{4.1}$$

where the limit always exists by Equation 18 in the paper by Denardo (1971).

For a cleaner presentation, the following definitions will be used in this chapter. Let the expected reward of each state-action pair (s, a) be denoted as $r(s, a)$. That is

$$r(s, a) = \sum_{s', r, l} p(s', r, l \mid s, a) r. \tag{4.2}$$

Similarly, let the expected length of each state-action pair (s, a) be denoted as $l(s, a)$. That is

$$l(s, a) = \sum_{s', r, l} p(s', r, l \mid s, a) l. \tag{4.3}$$

With a bit of abuse of notation let

$$p(s' \mid s, a) \doteq \sum_{r, l} p(s', r, l \mid s, a).$$

Let the expected reward starting from state s under policy π be denoted as $r_\pi(s)$. That is

$$r_\pi(s) = \sum_a \pi(a \mid s) r(s, a). \tag{4.4}$$

Let the expected action length starting from state s under policy π be denoted as $l_\pi(s)$. That is

$$l_\pi(s) = \sum_a \pi(a \mid s) l(s, a). \tag{4.5}$$

Let the transition matrix under policy π be denoted as P_π. That is

$$P_\pi(s, s') = \sum_a \pi(a \mid s) p(s' \mid s, a). \tag{4.6}$$

We say a policy $\pi \in \Pi$ is unichain if the SMDP's transition matrix P_π is unichain. We say an SMDP is communicating/weakly communicating if the MDP with state space $\mathcal{S}$, action space $\mathcal{A}$, reward space $\mathcal{R}$, and transition function $\sum_l p(s, r, l \mid s, a)$ is communicating/weakly communicating.

If a policy π is unichain, the reward rate of π does not depend on the starting state and hence we can denote it by just $r(\pi)$. The differential action-value function for a policy π is defined for all $s \in \mathcal{S}, a \in \mathcal{A}$ as

$$q_\pi(s,a) \doteq \lim_{n\to\infty} \frac{1}{n} \sum_{k=1}^{n} \sum_{t=1}^{k} \mathbb{E}_\pi[R_t - r(\pi) \mid S_0 = s, A_0 = a, A_{1:t-1} \sim \pi].$$

The *evaluation* equation for SMDPs:

$$q(s,a) = r(s,a) - \bar{r} \cdot l(s,a) + \sum_{s'} p(s' \mid s,a) \sum_{a'} \pi(a' \mid s') q(s',a'),$$

where q and $\bar{r}$ denote estimates of the action-value function and the reward rate respectively. Just as in the MDP case, if π is unichain, the SMDP evaluation equation has a unique solution of $\bar{r}$ — $r(\pi)$ and a unique solution of q only up to a constant (Puterman 1994).

If the SMDP is weakly communicating, the optimal reward rate

$$r_* \doteq \sup_{\pi\in\Pi} r(\pi, s)$$

does not depend on the start state s. In addition, r_* is the unique solution of $\bar{r}$ in the SMDP optimality equation:

$$q(s,a) = r(s,a) - \bar{r} \cdot l(s,a) + \sum_{s'} p(s' \mid s,a) \max_{a'} q(s',a'). \tag{4.7}$$

Denote the solution set of q in the above equation as $\mathcal{Q}$. Unlike the evaluation case, it has been shown by Schweitzer and Federgruen (1978) (I will use a shorthand S&F for their work from now on because I will need to refer to this work multiple times) that $\mathcal{Q}$ may have multiple degrees of freedom (members are not different by a constant vector). Their paper also characterizes the number of degrees of freedom of $\mathcal{Q}$ and shows that $\mathcal{Q}$ is non-empty, closed, unbounded, connected, and possibly non-convex. These results are nowadays perceived as fundamental in average-reward MDPs and SMDPs.

In fact, the results by S&F were developed for the *state-value* optimality equation

$$v(s) = \max_a \left\{ r(s,a) - \bar{r} \cdot l(s,a) + \sum_{s'} p(s' \mid s,a) v(s') \right\}, \tag{4.8}$$

rather than the action-value optimality equation (4.8). I now use the following transformation to show that the results by S&F also hold for the action-value optimality equation. This transformation shows that the action-value optimality equation (4.7) for any weakly communicating SMDP can be viewed as the state-value optimality equation (4.8) for some other weakly communicating SMDP.

Given an SMDP $\hat{\mathcal{M}}$ and its (4.7), define a function $v : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ such that $v((s,a)) \doteq q(s,a),\ \forall s, a$. Then (4.7) can be written as

$$\begin{aligned} v((s,a)) &= r(s,a) - \bar{r} \cdot l(s,a) + \sum_{s'} p(s' \mid s,a) \max_{a'} v((s',a')) \\ &= \max_{\pi \in \Pi^D} \left\{ \sum_{s',a'} \tilde{p}((s',a'),r,l \mid (s,a),\pi)(r - \bar{r} \cdot l + v((s',a'))) \right\}, \forall\ s \in \mathcal{S}\ a \in \mathcal{A}, \end{aligned} \tag{4.9}$$

where $\Pi^D \subseteq \Pi$ is the set of deterministic policies, $\tilde{p}((s',a'),r,l \mid (s,a),\pi) \doteq p(s',r,l \mid s,a)\mathbf{I}\{\pi(s') = a'\}$, where $\pi(s')$ is the unique action that the deterministic policy π assigns to state s'. The above equation is just the same as (4.8) but for a new SMDP $\hat{\mathcal{M}}'$ with $\mathcal{R}$ and $\mathcal{L}$ being the same as those of $\hat{\mathcal{M}}$, the state space being $\mathcal{S} \times \mathcal{A}$, the action space being Π^D, and the transition function being $\tilde{p}$. Finally, one can verify that $\hat{\mathcal{M}}'$ is also weakly communicating by showing that 1) if two states s, s' in $\hat{\mathcal{M}}$ communicate, for any $a, a' \in \mathcal{A}$, two states $(s,a), (s',a')$ in $\hat{\mathcal{M}}'$ communicate, 2) if a state s is transient in $\hat{\mathcal{M}}$, for any $a \in \mathcal{A}$, (s,a) is transient in $\hat{\mathcal{M}}'$, and 3) the communicating class is unique and closed.

4.2 A Sub-Optimality Bound in Weakly Communicating SMDPs

This section presents one of my contributed results. This result concerns weakly communicating SMDPs and is independent of options algorithms. The result bounds the sup-optimality of the reward rate achieved by greedy policies w.r.t. an arbitrary $q \in \mathbb{R}^{|\mathcal{S}|\times|\mathcal{A}|}$. This result is a generalization of Theorem 8.5.5 by Puterman (1994) from unichain MDPs to weakly communicating SMDPs.

For an arbitrary policy $\pi \in \Pi$, the induced reward rate varies across different start states. Let's denote the vector of reward rates for all start states by $r(\pi) = [r(\pi, 1), r(\pi, 2), \ldots, r(\pi, |\mathcal{S}|)]$ (labeling members in $\mathcal{S}$ by $1, 2, \ldots, |\mathcal{S}|$). However,

Proposition 4.1. *Given a weakly communicating SMDP and a vector $q \in \mathbb{R}^{|\mathcal{S}|\times|\mathcal{A}|}$, let π be a greedy policy w.r.t. q (i.e., $\pi(s) \in \operatorname{argmax} q(s, \cdot)$). Define an operator T as follows:*

$$Tq(s, a) \doteq r(s, a) + \sum_{s'} p(s' \mid s, a) \max_{a'} q(s', a'). \tag{4.10}$$

Then,

$$\max_s \{r_* - r(\pi, s)\} \leq sp\left(\frac{TQ - Q}{l}\right),$$

where $sp(x) = \max_i x(i) - \min x(i)$ denotes the span of vector x, and $\frac{a}{b}$ denotes the component-wise division of two vectors a, b of the same size.

To prove the above result, we shall first prove the following lemma.

Lemma 4.1. *For any $m \in \{1, 2, 3, \ldots\}$, $a, b \in \mathbb{R}^m$, $b > 0$, and for any $p \in \mathbb{R}^m$ such that $p \geq 0$, $\sum_s p(s) = 1$,*

$$\min_s \frac{a(s)}{b(s)} \leq \frac{p^\top a}{p^\top b} \leq \max_s \frac{a(s)}{b(s)}.$$

Proof. For any $s \in \{1, 2, 3, \ldots, m\}$,

$$\frac{a(s)}{b(s)} \geq \min_{s'} \frac{a(s')}{b(s')} \implies a(s) \geq b(s) \min_{s'} \frac{a(s')}{b(s')} \implies p(s)a(s) \geq p(s)b(s) \min_{s'} \frac{a(s')}{b(s')}.$$

Therefore, $p^\top a \geq p^\top b \min_{s'} \frac{a(s')}{b(s')}$. By our assumptions on b and p, $p^\top b > 0$, we have $\frac{p^\top a}{p^\top b} \geq \min_{s'} \frac{a(s')}{b(s')}$. $\frac{p^\top a}{p^\top b} \leq \max_s \frac{a(s)}{b(s)}$ can be shown in the same way. $\square$

I now show the proof of Proposition 4.1.

Proof.

It is known (see, e.g., S&F's Lemma 2.3) that for any $\pi \in \Pi$, $r(\pi)$ is the unique solution of $\bar{r} \in \mathbb{R}^{|\mathcal{S}|}$ in the following system of equations:

$$v = r_\pi - H_\pi \bar{r} + P_\pi v, \tag{4.11}$$

$$\bar{r} = P_\pi \bar{r}, \tag{4.12}$$

where $v, \bar{r} \in \mathbb{R}^{|\mathcal{S}|}$ are two vectors of unknown variables, $H_\pi \in \mathbb{R}^{|\mathcal{S}| \times |\mathcal{S}|}$ is defined as follows:

$$H_\pi(s, s') \doteq \sum_a \pi(a \mid s) \sum_{r,l} p(s', r, l \mid s, a) l.$$

It is also known that the solution set of v is not empty by S&F's Lemma 2.3.

I now transform (4.11) and (4.12) into the action-value form, from which it is easier to proceed. Fix a solution of v, v_*, let

$$\begin{aligned} g_\pi(s, a) &\doteq \sum_{s'} p(s' \mid s, a) r(\pi, s'), \\ q_*(s, a) &\doteq r(s, a) - \tilde{H}_\pi g_\pi(s, a) + \sum_{s'} p(s' \mid s, a) v_*(s'), \end{aligned}$$

where $\tilde{H}_\pi \in \mathbb{R}^{|\mathcal{S} \times \mathcal{A}| \times |\mathcal{S} \times \mathcal{A}|}$ is defined as follows:

$$\tilde{H}_\pi((s, a), (s', a')) \doteq \sum_{r,l} p(s', r, l \mid s, a) \pi(a' \mid s') l.$$

Then by (4.11)—(4.12):

$$\begin{aligned} q_* &= r - \tilde{H}_\pi g_\pi + \tilde{P}_\pi q_*, \\ g_\pi &= \tilde{P}_\pi g_\pi, \end{aligned}$$

where $\tilde{P}_\pi$ denote a $|\mathcal{S} \times \mathcal{A}| \times |\mathcal{S} \times \mathcal{A}|$ transition matrix. That is,

$$\tilde{P}_\pi((s, a), (s', a')) \doteq p(s' \mid s, a) \pi(a' \mid s').$$

Therefore (g_π, q_*) is a solution of (g, q) in the following system of equations

$$\begin{aligned} q &= r - \tilde{H}_\pi g + \tilde{P}_\pi q, \\ g &= \tilde{P}_\pi g. \end{aligned}$$

Just like (4.11)—(4.12), the above system of equations has a unique solution of g. Therefore g_π is this unique solution. Because P_π will not be used in this proof anymore, from now on, in this proof, I use P_π to denote $\tilde{P}_\pi$.

The above discussion is for arbitrary π. Now let π be a greedy policy of some $q \in \mathbb{R}^{\mathcal{S}\times\mathcal{A}}$, I will show that

$$\min_{s',a'} \frac{(Tq - q)(s', a')}{l(s', a')} \leq g_\pi(s, a) \leq r_* \leq \max_{s',a'} \frac{(Tq - q)(s', a')}{l(s', a')}, \forall s, a,$$

from which the desired bound

$$\max_s r_* - r(\pi, s) \leq sp\left(\frac{Tq - q}{l}\right)$$

can be obtained using $r(\pi, s) = \sum_a \pi(a \mid s) g_\pi(s, a)$ (by the definition of g_π).

To obtain the upper and lower bounds of g_π, I first write g_π in the following expression (S&F's Lemma 2.3):

$$g_\pi(s, a) = \sum_{m=1}^{n(\pi)} \phi_\pi^{(m)}(s, a) g_\pi^{(m)},$$

where $n(\pi)$ is the number of recurrent classes in the Markov chain induced by P_π, $\phi_\pi^m(s, a)$ is the probability of absorption in the m-th recurrent class, starting from state (s, a), and

$$g_\pi^{(m)} \doteq \frac{\sum_{s,a} d_\pi^{(m)}(s, a) r(s, a)}{\sum_{s,a} d_\pi^{(m)}(s, a) l(s, a)} = \frac{P_\pi^\infty r(s, a)}{P_\pi^\infty l(s, a)}$$
$$\text{for all } (s, a) \in \text{the } m\text{-th recurrent class},$$

where $d_\pi^{(m)}$ is the unique stationary distribution of P_π on the m-th recurrent class, and P_π^∞ is the *limiting matrix* of P_π:

$$P_\pi^\infty \doteq \lim_{n\to\infty} \frac{1}{n} \sum_{i=0}^{n-1} P_\pi^i. \tag{4.13}$$

Because $\mathcal{S}$ and $\mathcal{A}$ are finite, the above Cesaro limit exists and P_π^∞ is a stochastic matrix (has row sums equal to 1).

Note that, for any $s \in \mathcal{S}, a \in \mathcal{A}$,

$$g_\pi^{(m)} = \frac{P_\pi^\infty r(s, a)}{P_\pi^\infty l(s, a)} = \frac{P_\pi^\infty (r + P_\pi q - q)(s, a)}{P_\pi^\infty l(s, a)} \geq \min_{s',a'} \frac{(r + P_\pi q - q)(s', a')}{l(s', a')}$$
$$= \min_{s',a'} \frac{(Tq - q)(s', a')}{l(s', a')},$$

where the inequality holds because of Lemma 4.1 and the last equality holds because π is a greedy policy w.r.t. q. Therefore we have

$$\begin{aligned} g_\pi(s,a) &= \sum_{m=1}^{n(\pi)} \phi_\pi^{(m)}(s,a) g_\pi^{(m)} \geq \sum_{m=1}^{n(\pi)} \phi_\pi^{(m)}(s,a) \min_{s',a'} \frac{(Tq-q)(s',a')}{l(s',a')} \\ &= \min_{s',a'} \frac{(Tq-q)(s',a')}{l(s',a')}. \end{aligned} \tag{4.14}$$

On the other hand, consider a deterministic optimal policy $\pi_* \in \Pi_*^D$. For any s, a,

$$\begin{aligned} g_{\pi_*}^{(m)} &= \frac{P_{\pi_*}^\infty r(s,a)}{P_{\pi_*}^\infty l(s,a)} = \frac{P_{\pi_*}^\infty (r + P_{\pi_*} q - q)(s,a)}{P_{\pi_*}^\infty l(s,a)} \\ &\leq \max_{s',a'} \frac{(r + P_{\pi_*} q - q)(s',a')}{l(s',a')} \\ &\leq \max_{s',a'} \frac{(r + P_\pi q - q)(s',a')}{l(s',a')} \\ &= \max_{s',a'} \frac{(Tq-q)(s',a')}{l(s',a')}. \end{aligned}$$

The first inequality holds because of Lemma 4.1 and the second inequality holds because π is a greedy policy w.r.t. q. Then we have, for any s, a,

$$\begin{aligned} g_{\pi_*}(s,a) &= \sum_{m=1}^{n(\pi_*)} \phi_{\pi_*}^{(m)}(s,a) g^{(m)}(\pi_*) \\ &\leq \sum_{m=1}^{n(\pi_*)} \phi_{\pi_*}^{(m)}(s,a) \max_{s',a'} \frac{(Tq-q)(s',a')}{l(s',a')} = \max_{s',a'} \frac{(Tq-q)(s',a')}{l(s',a')}. \end{aligned}$$

Note that $g_{\pi_*}(s,a) = \sum_{s'} p(s' \mid s,a) r(\pi_*, s') = r_*$ where the second equation holds because $r(\pi_*, s) = r_*$ for all $s \in \mathcal{S}$ by the weakly communicating assumption. Therefore,

$$r_* \leq \max_{s',a'} \frac{(Tq-q)(s',a')}{l(s',a')}. \tag{4.15}$$

Combining (4.14) and (4.15), and noting that $g_\pi(s,a) = \sum_{s'} p(s' \mid s,a) r(\pi, s') \leq r_*$, we have, for any s, a,

$$\min_{s',a'} \frac{(Tq-q)(s',a')}{l(s',a')} \leq g_\pi(s,a) \leq r_* \leq \max_{s',a'} \frac{(Tq-q)(s',a')}{l(s',a')},$$

which is the desired bound.

□

4.3 Important Properties of the Solution Set

This section presents three new important results concerning the solution set of control algorithms introduced in the current and previous chapters.

Given an SMDP, all results concern the following set:

$$\mathcal{Q}_s \doteq \{q \in \mathbb{R}^{\mathcal{S}\times\mathcal{A}} : q \in \mathcal{Q}, q \text{ is a solution of } f(q) = r_*\}, \tag{4.16}$$

where $f : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ is a function satisfying Assumption 3.3 and r_* is the optimal reward rate of the SMDP.

Result 1: A Special Case in which the Solution Set Has a Single Degree of Freedom

The next lemma shows that $\mathcal{Q}$ is the set of constant vectors if all state-action pairs share the same "pair-reward rate" defined as

$$w(s, a) \doteq \frac{r(s, a)}{l(s, a)} \ \forall s, a, \tag{4.17}$$

where $r(s, a)$ and $l(s, a)$ are defined in (4.2) and (4.3), respectively. This pair-reward rate is independent of the policy and should not be confused with a policy's reward rate.

Lemma 4.2. *If the SMDP is weakly communicating and $w(s, a) = w(s', a')$ for all $s, s' \in \mathcal{S}, a, a' \in \mathcal{A}$, $\mathcal{Q} = \{c\boldsymbol{1} \mid c \in \mathbb{R}\}$.*

Proof. Given that all pair-reward rates are the same, any policy's reward rate is also this shared pair-reward rate. And because the solution of $\bar{r}$ in (4.7) is the average-reward rate of optimal policies, it is also this shared pair-reward rate. Then (4.7) reduces to,

$$q(s, a) = \left\{\sum_{s'} p(s' \mid s, a) \max_{a'} q(s', a')\right\}, \quad s \in \mathcal{S}.$$

Following a similar transformation as in the discussion around (4.9), the above equation can be rewritten as

$$q(s, a) = \max_{\pi\in\Pi^D} \left\{\sum_{s',a'} \tilde{p}((s', a') \mid (s, a), \pi) q(s', a')\right\}, \quad s \in \mathcal{S}, \tag{4.18}$$

where $\tilde{p}((s', a') \mid (s, a), \pi) = p(s' \mid s, a)\mathbf{I}\{\pi(s') = a'\}$. But this is just the state-value optimality of a weakly communicating MDP with state space $\mathcal{S} \times \mathcal{A}$, action space Π^D, all rewards being zero, and the probability of transitioning from state (s, a) and action $\pi \in \Pi^D$ to state (s', a') being $\tilde{p}((s', a') \mid (s, a), \pi)$.

Consider a set of states in this new weakly communicating MDP,

$$S_{\min} \doteq \left\{ i \in \mathcal{S} \times \mathcal{A}, q(i) = \min_{i'} q(i') \right\}$$

Then by (4.18), there is a zero probability of transiting from a state $i \in S_{\min}$ to a state $i' \notin S_{\min}$, regardless of the action chosen, in this MDP. Therefore $S_{\min}$ is a closed set. Because this MDP is weakly communicating, any closed set of states contains the unique communicating class. Therefore all states in the communicating class are in $S_{\min}$ and share the same value $\min_i q(i')$. The values of transient states are the same as the shared value of communicating states by Equation 4.5 by S&F and noting that all transient states are not in R^*.

□

The next corollary of Lemma 4.2 characterizes $\mathcal{Q}_s$ and is what I need for the proof of Theorem 3.6 and for the proof for options control algorithms to be introduced later in this chapter.

Corollary 4.1. *If an SMDP is weakly communicating, f satisfies Assumption 3.3, and $w(s, a) = f(\mathbf{0}), \forall s \in \mathcal{S}, a \in \mathcal{A}$, $\mathcal{Q}_s = \{\mathbf{0}\}$.*

Proof. Because all pair-reward rates are $f(\mathbf{0})$, $r_* = f(\mathbf{0})$. Combining this result with $r_* = f(q)$ for all $q \in \mathcal{Q}_s$, we have $f(q) = f(\mathbf{0})$ for all $q \in \mathcal{Q}_s$. Combining this result with all $q \in \mathcal{Q}_s \subseteq \mathcal{Q}$ being a constant vector (by Lemma 4.2) and Assumption 3.3 (ii), we have $\mathcal{Q}_s = \{\mathbf{0}\}$.

□

Result 2: Characterization of the Solution Set

Theorem 4.1. *If the SMDP is weakly communicating and Assumption 3.3 holds, $\mathcal{Q}_s$ is non-empty, compact, connected, and possibly non-convex.*

Remark: Recall that in the previous chapter, I stated Theorem 3.5 without providing its proof. Theorem 4.1 generalizes Theorem 3.5 because SMDPs generalize MDPs. And I show the proof of Theorem 4.1 here in this section.

Remark: S&F showed that the set of solutions of q in (4.7) is non-empty, closed, unbounded, connected, and possibly non-convex. The proof of Theorem 4.1 builds upon their result.

The rest of this section proves the above theorem.

Non-emptiness and Closedness

Let's divide both hand sides of (4.7) by $l(s,a)$ where $l(s,a) > 0$ is defined in (4.3),

$$0 = \frac{r(s,a) - \bar{r} \cdot l(s,a) + \sum_{s'} p(s' \mid s,a) \max_{a'} q(s',a') - q(s,a)}{l(s,a)}, \forall s, a.$$

The above equation is equivalent to (4.7) and thus has the same solutions of $(q, \bar{r})$ as (4.7).

Now rewrite the above equation as follows

$$\begin{aligned} 0 &= \frac{r(s,a)}{l(s,a)} - \bar{r} + \frac{\sum_{s'} p(s' \mid s,a) \max_{a'} q(s',a') - q(s,a)}{l(s,a)} \\ &= w(s,a) - \bar{r} + g(q)(s,a) - q(s,a), \forall s, a, \end{aligned} \tag{4.19}$$

where w is defined in (4.17), and

$$g(q)(s,a) \doteq \frac{\sum_{s'} p(s' \mid s,a) \max_{a'} q(s',a')}{l(s,a)} + \frac{l(s,a) - 1}{l(s,a)} q(s,a) \quad \forall s, a.$$

One can now show that (4.19) is a special case of (3.8) by choosing $\mathcal{I} = \mathcal{S} \times \mathcal{A}$, verifying that g satisfies Assumption 3.1, and verifying Assumption 3.2 using the weakly communicating SMDPs facts introduced right after (4.7). In addition, because r_* is the unique solution of $\bar{r}$ in (4.7), $\mathcal{Q}_\#$ reduces to $\mathcal{Q}_s$ in the current setting. Therefore, applying Lemma 3.1, we have $\mathcal{Q}_s$ is non-empty and closed.

Boundedness

We shall prove by contradiction. Suppose $\mathcal{Q}_s$ is unbounded, there exists a sequence $x_n \in \mathcal{Q}_s$ such that $m_n \doteq \|x_n\|_\infty \to \infty$ as $n \to \infty$. Let $y_n \doteq x_n / \|x_n\|_\infty$, then $\|y_n\|_\infty = 1$. We choose a subsequence of x_n, x_{n_l}, such that the corresponding subsequence of y_n, y_{n_l} is convergent. The existence can be seen by noting that the solution set of $y \in \mathbb{R}^{|\mathcal{S}|\times|\mathcal{A}|}$ in $\|y\|_\infty = 1$ is compact and any sequence in a compact set has a convergent subsequence. With a bit of abuse of notation, from now on, I use x_n and y_n to denote the two subsequences instead of the original sequences to simplify notations.

By $x_n \in \mathcal{Q}_s$, we have

$$x_n(s,a) = r(s,a) - r_* \cdot l(s,a) + \sum_{s'} p(s' \mid s,a) \max_{a'} x_n(s',a'), \forall\ s \in \mathcal{S}, a \in \mathcal{A}$$
$$f(x_n) = r_*.$$

Divide both two equations by m_n, we have

$$y_n(s,a) = \frac{r(s,a) - r_* \cdot l(s,a)}{m_n} + \sum_{s'} p(s' \mid s,a) \max_{a'} y_n(s',a'), \forall\ s \in \mathcal{S}$$
$$f(y_n) = f(\mathbf{0}) + \frac{r_* - f(\mathbf{0})}{m_n},$$

where the second equation uses Assumption 3.3(iii). Let y_∞ be the point y_n converges to as $n \to \infty$. Take $n \to \infty$ in the above two equations, we have

$$y_\infty(s,a) = \sum_{s',r,l} p(s',r,l \mid s,a) \max_{a'} y_\infty(s',a'), \forall\ s \in \mathcal{S} \tag{4.20}$$
$$f(y_\infty) = f(\mathbf{0}), \tag{4.21}$$

where the second equation holds because f is continuous (Assumption 3.3(i)) and $y_n \to y_\infty$.

The first equation is the action-value optimality equation of an SMDP (4.7) with all pair reward rate being $f(\mathbf{0})$ (i.e., $w(s,a) = f(\mathbf{0})$ where w is defined in (4.17)). Therefore optimal reward rate r_* for this SMDP is also $f(\mathbf{0})$. Therefore $\mathcal{Q}_s$ for this SMDP is defined by (4.20) and (4.21). We can now apply Corollary 4.1 to conclude y_∞ equals to $\mathbf{0}$. But this contradicts our assumption of $\|y_n\|_\infty = 1$. Therefore $\mathcal{Q}_s$ is bounded.

Connectedness

Define a function that takes a $q \in \mathcal{Q}$ as input and produces an element in $\mathcal{Q}_s$ as output. Specifically, let $z : \mathcal{Q} \to \mathcal{Q}_s$ with $z(q) = q + x\mathbf{1}$, where x is the solution of $f(q + x\mathbf{1}) = r_*$. Note that $x = (r_* - f(q))/u$ because $f(q + x\mathbf{1}) = f(q) + xu$ by Assumption 3.3(ii).

Note that z is Lipschitz continuous because $z(q) = q + (r_* - f(q))\mathbf{1}/u$ and f is Lipschitz continuous by Assumption 3.3(i).

Finally, because $\mathcal{Q}$ is connected by S&F (see the discussion right after (4.7)) and the image of any continuous function on a connected set is connected, $z(\mathcal{Q})$ is connected. Note that every point in $z(\mathcal{Q})$ belongs to $\mathcal{Q}_s$ by definition. Every point in $\mathcal{Q}_s$ is also a point in $z(\mathcal{Q})$ (pick any $x \in \mathcal{Q}_s$, then $x \in \mathcal{Q}$ and that $z(x) = x$). Therefore $\mathcal{Q}_s = z(\mathcal{Q})$ is connected.

Non-convexity

We now show that $\mathcal{Q}_s$ is not necessarily convex by showing a counterexample.

Note that MDPs are special cases of SMDPs, we, therefore, show that $\mathcal{Q}_s$ is not convex for an MDP. Consider the MDP shown in the left subfigure of Figure 4.1. This is a communicating MDP. The optimal reward rate is 0.

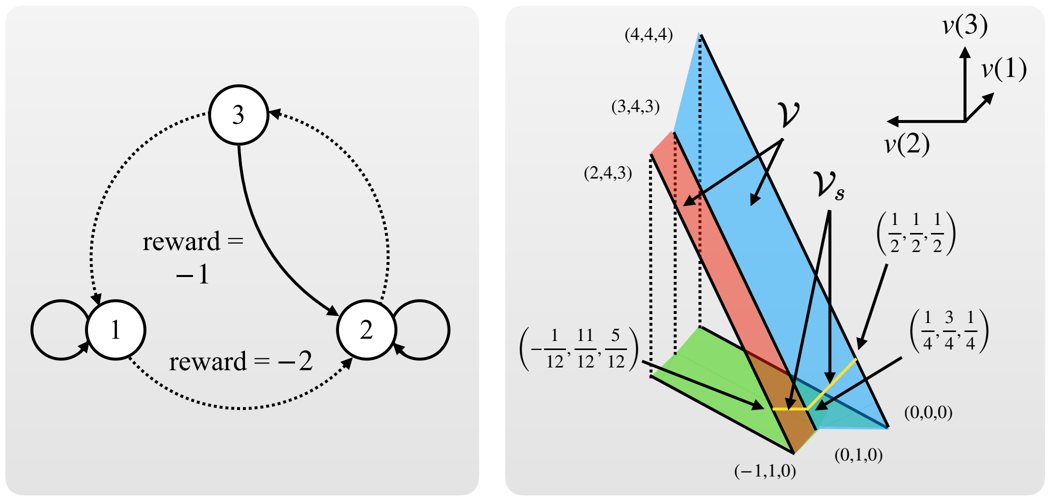

3
1
2
reward = −1
reward = −2
(4,4,4)
(3,4,3)
(2,4,3)
$v(3)$
$v(1)$
$v(2)$
$\mathcal{V}$
$\mathcal{V}_s$
$\left(\frac{1}{2},\frac{1}{2},\frac{1}{2}\right)$
$\left(\frac{1}{4},\frac{3}{4},\frac{1}{4}\right)$
$\left(-\frac{1}{12},\frac{11}{12},\frac{5}{12}\right)$
(0,0,0)
(0,1,0)
(−1,1,0)

Figure 4.1: Illustration example. *Left*: the example MDP. There are three states marked by three circles respectively. There are two actions `solid` and `dashed`, both have deterministic effects. Taking action `solid` at state *3* results in a reward of -1. Taking action `dashed` at state *1* results in a reward of -2. All other rewards are 0. *Right*: a graphical explanation of $\mathcal{V}, \mathcal{V}_s$. The two yellow line segments together represent $\mathcal{V}_s$. The red and blue regions together represent $\mathcal{V}$.

Let s and d denote `solid` and `dashed` respectively. Let $f(q) = \sum_{s,a} q(s,a)$. Such a choice of f satisfies the assumption on f (Assumption 3.3).

By (4.16), for any $q \in \mathcal{Q}_s$,

$$q(1,s) + q(1,d) + q(2,s) + q(2,d) + q(3,s) + q(3,d) = 0$$

and

$$\begin{aligned}
q(1,s) &= 0 - 0 + \max(q(1,s), q(1,d)) && (4.22)\\
q(1,d) &= -2 - 0 + \max(q(2,s), q(2,d))\\
q(2,s) &= 0 - 0 + \max(q(2,s), q(2,d))\\
q(2,d) &= 0 - 0 + \max(q(3,s), q(3,d))\\
q(3,s) &= -1 - 0 + \max(q(2,s), q(2,d))\\
q(3,d) &= 0 - 0 + \max(q(1,s), q(1,d)), && (4.23)
\end{aligned}$$

which implies

$$\begin{aligned}
q(1,s) &\geq q(1,d)\\
q(1,d) &= -2 + \max(q(2,s), q(2,d))\\
q(2,s) &\geq q(2,d)\\
q(2,d) &= \max(q(3,s), q(3,d))\\
q(3,s) &= -1 + \max(q(2,s), q(2,d))\\
q(3,d) &= \max(q(1,s), q(1,d))
\end{aligned}$$

Consider two solutions $q_1, q_2 \in \mathcal{Q}_s$ defined as follows:

$$\begin{aligned}
q_1(1,s) &= \frac{1}{2}, q_1(1,d) = -\frac{3}{2},\\
q_1(2,s) &= \frac{1}{2}, q_1(2,d) = \frac{1}{2},\\
q_1(3,s) &= -\frac{1}{2}, q_1(3,d) = \frac{1}{2},\\
q_2(1,s) &= -\frac{2}{3}, q_2(1,d) = -\frac{2}{3},\\
q_2(2,s) &= \frac{4}{3}, q_2(2,d) = \frac{1}{3},\\
q_2(3,s) &= \frac{1}{3}, q_2(3,d) = -\frac{2}{3}.
\end{aligned}$$

The midpoint of q_1 and q_2, $\bar{q} \doteq 0.5q_1 + 0.5q_2$, satisfies

$$\begin{aligned}
&\bar{q}(1,s) = -\frac{1}{12}, \bar{q}(1,d) = -\frac{13}{12}, \\
&\bar{q}(2,s) = \frac{11}{12}, \bar{q}(2,d) = \frac{5}{12}, \\
&\bar{q}(3,s) = -\frac{1}{12}, \bar{q}(3,d) = -\frac{1}{12}.
\end{aligned}$$

Note that

$$\frac{5}{12} = \bar{q}(2,d) \neq \max(\bar{q}(3,s), \bar{q}(3,d)) = -\frac{1}{12}.$$

Therefore $\bar{q}$ does not satisfy the action-value optimality equation (Equation 3.7) and $\bar{q} \notin \mathcal{Q}_s$. Thus $\mathcal{Q}_s$ is not convex. The proof is finished.

Finally, I would like to illustrate graphically the set $\mathcal{Q}$ and $\mathcal{Q}_s$ in some way for the readers to better understand these two sets. Unfortunately, even in a small MDP with just two states and two actions, the dimension of each element in $\mathcal{Q}$ and $\mathcal{Q}_s$ is four, which is more than what I can illustrate graphically. Therefore, I choose to show the maximum action value for each state in the MDP drawn in the left subfigure of Figure 4.1. Because there are only three states in the MDP, I can show the maximum action value for all states graphically. Specifically, I will illustrate the set $\mathcal{V} \doteq \{v \in \mathbb{R}^{\mathcal{S}} : v(s) = \max_a q(s,a)\ \forall s \in \mathcal{S} \text{ for some } q \in \mathcal{Q}\}$, and $\mathcal{V}_s \doteq \{v \in \mathbb{R}^{\mathcal{S}} : v(s) = \max_a q(s,a)\ \forall s \in \mathcal{S} \text{ for some } q \in \mathcal{Q}_s\}$.

One should expect to see that both $\mathcal{V}$ and $\mathcal{V}_s$ are non-empty, connected, and closed. In addition, $\mathcal{V}$ is unbounded while $\mathcal{V}_s$ is bounded. To see these properties, note that by our construction, $\mathcal{V}$ is the solution set of the state-value optimality equation (4.8). Therefore by S&F's result, $\mathcal{V}$ is non-empty, connected, closed, and bounded. On the other hand, because $\mathcal{Q}_s$ is non-empty, connected, closed, bounded by the theorem just proved, $\mathcal{V}_s$ is also non-empty, connected, closed, bounded. To see this point, note that $h : \mathbb{R}^{|\mathcal{S}| \times \mathcal{A}} \to \mathbb{R}^{|\mathcal{S}|}$ with $h(q)(s) \doteq \max_a q(s,a)$ is Lipschitz continuous, and the image of a bounded/connected set through a Lipschitz continuous function is bounded/connected. Furthermore, for any $q \in \mathcal{Q}_s$ and its associated v, $q(s,a) = h'(v)(s,a) \doteq \sum_{s'} p(s' \mid s,a) v(s')$ by the definition of v and $\mathcal{Q}_s$. Using the continuity of h' and the fact that

$\mathcal{Q}_s$ closed, we have the pre-image of $\mathcal{Q}_s$, $\mathcal{V}_s$, is also closed. Verifying the non-emptiness is trivial.

Now consider the example MDP, for any $q \in \mathcal{Q}$, using $v(s) \doteq \max_a q(s,a)\ \forall s \in \mathcal{S}$ and $\hat{r}_* = 0$, we have $q(s,a) = r(s,a) + \sum_{s'} p(s' \mid s,a) v(s')$ for all s. Using this equation and (4.22)—(4.23), we have, for any $q \in \mathcal{Q}$, its associated v satisfies

$$\begin{aligned}
v(1) &= \max(v(1), -2 + v(2)),\\
v(2) &= \max(v(2), v(3)),\\
v(3) &= \max(v(1), -1 + v(2)),
\end{aligned}$$

which implies,

$$\begin{aligned}
v(1) &\geq -2 + v(2),\\
v(2) &\geq v(3),\\
v(3) &= \max(v(1), -1 + v(2)).
\end{aligned}$$

Therefore,

$$\mathcal{V} = \{v \in \mathbb{R}^3 \mid v(1) \geq -2 + v(2); v(2) \geq v(3); v(3) = \max(v(1), -1 + v(2))\}.$$

Similarly,

$$\mathcal{V}_s = \{v \in \mathcal{V} \mid 2v(1) + 3v(2) + v(3) = 3\}.$$

The right subfigure of Figure 4.1 shows $\mathcal{V}$ and $\mathcal{V}_s$. It can be seen that $\mathcal{V}$ corresponds to the union of two connected rectangles, each of which has an infinite length and a finite width. It is clear that, in this example, $\mathcal{V}$ is non-empty, connected, closed, unbounded, and non-convex. Therefore this example verifies S&F's characterization of $\mathcal{V}$.

$\mathcal{V}_s$ corresponds to the two connected yellow line segments in the figure. It is clear that, in this example, $\mathcal{V}_s$ is non-empty, connected, closed, and bounded. This observation is consistent with my above discussion about the properties of $\mathcal{V}_s$.

Result 3: Convergence to $\mathcal{Q}_s$ implies convergence of greedy policies

Lemma 4.3. *Assume that an SMDP is weakly communicating and Assumption 3.3 holds. Suppose a sequence of random vectors $Q_n \in \mathbb{R}^d$ converges to $\mathcal{Q}_s$ almost surely, let π_n be a greedy policy w.r.t. Q_n (i.e., $\pi_n(s) \in \operatorname{argmax} Q_n(s, \cdot)$), then, almost surely, $\pi_n \in \Pi_*^D$ for sufficiently large n, where Π_*^D is the set of deterministic optimal policies.*

Proof. Consider a member $\bar{q}$ of $\mathcal{Q}_s$, any greedy policy w.r.t. the member is optimal. Let $\mathcal{A}_s^* \doteq \{a : \bar{q}(s,a) = \max_{a'} \bar{q}(s,a')\}$. Choose $\epsilon(\bar{q}) = \min_{s, a \notin \mathcal{A}_s^*} \max_{a'} \bar{q}(s,a') - \bar{q}(s,a)$. Denote the $\epsilon(\bar{q})$-neighborhood of $\bar{q}$, $G_{\bar{q}} \doteq \{q \in \mathbb{R}^{\mathcal{S}\times\mathcal{A}} : \|q - \bar{q}\|_\infty < \epsilon(\bar{q})\}$. Let $G \doteq \cup_{q \in \mathcal{Q}_s} G_q$, then every member of G has its greedy policies optimal and $\mathcal{Q}_s \subseteq G$ by construction. Further, G is an open set because it is a union of open sets.

Because G is an open set, its complement G^c is closed. Note that the distance between a closed set and a compact set is positive if the two sets are disjoint. Therefore the distance between G^c and $\mathcal{Q}_s$ is positive by Theorem 4.1 and $\mathcal{Q}_s \subseteq G$. Denote this distance as $\epsilon > 0$. Because Q_n converges a.s. to $\mathcal{Q}_s$, there exists a sample-path dependent n_0 such that for all $n \geq n_0$, the distance between Q_n and $\mathcal{Q}_s$ is less than ϵ, which means that $Q_n \in G$ for all $n \geq n_0$. Because every member of G has its greedy policies optimal, greedy policies of Q_n are optimal for all $n \geq n_0$.

□

4.4 Problem Setup

This section presents the problem setup for this chapter. Specifically, this section presents the definition of options and defines learning and planning problems with a set of options.

This chapter formalizes an agent's interaction with its world by a finite Markov decision process (MDP) $\mathcal{M} = (\mathcal{S}, \mathcal{A}, \mathcal{R}, d_0, p)$ and a finite set of options $\mathcal{O}$. Each option o in $\mathcal{O}$ has two components: the option's *policy* $\pi^o : \mathcal{S} \to \Delta(\mathcal{A})$, and a probability distribution of the option's *termination* $\beta^o \in \mathcal{S} \to \Delta(\{0,1\})$. Here 1 denotes termination and 0 denotes continuing. Let's use $\pi(a \mid s, o)$ to denote $\pi^o(a, s)$ and $\beta(s, o)$ to denote $\beta^o(s)$. Sutton et al.'s (1999) options additionally have an *initiation* set that consists of the states at which the option can be initiated. To simplify the presentation, we allow all options to be initiated in all states of the state space; the algorithms and theoretical results can easily be extended to incorporate initiation from specific states. Finally, we call a policy that produces the probability of choosing each option at each state a *hierarchical policy*.

If an option o is initiated at time t at state S_t, then the action A_t is chosen according to the option's policy $\pi(\cdot \mid S_t, o)$. The agent then observes the next state S_{t+1} and reward R_{t+1} according to p. The option terminates at S_{t+1} with probability $\beta(S_{t+1}, o)$ or continues with action A_{t+1} chosen according to $\pi(\cdot \mid S_{t+1}, o)$. It then possibly terminates in S_{t+2} according to $\beta(S_{t+2}, o)$, and so on. At an option termination, one way to govern an agent's behavior is to choose a new option according to a behavior hierarchical policy b. In this case, when an option terminates at time t, the next option is selected stochastically according to $b(\cdot \mid S_t)$. The option initiates at S_t and terminates at S_{t+K}, where K is a random variable denoting the number of time steps the option executed. At S_{t+K}, a new option is again chosen according to $b(\cdot \mid S_{t+K})$, and so on. We use the notation O_t to denote whatever option is being executed at time step t. Note that O_t will remain the same for as many steps as the option executes. Also, note that actions are special options: every action a is an option o that terminates after exactly one step ($\beta(s, o) = 1, \ \forall\ s$) and has its policy picking a in every state ($\pi(a \mid s, o) = 1, \ \forall\ s$).

Let T_n denote the time step when the $n-1^{\text{th}}$ option terminates and the n^{th} option is chosen. Denote the n^{th} option by $\hat{O}_n \doteq O_{T_n}$, its starting state by $\hat{S}_n \doteq S_{T_n}$, the cumulative reward during its execution by $\hat{R}_{n+1} \doteq \sum_{t=T_n+1}^{T_{n+1}} R_t$, the state it terminates in by $\hat{S}_{n+1} \doteq S_{T_{n+1}}$, and its length by $\hat{L}_{n+1} \doteq T_{n+1} - T_n$.

Let $\mathcal{L}$ be the set of all possible lengths of options and $\hat{\mathcal{R}}$ be the set of all possible cumulative rewards. Note that $\mathcal{R}$ and $\hat{\mathcal{R}}$ are not necessarily the same—while $\mathcal{R}$ can be finite, $\hat{\mathcal{R}}$ is typically countably infinite because $\mathcal{L}$ is typically countably infinite. Let $\hat{p}(s', r, l \mid s, o)$ be, when executing option o starting from state s, the probability of terminating at state s', with cumulative reward r and length l. Formally, for any $s, s' \in \mathcal{S}, o \in \mathcal{O}, r \in \hat{\mathcal{R}}, l \in \mathcal{L}$, $\hat{p}$ can be defined recursively in the following way. For each state $s \in \mathcal{S}$ and each option $o \in \mathcal{O}$,

$$\hat{p}(s', r, l \mid s, o) \doteq \sum_a \pi(a \mid s, o) \sum_{\tilde{s}, \tilde{r}} p(\tilde{s}, \tilde{r} \mid s, a)$$
$$[\beta(\tilde{s}, o)\mathbf{I}(\tilde{s} = s', \tilde{r} = r, \tilde{l} = 1) + (1 - \beta(\tilde{s}, o))\hat{p}(s', r - \tilde{r}, l - 1 \mid \tilde{s}, o)], \quad (4.24)$$

where $\mathbf{I}$ is the indicator function.

It is natural to assume that every option terminates eventually regardless of the start state. This is formally stated in the next assumption.

Assumption 4.1. For each option $o \in \mathcal{O}$, when executing the option, there is a non-zero probability of terminating the option after at most $|\mathcal{S}|$ stages, regardless of the initial states.

Throughout this chapter, the above assumption is assumed to hold. If the above assumption is not satisfied, there exists an option that does not terminate after $|\mathcal{S}|$ stages, starting from a state. This can only happen if the option does not terminate starting from that state, because there are only $|\mathcal{S}|$ states and there is a positive probability to reach all states within $|\mathcal{S}|$ stages if they are reachable from the start state.

Proposition 4.2. *Under Assumption 4.1, the expectations of the execution time and cumulative reward of every option at every state exist and are finite. So do the variances.*

Proof. Note that the execution time of each option $o \in \mathcal{O}$ is the return of executing this option's policy in a stochastic shortest path MDP (SSP-MDP, Bertsekas 2007) with state space $\mathcal{S} \cup \{\perp\}$ ($\perp$ is the "terminal" state), action space $\mathcal{A}$, reward space $\{0, 1\}$, and transition function $\hat{p}$ satisfying:

$$\hat{p}(s', 1 \mid s, a) \doteq (1 - \beta(s', o)) \sum_r p(s', r \mid s, a), \quad \forall\ s, s' \neq \perp, a \in \mathcal{A},$$
$$\hat{p}(\perp, 1 \mid s, a) \doteq \sum_{s'} \beta(s', o) \sum_r p(s', r \mid s, a), \quad \forall\ s \neq \perp, a \in \mathcal{A},$$
$$\hat{p}(\perp, 0 \mid \perp, a) \doteq 1, \quad \forall\ a \in \mathcal{A}.$$

Also, note that by Assumption 4.1, the option's policy is a 'proper' policy (Bertsekas 2007) in the SSP-MDP. That is, when using the policy, the SSP-MDP reaches the terminal state eventually regardless of the start state. Because the expected value of every proper policy of an SSP-MDP exists and is finite (Section 2.1 of Bertsekas (2007)), the expected value of the execution time of option o exists.

The variance always exists for any random variable. The variance is finite because there is a non-zero probability of terminating the option after at most $|S|$ stages (Assumption 4.1).

Similarly, the cumulative reward of each option o is the return of executing o's policy in a SSP-MDP with state space $\mathcal{S} \cup \{\perp\}$, action space $\mathcal{A}$, reward space $\mathcal{R} \cup \{0\}$ and transition function $\hat{p}$ satisfying:

$$\begin{aligned}
&\hat{p}(s', r \mid s, a) \doteq (1 - \beta(s', o))p(s', r \mid s, a), \quad \forall\ s, s' \neq \perp, a \in \mathcal{A} \\
&\hat{p}(\perp, r \mid s, a) \doteq \sum_{s'} \beta(s', o)p(s', r \mid s, a), \quad \forall\ s \neq \perp, a \in \mathcal{A} \\
&\hat{p}(\perp, 0 \mid \perp, a) \doteq 1, \quad \forall\ a \in \mathcal{A}.
\end{aligned}$$

Again the option's policy is proper and the expected value of the cumulative reward of option o exists. The variance is finite because the reward space $\mathcal{R}$ is finite and there is a non-zero probability of terminating the option after at most $|S|$ stages (Assumption 4.1). □

An MDP $(\mathcal{S}, \mathcal{A}, \mathcal{R}, d_0, p)$ and a set of options $\mathcal{O}$ together result in an SMDP $(\mathcal{S}, \mathcal{O}, \hat{\mathcal{R}}, \mathcal{L}, d_0, \hat{p})$ given that Assumption 4.1 holds. Let $\hat{\Pi}$ denote the set of stationary hierarchical policies. Given a hierarchical policy $\mu \in \hat{\Pi}$, the reward rate achieved by the hierarchical policy in the MDP, starting from state s, denoted as $\hat{r}(\mu, s)$, is exactly μ's reward rate in the resulting SMDP, starting from state s. The best possible reward rate achieved by a hierarchical policy, denoted as $\hat{r}_*$ is the optimal reward rate in the resulting SMDP (assume that the SMDP is weakly communicating). Given a hierarchical policy $\mu \in \hat{\Pi}$, we call

$$q_\mu(s, o) \doteq \lim_{n \to \infty} \frac{1}{n} \sum_{k=1}^{n} \sum_{t=1}^{k} \mathbb{E}_\mu[R_t - \hat{r}(\mu) \mid S_0 = s, O_0 = o, O_{1:t-1} \sim \mu].$$

the differential option-value function of μ and denote it as $\hat{q}_\mu$. Finally, the SMDP's action-value evaluation equation is

$$q(s,o) = r(s,o) - \bar{r} \cdot l(s,o) + \sum_{s'} \hat{p}(s' \mid s,o) \sum_{o'} \mu(o' \mid s') q(s',o'), \tag{4.25}$$

where

$$\hat{r}(s,o) \doteq \sum_{s',r,l} \hat{p}(s',r,l \mid s,o) r, \tag{4.26}$$

$$\hat{l}(s,o) \doteq \sum_{s',r,l} \hat{p}(s',r,l \mid s,o) l, \tag{4.27}$$

$$\hat{p}(s' \mid s,o) \doteq \sum_{r,l} \hat{p}(s',r,l \mid s,o) \tag{4.28}$$

are the expected reward and length and state dynamics for each transition. The SMDP's action-value optimality equation is

$$q(s,o) = r(s,o) - \bar{r} \cdot l(s,o) + \sum_{s'} \hat{p}(s' \mid s,o) \max_{o'} q(s',o'). \tag{4.29}$$

Given an MDP, a set of options, and a unichain hierarchical policy $\mu \in \hat{\Pi}$, the goal of prediction problems is to identify $\hat{r}(\mu)$ and $\hat{q}_\mu$. Given a weakly communicating MDP and a set of options that result in a weakly communicating SMDP, the goal of control problems is to find a policy that achieves $\hat{r}_*$. Denote the set of deterministic optimal policies as $\hat{\Pi}_*^D$. For both prediction and control problems, this chapter considers off-policy learning problems and planning problems. In off-policy learning problems, algorithms may follow a hierarchical policy that can be different from the agent's learned hierarchical policy. Planning problems in this chapter include those that use standard action models and more jumpy option models. More detail of these models will be provided when needed.

4.5 Inter-Option (SMDP) Algorithms

This section introduces inter-option prediction and control algorithms and the associated convergence theories.

Inter-option algorithms operate on a sequence of option transitions. At the option-transition level, interacting with an MDP by choosing from a set of options and executing them is equivalent to interacting with the SMDP resulting from the MDP and the set of options. Therefore inter-option algorithms are also SMDP algorithms. In this sense, the algorithms introduced in this section extend the differential learning and planning algorithms from MDPs to SMDPs.

I begin with the control learning algorithm and then move on to the planning and prediction algorithms.

Learning algorithms operate on a stream of experience. Inter-option algorithms operate on a stream of option transitions: $\ldots, \hat{S}_n, \hat{O}_n, \hat{R}_{n+1}, \hat{L}_{n+1}, \hat{S}_{n+1}, \ldots$ generated by following some hierarchical behavior policy that chooses options.

Recall Differential Q-learning introduced in Section 3.5:

$$\begin{aligned} Q_{t+1}(S_t, A_t) &\doteq Q_t(S_t, A_t) + \alpha_{\nu_t(S_t,A_t)}\delta_t, \\ Q_{t+1}(s, a) &\doteq Q_t(s, a), \forall (s, a) \neq (S_t, A_t), \\ \bar{R}_{t+1} &\doteq \bar{R}_t + \eta\alpha_{\nu_t(S_t,A_t)}\delta_t, \end{aligned}$$

where Q_t is a vector of size $|\mathcal{S} \times \mathcal{A}|$ that approximates a solution of q in the action-value optimality equation for MDPs, $\bar{R}_t$ is a scalar estimate of the optimal reward rate, $\{\alpha_n\}_{n\geq 1}$ is a step–size sequence, $\nu_t(S_t, A_t)$ is the number of times the (S_t, A_t) pair has been updated up to and include time step t, η is a positive constant, and δ_t is the temporal-difference (TD) error: $\delta_t \doteq R_t - \bar{R}_t + \max_a Q_t(S_{t+1}, a) - Q_t(S_t, A_t)$. The most straightforward inter-option extension of Differential Q-learning is:

$$\begin{aligned} Q_{n+1}(\hat{S}_n, \hat{O}_n) &\doteq Q_n(\hat{S}_n, \hat{O}_n) + \alpha_{\nu_n(\hat{S}_n,\hat{O}_n)}\delta_n, && (4.30) \\ Q_{n+1}(s, o) &\doteq Q_n(s, o), \forall (s, o) \neq (\hat{S}_n, \hat{O}_n), \\ \bar{R}_{n+1} &\doteq \bar{R}_n + \eta\alpha_{\nu_n(\hat{S}_n,\hat{O}_n)}\delta_n, && (4.31) \end{aligned}$$

where Q_n is a vector of size $|\mathcal{S} \times \mathcal{O}|$ that approximates a solution of q in (4.29), $\bar{R}_n$ is a scalar estimate of $\hat{r}_*$, $\nu_n(\hat{S}_n, \hat{O}_n)$ is the number of times the $(\hat{S}_n, \hat{O}_n)$ pair has been updated up to and include step n, and δ_n is the TD error:

$$\delta_n \doteq \hat{R}_n - \hat{L}_n\bar{R}_n + \max_o Q_n(\hat{S}_{n+1}, o) - Q_n(\hat{S}_n, \hat{O}_n). \tag{4.32}$$

Such an algorithm is prone to instability because the *sampled* option length $\hat{L}_n$ can be quite large, and any error in the reward-rate estimate $\bar{R}_n$ gets multiplied by the potentially-large option length. Using small step sizes might make the updates relatively stable, but at the cost of slowing down learning for options of shorter lengths. This could make the choice of step sizes quite critical, especially when the range of the options' lengths is large and unknown. Alternatively, inspired by Schweitzer (1971), I propose scaling the updates by the *estimated* length of the option being executed:

$$Q_{n+1}(\hat{S}_n, \hat{O}_n) \doteq Q_n(\hat{S}_n, \hat{O}_n) + \alpha_{\nu_n(\hat{S}_n,\hat{O}_n)}\delta_n / L_n(\hat{S}_n, \hat{O}_n), \tag{4.33}$$

$$Q_{n+1}(s, o) \doteq Q_n(s, o), \ \forall (s, o) \neq (\hat{S}_n, \hat{O}_n),$$

$$\bar{R}_{n+1} \doteq \bar{R}_n + \eta\alpha_{\nu_n(\hat{S}_n,\hat{O}_n)}\delta_n / L_n(\hat{S}_n, \hat{O}_n), \tag{4.34}$$

where $L_n(\cdot, \cdot)$ comes from an additional vector of estimates $L : \mathcal{S} \times \mathcal{O} \to \mathbb{R}$ that approximates the expected lengths of state-option pairs, updated from experience by:

$$L_{n+1}(\hat{S}_n, \hat{O}_n) \doteq L_n(\hat{S}_n, \hat{O}_n) + \beta_{\nu_n(\hat{S}_n,\hat{O}_n)}(\hat{L}_n - L_n(\hat{S}_n, \hat{O}_n)),$$

$$L_{n+1}(s, o) \doteq L_n(s, o), \ \forall (s, o) \neq (\hat{S}_n, \hat{O}_n), \tag{4.35}$$

where β_n is an another step–size sequence. The TD-error δ_n in (4.33) and (4.34) is

$$\delta_n \doteq \hat{R}_n - L_n(\hat{S}_n, \hat{O}_n)\bar{R}_n + \max_o Q_n(\hat{S}_{n+1}, o) - Q_n(\hat{S}_n, \hat{O}_n), \tag{4.36}$$

which is different from (4.32) with the estimated expected option length $L_n(\hat{S}_n, \hat{O}_n)$ being used instead of the sampled option length $\hat{L}_n$. Equations 4.33–4.36 make up the *inter-option Differential Q-learning* algorithm. The pseudo-code of this algorithm is provided in Algorithm 7.

Remark: In the algorithm, $L_n(s, o)$ is used to estimate the expected option length $\hat{l}(s, o)$. The way I choose to update L_n is just by doing a simple sample average. One could, of course, use any method, as long as $L_n(s, o)$ converges a.s. to the expected option length $\hat{l}(s, o)$ for each s, o.

Remark: Q_0 and R_0 can be initialized arbitrarily but L_0 can not be initialized to $\mathbf{0}$ because it is the divisor for both (4.33) and (4.34) for the first update. I will assume that $L_0 > \mathbf{0}$. In this way, no elements in L_n will reach a value of zero for all $n \geq 0$ because an option's length is at least one ($\hat{L}_n \geq 1$).

Remark: The scaling factor in the algorithm needs to be the expected option length $L_n(\hat{S}_n, \hat{O}_n)$ and not the sampled option length $\hat{L}_n$. If we use the sampled option length $\hat{L}_n$ as the scaling factor, the update rules of the algorithm become

$$Q_{n+1}(\hat{S}_n, \hat{O}_n) \doteq Q_n(\hat{S}_n, \hat{O}_n) + \alpha_{\nu_n(\hat{S}_n, \hat{O}_n)} \delta_n / \hat{L}_n, \tag{4.37}$$

$$\bar{R}_{n+1} \doteq \bar{R}_n + \eta \alpha_{\nu_n(\hat{S}_n, \hat{O}_n)} \delta_n / \hat{L}_n. \tag{4.38}$$

The above two updates can not guarantee convergence to the desired values because $\mathbb{E}[\delta_n / \hat{L}_n] = 0$ does not imply that the Bellman equation $\mathbb{E}[\delta_n] = 0$ is satisfied.

Remark: There is yet another way of extending Differential Q-learning to the inter-option algorithm that appears to work properly at first glance but does not actually. This way uses, for each option, the average-reward rate per step instead of the total reward as the reward of the option. In particular, such an extension use update rules (4.30) and (4.31), but with TD error defined as:

$$\delta'_n \doteq \hat{R}_n / \hat{L}_n - \bar{R}_n + \max_o Q_n(\hat{S}_{n+1}, o) - Q_n(\hat{S}_n, \hat{O}_n)$$

Unfortunately, such an extension can not guarantee convergence to a desired point. Specifically, the extension, if converges, converges to a solution of $\mathbb{E}[\delta'_n] = 0$, which is, again, not necessarily a solution of the Bellman equation $\mathbb{E}[\delta_n] = 0$ (4.29).

We are now ready to present the convergence result of inter-option Differential Q-learning. This result shows that the option-length estimates L_n in inter-option Differential Q-learning converge to the expected lengths $\hat{l}$. It also shows that the reward-rate estimate $\bar{R}_n$ converges to the optimal reward rate estimate $\hat{r}_*$, and that the option-value estimates Q_n converges to the set of solutions of (4.29) and

$$\hat{r}_* = \eta \sum q - \eta \sum Q_0 + \bar{R}_0. \tag{4.39}$$

Denote this set as $\hat{\mathcal{Q}}_\infty$. By Theorem 4.1, $\hat{\mathcal{Q}}_\infty$ is non-empty, closed, bounded, connected and possibly non-convex.

Algorithm 7: Inter-option Differential Q-learning

```
Input: Behavioral policy $b$
Algorithm parameters: step-size sequences $\alpha_n, \beta_n$, parameter $\eta$
1  Initialize $Q(s,o)\ \forall\, s \in \mathcal{S}, o \in \mathcal{O}, \bar{R}$ arbitrarily (e.g., to zero)
2  Initialize $L(s,o)\ \forall\, s \in \mathcal{S}, o \in \mathcal{O}$ to be any value greater than 0
3  $\nu(s,o) \leftarrow 0\ \forall s, o$
4  Obtain initial $S$
5  while still time to train do
6      Initialize $\hat{L} \leftarrow 0, \hat{R} \leftarrow 0, S_{tmp} \leftarrow S$
7      $O \leftarrow$ option sampled from $b(\cdot \mid S)$
8      $\nu(S,O) \leftarrow \nu(S,O) + 1$
9      do
10         Sample primitive action $A \sim \pi(\cdot \mid S, O)$
11         Take action $A$, observe $R, S'$
12         $\hat{L} \leftarrow \hat{L} + 1$
13         $\hat{R} \leftarrow \hat{R} + R$
14         $S \leftarrow S'$
15     while $O$ doesn't terminate in $S'$
16     $S \leftarrow S_{tmp}$
17     $L(S,O) \leftarrow L(S,O) + \beta_{\nu(S,O)}\big(\hat{L} - L(S,O)\big)$
18     $\delta \leftarrow \hat{R} - \bar{R} \cdot L(S,O) + \max_o Q(S',o) - Q(S,O)$
19     $Q(S,O) \leftarrow Q(S,O) + \alpha_{\nu(S,O)}\delta / L(S,O)$
20     $\bar{R} \leftarrow \bar{R} + \eta\alpha_{\nu(S,O)}\delta / L(S,O)$
21     $S \leftarrow S'$
22 end
23 return $Q$
```

Theorem 4.2. *Given an MDP and a set of options, if the resulting SMDP is weakly communicating, Assumption 3.5 (i) holds for both $\{\alpha_n\}$ and $\{\beta_n\}$, Assumption 3.5 (ii) holds for $\{\alpha_n\}$, Assumption 3.6 (with $\mathcal{I} = \mathcal{S} \times \mathcal{O}$) holds for $\{\alpha_n\}$, and $L_0(s,o) > 0$ for all $s \in \mathcal{S}$, $o \in \mathcal{O}$, inter-option Differential Q-learning* (4.33)–(4.36) *converges almost surely, 1) L_n to $\hat{l}$, 2) $\bar{R}_n$ to $\hat{r}_*$, 3) Q_n to a sample-path dependent compact connected subset of $\hat{\mathcal{Q}}_\infty$, and 4) almost surely, $\mu_n \in \Pi_*^D$ for sufficiently large n, where μ_n is any greedy policy w.r.t. Q_n.*

The proof of this theorem will be presented at the end of this section.

If inter-option Differential Q-learning is applied to a simulated experience generated from an option model of the world, then it becomes a planning algorithm, which we call *inter-option Differential Q-planning.* Formally, the option model is a function $\tilde{p} : \mathcal{S} \times \mathcal{O} \to \Delta(\mathcal{S} \times \hat{\mathcal{R}} \times \mathcal{L})$, analogous to $\hat{p}$, that, like $\hat{p}$, sums to 1: $\sum_{s',r,l} \tilde{p}(s', r, l \mid s, o) = 1$ for all s, o. A model SMDP can be thus constructed using $\tilde{p}$ and $\mathcal{S}, \mathcal{O}, \hat{\mathcal{R}}, \mathcal{L}, d_0$. Given this model SMDP, we can define $\tilde{l}$, $\tilde{\mathcal{Q}}_\infty$, $\tilde{r}_*$, and $\tilde{\Pi}^D_*$ in the same way as $\hat{l}$, $\hat{\mathcal{Q}}_\infty$, $\hat{r}_*$, $\hat{\Pi}^D_*$ but for the model SMDP.

The simulated transitions are generated as follows: at each planning step n, the agent arbitrarily chooses a state $\tilde{S}_n$ and an option $\tilde{O}_n$, and applies $\tilde{p}$ to generate a simulated resulting state and reward $\tilde{S}'_n, \tilde{R}_n, \tilde{L}_n \sim \tilde{p}(\cdot, \cdot, \cdot \mid \tilde{S}_n, \tilde{O}_n)$.

Like inter-option Differential Q-learning, inter-option Differential Q-planning maintains a table of value estimates $Q_n : \mathcal{S} \times \mathcal{O} \to \mathbb{R}$, a table of option-length estimates $L_n : \mathcal{S} \times \mathcal{O} \to \mathbb{R}$, and a reward-rate estimate $\bar{R}_n$. At each planning step n, these estimates are updated by (4.33)–(4.36), just as in inter-option Differential Q-learning, except now using $\tilde{S}_n, \tilde{O}_n, \tilde{R}_n, \tilde{L}_n, \tilde{S}'_n$ instead of $\hat{S}_n, \hat{O}_n, \hat{R}_{n+1}, \hat{L}_n, \hat{S}_{n+1}$.

Theorem 4.3. *Under the same assumptions as in Theorem 4.2 (except now for the model SMDP corresponding to $\tilde{p}$ rather than $\hat{p}$), inter-option Differential Q-planning converges almost surely, 1) L_n to $\tilde{l}$, 2) $\bar{R}_n$ to $\tilde{r}_*$, 3) Q_n to a sample-path dependent compact connected subset of $\tilde{\mathcal{Q}}_\infty$, and 4) almost surely, $\mu_n \in \tilde{\Pi}^D_*$ for a sufficiently large n, where μ_n is any greedy policy w.r.t. Q_n.*

Remark: If the model SMDP produces the expected option length instead of the sample option length, we may directly use the option length produced by the model as L_n, the estimated option lengths, and remove the update rule that estimates L_n (Equation 4.35) from the update rules of inter-option Differential Q-planning, therefore simplifying this algorithm.

Our prediction learning algorithm, called *inter-option Differential Q-evaluation-learning*, also has update rules (4.33–4.35) but with the TD error:

$$\delta_n \doteq \hat{R}_n - L_n(\hat{S}_n, \hat{O}_n)\bar{R}_n + \sum_o \mu(o \mid \hat{S}_{n+1}) Q_n(\hat{S}_{n+1}, o) - Q_n(\hat{S}_n, \hat{O}_n). \quad (4.40)$$

The pseudo-code of this algorithm is provided in Algorithm 8.

```
Algorithm 8: Inter-option Differential Q-evaluation-learning
  Input: Behavioral policy $b$, target policy $\mu$
  Algorithm parameters: step-size sequences $\alpha_n, \beta_n$, parameter $\eta$
 1 Initialize $Q(s, o)\ \forall\, s \in \mathcal{S}, o \in \mathcal{O}, \bar{R}$ arbitrarily (e.g., to zero)
 2 Initialize $L(s, o)\ \forall\, s \in \mathcal{S}, o \in \mathcal{O}$ to be any value greater than 0
 3 $\nu(s, o) \leftarrow 0\ \forall s, o$
 4 Obtain initial $S$
 5 while still time to train do
 6     Initialize $\hat{L} \leftarrow 0, \hat{R} \leftarrow 0, S_{tmp} \leftarrow S$
 7     $O \leftarrow$ option sampled from $b(\cdot \mid S)$
 8     $\nu(S, O) \leftarrow \nu(S, O) + 1$
 9     do
10         Sample primitive action $A \sim \pi(\cdot \mid S, O)$
11         Take action $A$, observe $R, S'$
12         $\hat{L} \leftarrow \hat{L} + 1$
13         $\hat{R} \leftarrow \hat{R} + R$
14         $S \leftarrow S'$
15     while $O$ doesn't terminate in $S'$
16     $S \leftarrow S_{tmp}$
17     $L(S, O) \leftarrow L(S, O) + \beta_{\nu(S,O)}\big(\hat{L} - L(S, O)\big)$
18     $\delta \leftarrow \hat{R} - \bar{R} \cdot L(S, O) + \sum_o \mu(o \mid S')Q(S', o) - Q(S, O)$
19     $Q(S, O) \leftarrow Q(S, O) + \alpha_{\nu(S,O)}\delta / L(S, O)$
20     $\bar{R} \leftarrow \bar{R} + \eta\alpha_{\nu(S,O)}\delta / L(S, O)$
21     $S \leftarrow S'$
22 end
23 return $Q$
```

If the target policy is unichain in the SMDP, the solution of (4.25) and

$$\hat{r}(\mu) - \bar{R}_0 = \eta\left(\sum q - \sum Q_0\right) \tag{4.41}$$

is unique. Denote this solution as $\hat{q}_\infty$. The following theorem shows that our prediction learning algorithm converges to $\hat{q}_\infty$.

Theorem 4.4. *Given an MDP and a set of options, if the hierarchical policy μ is unichain in the resulting SMDP, Assumption 3.5(i) holds for both $\{\alpha_n\}$ and $\{\beta_n\}$, Assumption 3.5(ii) holds for $\{\alpha_n\}$, Assumption 3.6 (with $\mathcal{I} = \mathcal{S} \times \mathcal{O}$) holds for $\{\alpha_n\}$, and $L_0(s, o) > 0$ for all $s \in \mathcal{S}, o \in \mathcal{O}$, inter-option Differential Q-evaluation-learning (4.33–4.35, 4.40) converges almost surely, 1) L_n to $\hat{l}$, 2) $\bar{R}_n$ to $\hat{r}(\mu)$, and 3) $Q_n(s, o)$ to $\hat{q}_\infty$.*

Just like inter-option Differential Q-planning, the planning version of inter-option Q-evaluation-learning uses simulated transitions. A similar convergence result for inter-option Q-evaluation-planning can be obtained as follows. Define $\tilde{q}_\infty$ and $\tilde{r}(\mu)$ the same way as the $\hat{q}_\infty$ and $\hat{r}(\mu)$ but for the model SMDP instead of SMDP resulting from the real MDP and the given set of options.

Theorem 4.5. *Under the same assumptions as in Theorem 4.4 (except now for the model SMDP corresponding to $\tilde{p}$ rather than $\hat{p}$), inter-option Differential Q-evaluation-planning converges almost surely, 1) L_n to $\tilde{l}$, 2) $\bar{R}_n$ to $\tilde{r}(\mu)$, and 3) $Q_n(s, o)$ to $\tilde{q}_\infty$.*

The rest of this section shows proof for Theorem 4.2. The proofs of Theorems 4.3, 4.4, 4.5 are very similar and are therefore omitted.

Following a similar transformation of Differential Q-learning to a single update rule, we can transform (4.33)–(4.36) to the following update rule:

$$\begin{aligned}
&Q_{n+1}(\hat{S}_n, \hat{O}_n) = Q_n(\hat{S}_n, \hat{O}_n) \\
&\quad + \alpha_{\nu_n(\hat{S}_n, \hat{O}_n)} \left(\frac{\hat{R}_{n+1} - L_n(\hat{S}_n, \hat{O}_n) f(Q_n) + \max_{o'} Q_n(\hat{S}_{n+1}, o') - Q_n(\hat{S}_n, \hat{O}_n)}{L_n(\hat{S}_n, \hat{O}_n)} \right), \\
&Q_{n+1}(\hat{S}_n, \hat{O}_n) = Q_n(\hat{S}_n, \hat{O}_n), \qquad (4.42)
\end{aligned}$$

where $f : \mathbb{R}^{\mathcal{S} \times \mathcal{O}} \to \mathbb{R}$ is defined as follows. For any $q \in \mathbb{R}^{\mathcal{S} \times \mathcal{O}}$,

$$f(q) \doteq \eta \sum q - \eta \sum Q_0 + \bar{R}_0. \qquad (4.43)$$

We now show that (4.42) is a special case of the General RVI Q's update (3.9). To see this point, we hypothesize three $|\mathcal{S} \times \mathcal{O}|$-sized random processes $\{S'_n\}, \{R'_n\}, \{L'_n\}$ such that for any n, s, o, $S'_n(s, o), R'_n(s, o), L'_n(s, o) \sim p(\cdot, \cdot, \cdot \mid s, o)$. Now consider the stream of experience $\ldots, \hat{S}_n, \hat{O}_n, \hat{R}_{n+1}, \hat{L}_{n+1}, \hat{S}_{n+1}, \ldots$. Then we have $\hat{S}_{n+1} = S'_n(S_n, O_n)$, $\hat{R}_{n+1} = R'_n(S_n, O_n)$, $\hat{L}_{n+1} = L'_n(S_n, O_n)$.

Equation 3.9 reduces (4.42) by choosing $i = (s, o), Y_n = \{(\hat{S}_n, \hat{O}_n)\}$,

$$
\begin{aligned}
r(i) &= \hat{r}(s, o)/\hat{l}(s, o), \\
g(Q_n)(i) &= \frac{\sum_{s'} \hat{p}(s' \mid s, o) \max_{o'} Q_n(s', o')}{\hat{l}(s, o)} + \frac{\hat{l}(s, o) - 1}{\hat{l}(s, o)} Q_n(s, o), \\
M_{n+1}(i) &= \frac{R'_n(s, o) - \hat{r}(s, o)}{\hat{l}(s, o)} \\
&\quad + \frac{\max_{o'} Q_n(S'_n(s, o), o') - \sum_{s'} \hat{p}(s' \mid s, o) \max_{o'} Q_n(s', o')}{\hat{l}(s, o)}, \\
\epsilon_{n+1}(i) &= \frac{R'_n(s, o) - L_n(s, o) f(Q_n) + \max_{o'} Q_n(S'_n(s, o), o') - Q_n(s, o)}{L_n(s, o)} \\
&\quad - \frac{R'_n(s, o) - \hat{l}(s, o) f(Q_n) + \max_{o'} Q_n(S'_n(s, o), o') - Q_n(s, o)}{\hat{l}(s, o)}.
\end{aligned}
$$

We now show that the assumptions required by General RVI Q are all satisfied.

1. Assumption 3.1 can be verified for $g(q)$ easily.

2. Assumption 3.2 is satisfied. Note that (3.8) reduces to (4.29) in the current setting. Because the MDP $\mathcal{M}$ is weakly communicating, $r_{\#} = \hat{r}_*$ is the only solution of $\bar{r}$ in (4.29). Furthermore, there are multiple solutions of q in (4.29)

3. Assumption 3.3 can be verified easily for f defined in (4.43)

4. Assumption 3.4 holds because of Corollary 4.1.

5. Assumptions 3.5, 3.6 are given in the theorem statement.

6. Assumption 3.7 (i) holds because the state, option, and action spaces are all finite.

7. Assumption 3.7 (ii) holds. Note that $\hat{L}_n$ has a finite variance by Proposition 4.2, therefore standard stochastic approximation result (Blum 1954) can be applied to show that L_n converges to $\hat{l}$ almost surely given the assumption on β_n. Because $L_n > \mathbf{0}$ (note that $L_0 > \mathbf{0}$ and options execution time steps are always no less than a single time step) and ϵ_n

is a continuous function of L_n for all $L_n > \mathbf{0}$, by continuous mapping theorem, ϵ_n converges to $\mathbf{0}$ almost surely.

All the assumptions required by Theorem 3.1 are verified, thus a.s. $Q_n \to \mathcal{Q}_\#$, which is $\hat{\mathcal{Q}}_\infty$ in the current setting. The rest of the proof follows Lemma 4.3.

4.6 Inter-Option Experiments

This section empirically studies both inter-option prediction and control learning algorithms introduced in the previous section.

Throughout this chapter, the test domain is again the four-room domain (c.f. Figure 3.1), except that in this chapter for different experiments, different rewarding states are chosen. See Figure 4.2 for a visualization of the four-room domain used in this chapter. For each experiment, among states `G1`, `G2`, and `G3`, only one of them is the rewarding state, and the other two states are just normal states. All states' rewards are zero, except for the rewarding state, which has a reward of one regardless of the action taken at the state. In addition, the dynamics for all states are normal except for the rewarding state—taking any action from the rewarding state moves the agent to the yellow cell.

In addition to the four primitive actions, the agent has eight options that

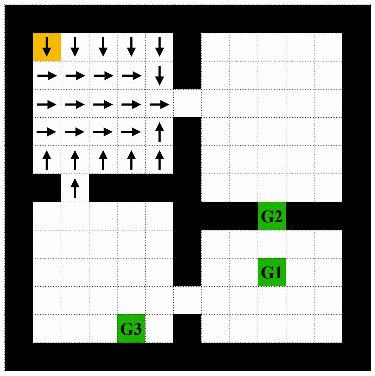


Figure 4.2: A continuing variant of the four-room domain where the task is to repeatedly go from the yellow start state to one of the three green rewarding states. There is one rewarding state per experiment, chosen to demonstrate particular aspects of the proposed algorithms. Also shown is an option's policy to go to the upper hallway cell.

take it from a given room to the hallways adjoining the room. The arrows in Figure 4.2 illustrate the policy of one of the eight options. For this option, the policy in the empty cells (not marked with arrows) is to uniformly randomly pick among the four primitive actions. The termination probability is 0 for all cells with arrows and 1 for the empty cells. The other seven options are defined similarly. Denote the set of primitive actions as $\mathcal{A}$ and the set of hallway options as $\mathcal{H}$. Including the primitive actions, the agent has 12 options in total.

The first set of experiments is to show that inter-option Differential Q-evaluation-learning can learn the reward rate and the relative values well, confirming Theorem 4.4. For this first set of experiments, I used `G1` as the rewarding state. The target policy is an optimal hierarchical policy, which repeatedly moves the agent from the starting state to the rewarding state with the fewest number of steps. The shortest path to `G1` from the starting state takes 16 time steps, and there is an extra step taking the agent back to the yellow cell from `G1`, hence the best possible reward rate for this task is $1/17 \approx 0.0588$. For each state, the behavior policy picks the target policy's action w.p. 0.9 and picks a random action w.p. 0.1. For each of the two step-size sequences α_n and β_n, I tested five choices: $2^{-x}, x \in \{1, 3, 5, 7, 9\}$. In addition, I tested five choices of η : $10^{-x}, x \in \{0, 1, 2, 3, 4\}$. Q and $\bar{R}$ were initialized to 0, L to 1. Each parameter setting was run for $500,000$ steps and repeated 30 times.

Just as in Section 3.4, the error metrics here are

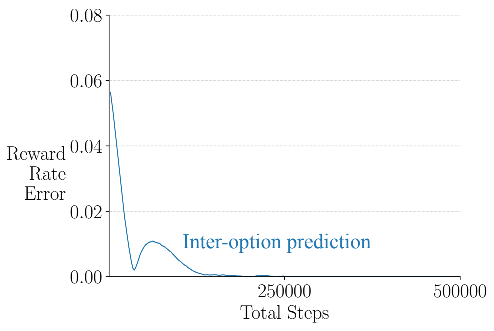
$$\text{Reward Rate Error}_t \doteq |\hat{r}_* - \bar{R}_t|,$$

where $\bar{R}_t \doteq \bar{R}_n$ for all $T_n \leq t < T_{n+1}$, and

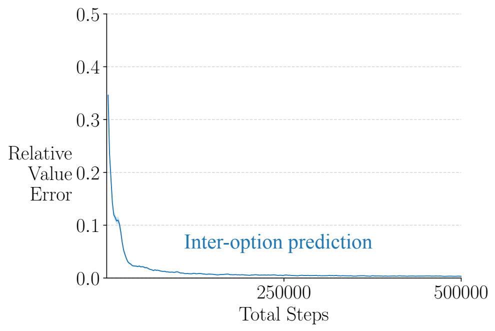
$$\text{Relative Value Error}_t \doteq |(\hat{q}_\infty(S_t, O_t) - \hat{q}_\infty(s_0, o_0)) - (Q_n(S_t, O_t) - Q_n(s_0, o_0))|,$$

where $Q_t \doteq Q_n$ for all $T_n \leq t < T_{n+1}$.

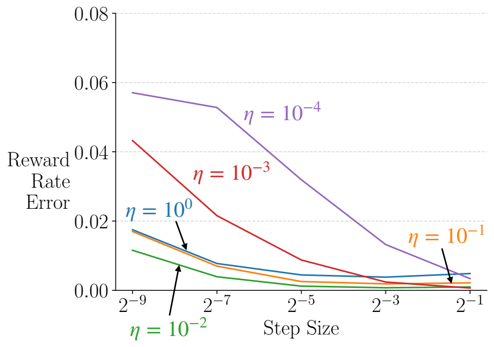
The learning curves of the two errors are shown in Figure 4.3a and Figure 4.3b, with parameter setting chosen to be the best asymptotically (smallest errors averaged over the last $10,000$ steps). The sensitivity curves of the two



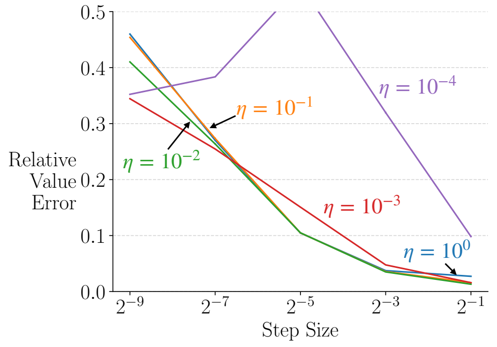
(a) Each point marks the reward rate error averaged over the past $2,000$ steps. Parameters were chosen to be to minimize errors over the last $10,000$ steps. Here these parameters are $\alpha = 2^{-1}, \eta = 10^{-4}, \beta = 2^{-9}$.



(b) Each point marks the relative value error averaged over the past $2,000$ steps. Parameters were chosen to be to minimize errors over the last $10,000$ steps. Here these parameters are $\alpha = 2^{-1}, \eta = 10^{-3}, \beta = 2^{-3}$



(c) Each point is the reward rate error averaged over the entire $500,000$ steps. The error bars denote one standard error. The x-axis is the step size α. $\beta = 1/2$. Each curve corresponds to one choice of η.



(d) Each point is the relative value error averaged over the entire $500,000$ steps. The error bars and the x-axis have the same meaning as in the left plot.

Figure 4.3: Plots showing learning curves and parameter studies for inter-option Differential Q-evaluation-learning in the continuing four-room domain when the goal was to go to `G1`. The algorithm used the set of primitive actions and the set of hallway options $\mathcal{O} = \mathcal{A} + \mathcal{H}$.

errors w.r.t. α and η with $\beta = 1/2$ are shown in Figure 4.3c and Figure 4.3d. It turns out the algorithm is not sensitive to the choice of β in this experiment. The sensitivity curves of the two errors with other choices of β are deferred to Section B.1.

The learning curves show that the algorithm indeed learned the reward rate and the differential value function (up to an additive constant) of the target policy, when following a different behavior policy. The sensitivity curves show that the prediction errors were not sensitive to the choice of η for a relatively large η ($\geq 10^{-2}$ in this experiment).

In the second experiment, I tested inter-option Differential Q-learning with three different sets of options, $\mathcal{O} \in \{\mathcal{A}, \mathcal{H}, \mathcal{A}+\mathcal{H}\}$. This experiment is to show that learning with a set of hallway options can be faster than it with only primitive actions. The task is again to reach the green cell `G1`, which the agent can achieve with a combination of options and primitive actions. The tested parameters are the same as those used in inter-option Differential Q-evaluation-learning experiments. Figure 4.4 shows a typical learning curve for each of the three sets of options, with $\alpha = 2^{-3}$, $\beta = 2^{-1}$, and $\eta = 10^{-1}$.

The learning curves in Figure 4.4 show that the agent achieved a relatively stable reward rate after $300,000$ steps in all three cases. Using just primitive actions $\mathcal{A}$, the learning curve rose the slowest, indicating that hallway options indeed helped the agent reach the goal faster. Using just primitive actions or using primitive actions and hallway options, the learning curve achieved the optimal reward rate of 0.0588. But solely using the hallway options $\mathcal{H}$ did not achieve the optimal reward rate as the goal `G1` is not a hallway state. These observations mirror those by Sutton, Precup, and Singh (1999) with the discounted formulation.

The third set of experiments is to study the parameter sensitivity of inter-option Differential Q-learning. To this end, I performed a parameter study for $\mathcal{O} = \mathcal{A} + \mathcal{H}$ w.r.t. α and η, with $\beta = 2^{-1}$ (left three subfigures of Figure 4.5). It turns out the algorithm was not sensitive to the choice of β in this experiment and the sensitivity curves w.r.t. other choices of β are deferred to Section B.1. I also tested Gosavi's (2004) algorithm as a baseline. I chose not to compare the proposed algorithms in this section with Sutton et al.'s (1999) discounted versions because the discounted and average-reward problem formulations are different; comparing the performance of their respective solution methods would be inappropriate and difficult to interpret. Gosavi's (2004) algorithm estimates the reward rate by tracking the cumulative reward

$\bar{C}$ obtained by the options and dividing it by another estimate $\bar{T}$ that tracks the length of the options. If the n^{th} option executed is a greedy choice, then these estimates are updated using:

$$\begin{aligned}
\bar{C}_{n+1} &\doteq \bar{C}_n + \beta_n(\hat{R}_n - \bar{C}_n), \\
\bar{T}_{n+1} &\doteq \bar{T}_n + \beta_n(\hat{L}_n - \bar{T}_n), \\
\bar{R}_{n+1} &\doteq \bar{C}_{n+1}/\bar{T}_{n+1}.
\end{aligned}$$

When $\hat{O}_n$ is not greedy, $\bar{R}_{n+1} = \bar{R}_n$. The option-value function is updated with (4.30) using δ_n as defined in (4.32). α_n and β_n are two step-size sequences. The sensitivity of this algorithm with $\mathcal{O} = \mathcal{A} + \mathcal{H}$ is shown in the right three subfigure of Figure 4.5.

The sensitivity curves of inter-option Differential Q-learning (left three subfigures of Figure 4.5) indicate that, in this four-room domain, the algorithm was not sensitive to parameter η unless η was too small, performed well for a wide range of step sizes α, and showed low variance across different runs. Compared to inter-option Differential Q-learning, Gosavi's (2004) algorithm has one less parameter, but the speed of policy learning was found to be more sensitive to the values of both its step-size parameters. In addition, both errors are much larger, compared with inter-option Differential Q-learning. The error

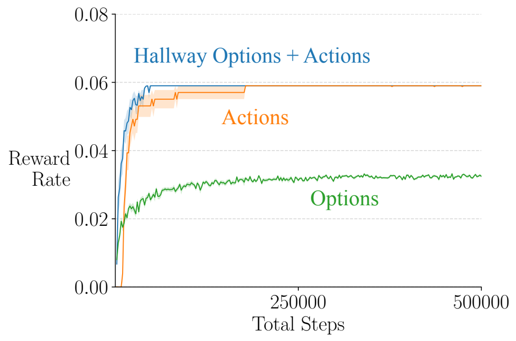


Figure 4.4: Plots showing some learning curves and the parameter study of inter-option Differential Q-learning on the continuing four-room domain when the goal was to go to `G1`. A point on the solid line denotes the reward rate over the last 2000 time steps and the shaded region indicates one standard error. The behavior using the three different sets of options was as expected.

Algorithm 9: Gosavi's (2004) SMDP algorithm

```
Input: Behavioral policy $b$'s parameters (e.g., $\epsilon$ for $\epsilon$-greedy)
Algorithm parameters: step-size sequences $\alpha_n, \beta_n$
1  Initialize $Q(s,o)\ \forall\, s \in \mathcal{S}, o \in \mathcal{O}, \bar{R}, C, T$ arbitrarily (e.g., to zero);
2  $\nu(s,o) \leftarrow 0\ \forall s, o$
3  $N \leftarrow 0$
4  Obtain initial $S$
5  while still time to train do
6      Initialize $\hat{L} \leftarrow 0, \hat{R} \leftarrow 0, S_{tmp} \leftarrow S$
7      $O \leftarrow$ option sampled from $b(\cdot \mid S)$
8      $\nu(S,O) \leftarrow \nu(S,O) + 1$
9      do
10         Sample primitive action $A \sim \pi(\cdot \mid S, O)$
11         Take action $A$, observe $R, S'$
12         $\hat{L} \leftarrow \hat{L} + 1$
13         $\hat{R} \leftarrow \hat{R} + R$
14         $S \leftarrow S'$
15     while $O$ doesn't terminate in $S'$
16     $S \leftarrow S_{tmp}$
17     $\delta \leftarrow \hat{R} - \bar{R} \cdot \hat{L} + \max_o Q(S', o) - Q(S, O)$
18     $Q(S,O) \leftarrow Q(S,O) + \alpha_{\nu(S,O)}\delta$
19     if $O \in \operatorname{argmax} Q(S, \cdot)$ then
20         $C \leftarrow C + \beta_N(\hat{R} - C)$
21         $L \leftarrow L + \beta_N(\hat{L} - L)$
22         $\bar{R} \leftarrow C/L$
23     end
24     $S \leftarrow S'$
25     $N \leftarrow N + 1$
26 end
27 return $Q$
```

bars were also generally larger, suggesting that the variance across different runs was higher.

To conclude, the experiments with the continuing four-room domain show that inter-option Differential Q-learning indeed finds the optimal policy given a set of options, in accordance with Theorem 4.2. In addition, its performance seems more robust to the choices of parameters and also learns the reward rate and differential action-value function better compared to the baseline.

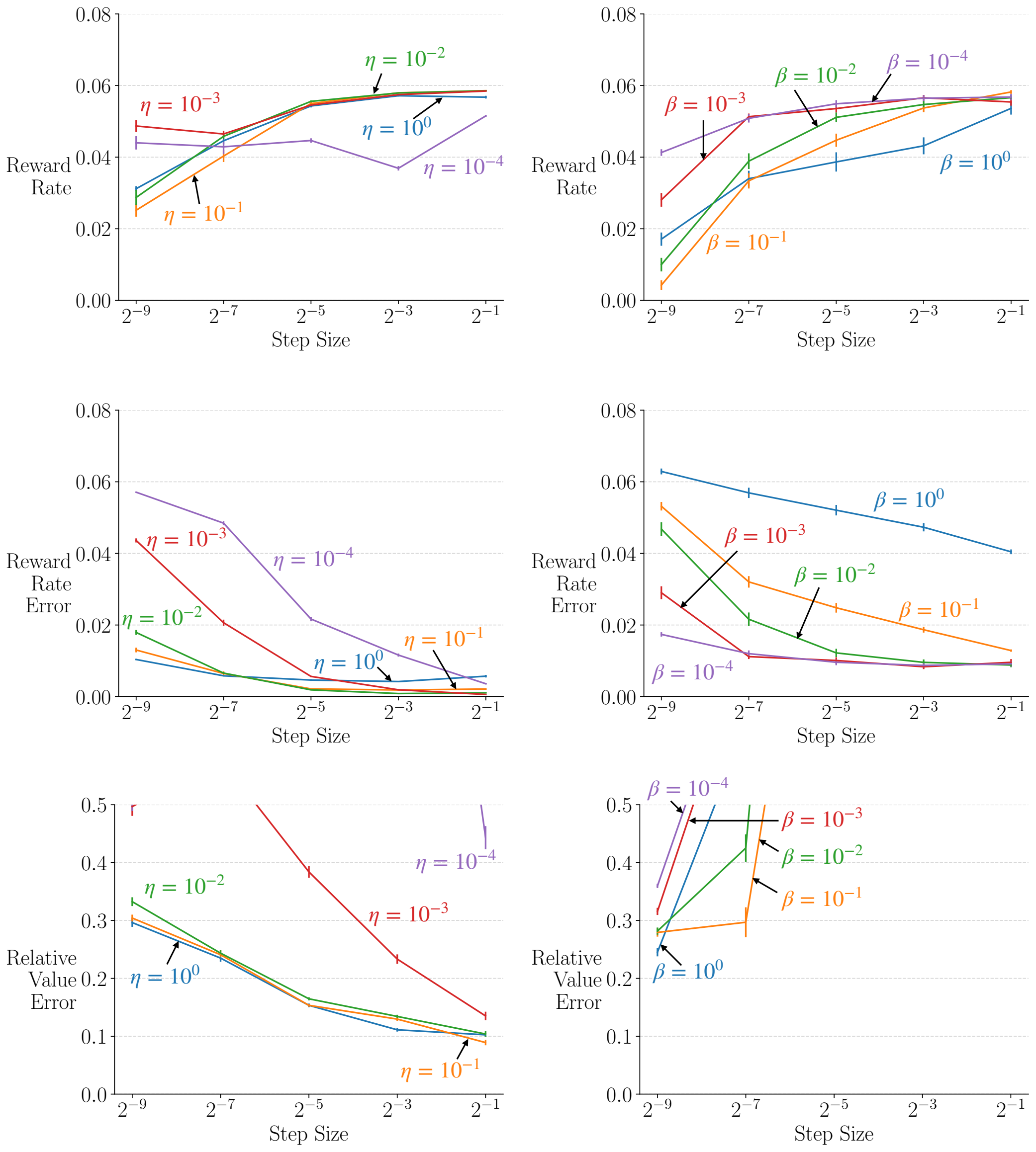


Figure 4.5: Parameter studies showing our inter-option Differential Q-learning's rate of learning performed well for a wider range of parameters, compared to the baseline algorithm (Gosavi 2004). $\mathcal{O} = \mathcal{A} + \mathcal{H}$ and $\beta = 2^{-1}$ in all six subfigures. The left three subfigures show the sensitivity curves of inter-option Q-learning's reward rate achieved by its learned policy, reward rate error, and relative value error. The right three subfigures show the same quantities but for Gosavi's (2004) algorithm. The experiment setting and the plot axes are the same as mentioned in Figure 4.3's caption.

4.7 Intra-Option Value Learning and Planning Algorithms

This section introduces intra-option value learning and planning algorithms. The objectives are the same as that of inter-option value learning algorithms. Intra-option algorithms learn from every transition $S_t, A_t, R_{t+1}, S_{t+1}$ that may not even be generated by following a hierarchical policy. Moreover, intra-option algorithms also make updates for *all* options, including ones that may potentially never be executed.

Intra-option prediction and control algorithms are stochastic approximation algorithms solving the intra-option evaluation and optimality equations respectively. Intra-option prediction methods find the reward rate and the differential option-value function of a hierarchical policy by solving the *intra-option evaluation* equation.

$$q(s,o) = \sum_a \pi(a \mid s,o) \sum_{s',r} p(s',r \mid s,a)(r - \bar{r} + u^q(s',o)), \quad \forall\ s \in \mathcal{S}, o \in \mathcal{O}, \tag{4.44}$$

where

$$u^q(s',o) = u^q_\mu(s',o) \doteq \big(1 - \beta(s',o)\big)q(s',o) + \beta(s',o)\sum_{o'} \mu(o' \mid s')q(s',o'). \tag{4.45}$$

Intra-option control methods find an optimal policy by solving the *intra-option optimality* equation (4.44), where

$$u^q(s',o) = u^q_*(s',o) \doteq \big(1 - \beta(s',o)\big)q(s',o) + \beta(s',o)\max_{o'} q(s',o'). \tag{4.46}$$

The following proposition shows that the set of solutions of the SMDP evaluation/optimality equation is the same as that of the intra-option evaluation/optimality equation.

Proposition 4.3. *Any solution of* (4.25) *is also a solution of* (4.44) *with* u^q *defined in* (4.45) *and vice versa. Any solution of* (4.29) *is also a solution of* (4.44) *with* u^q *defined in* (4.46) *and vice versa.*

Proof. We only show the equality for optimality equations. The equality for evaluation equations can be derived similarly.

$$\begin{aligned}
&q(s,o)\\
&=\sum_{s',r,l}\hat{p}(s',r,l \mid s,o)(r-l\bar{r}+\max_{o'}q(s',o'))\\
&=\sum_{s',r,l}\sum_{a}\pi(a \mid s,o)\sum_{\tilde{s},\tilde{r}}p(\tilde{s},\tilde{r} \mid s,a)[\beta(\tilde{s},o)\mathbf{I}\{s'=\tilde{s},r=\tilde{r},l=1\}\\
&+(1-\beta(\tilde{s},o))\hat{p}(s',r-\tilde{r},l-1 \mid \tilde{s},o)](r-l\bar{r}+\max_{o'}q(s',o')) \qquad \text{By (4.24)}\\
&=\sum_{a}\pi(a \mid s,o)\sum_{\tilde{s},\tilde{r}}p(\tilde{s},\tilde{r} \mid s,a)\Big(\beta(\tilde{s},o)(\tilde{r}-\bar{r}+\max_{o'}q(\tilde{s},o'))\\
&+(1-\beta(\tilde{s},o))\sum_{s',r,l}\hat{p}(s',r,l \mid \tilde{s},o)](r+\tilde{r}-(l+1)\bar{r}+\max_{o'}q(\tilde{s},o'))\Big).
\end{aligned}$$

Note that $\sum_{s',r,l}\hat{p}(s',r,l \mid s',o)](r+\tilde{r}-(l+1)\bar{r}+\max_{o'}q(\tilde{s},o'))=\tilde{r}-\bar{r}+\sum_{s',r,l}\hat{p}(s',r,l \mid s',o)(r-l\bar{r}+\max_{o'}q(\tilde{s},o'))=r-\bar{r}+q(\tilde{s},o)$. Therefore,

$$\begin{aligned}
q(s,o)&=\sum_{a}\pi(a \mid s,o)\sum_{\tilde{s},\tilde{r}}p(\tilde{s},\tilde{r} \mid s,a)\\
&\Big(r-\bar{r}+\beta(\tilde{s},o)\max_{o'}q(\tilde{s},o')+(1-\beta(\tilde{s},o))q(\tilde{s},o)\Big)\\
&=\sum_{a}\pi(a \mid s,o)\sum_{\tilde{s},\tilde{r}}p(\tilde{s},\tilde{r} \mid s,a)(r-\bar{r}+u_*^q(\tilde{s},o)).
\end{aligned}$$

Therefore any solution of (4.29) must be a solution of (4.44) and (4.46) and vice versa. □

I now start to introduce the intra-option algorithms, starting from the control learning algorithm.

Intra-option learning algorithms operate on a stream of experience generated by following some history-dependent behavior policy that chooses actions $b_t : \mathcal{H}_t \to \mathcal{A}$, where $\mathcal{H}_t$ is the set of all possible histories up to time step t. Here the history up to time step t includes all the information revealed up to time step t, including states and rewards observed up to time step t, and all actions and options (if they are used) chosen before time step t. This broader class of experience includes the experience generated by following a hierarchical policy as a special case. Let S_t denote the state and R_t denote the reward observed at time step t, and let A_t denote the action chosen at time step t.

Both the prediction and control algorithms maintain a vector of estimates $Q(s, o)$ and a scalar estimate $\bar{R}$, just like our inter-option algorithms. However, unlike inter-option algorithms, intra-option algorithms need not maintain an estimator for option lengths (L) because they make updates after every transition. Our control algorithm, called *intra-option Differential Q-learning*, updates the estimates Q and $\bar{R}$ by:

$$Q_{t+1}(S_t, o) \doteq Q_t(S_t, o) + \alpha_{\nu_t(S_t,o)} \rho_t(o) \delta_t(o), \quad \forall\ o \in \mathcal{O}, \tag{4.47}$$

$$\bar{R}_{t+1} \doteq \bar{R}_t + \eta \sum_{o \in \mathcal{O}} \alpha_{\nu_t(S_t,o)} \rho_t(o) \delta_t(o), \tag{4.48}$$

where α_n is a step-size sequence, $\rho_t(o) \doteq \pi(A_t \mid S_t, o)/b(A_t \mid H_t)$ for all $t \geq 0$ is the importance sampling ratio, and:

$$\delta_t(o) \doteq R_{t+1} - \bar{R}_t + u_*^{Q_t}(S_{t+1}, o) - Q_t(S_t, o). \tag{4.49}$$

Similar to Assumption 3.9, I will assume the following.

Assumption 4.2. There exists some positive δ such that if $\pi(a \mid s, o) > 0$, then $b(a \mid h) > \delta$ for all $h \in \mathcal{H}_t$ ending with s, $t \geq 0$.

This assumption guarantees that $\rho_t(o)$ is bounded above by a finite number.

Remark: The intra-option learning methods introduced in this section can be used with options having stochastic policies. This is possible with the use of the important sampling ratios as described above. Sutton et al.'s (1999) discounted intra-option learning methods were restricted to options having deterministic policies.

Theorem 4.6. *Given an MDP and a set of options, if the resulting SMDP is weakly communicating, Assumptions 3.5, 3.6 (with $\mathcal{I} = \mathcal{S} \times \mathcal{O}$), and Assumption 4.2 hold, intra-option Differential Q-learning* (4.47)—(4.49) *converges almost surely, 1) $\bar{R}_t$ to $\hat{r}_*$, 2) Q_t to a sample-path dependent compact connected subset of $\hat{\mathcal{Q}}_\infty$, and 3) almost surely, $\mu_t \in \hat{\Pi}_*^D$ for sufficiently large t, where μ_t is a greedy policy w.r.t. Q_t.*

Algorithm 10: Intra-option Differential Q-learning

```
Input: Behavioral policy $b$'s parameters (e.g., $\epsilon$ for $\epsilon$-greedy)
Algorithm parameters: step-size sequence $\alpha_n$, parameter $\eta$
1  Initialize $Q(s,o)\ \forall\, s \in \mathcal{S}, o \in \mathcal{O}, \bar{R}$ arbitrarily (e.g., to zero)
2  $\nu(s,o) \leftarrow 0\ \forall s, o$
3  Obtain initial $S$
4  $H \leftarrow S$
5  while still time to train do
6      Sample primitive action $A \sim b(\cdot \mid H)$
7      Take action $A$, observe $R, S'$
8      $\Delta = 0$
9      for all options $o$ do
10         $\nu(S,o) \leftarrow \nu(S,o) + 1$
11         $\rho \leftarrow \pi(A \mid S, o)/b(A \mid H)$
12         $\delta \leftarrow R - \bar{R} + \Big(\big(1 - \beta(S',o)\big)Q(S',o) +$
             $\beta(S',o) \max_{o'} Q(S',o')\Big) - Q(S,o)$
13         $Q(S,o) \leftarrow Q(S,o) + \alpha_{\nu(S,o)}\rho\delta$
14         $\Delta \leftarrow \Delta + \eta\alpha_{\nu(S,o)}\rho\delta$
15     end
16     $\bar{R} \leftarrow \bar{R} + \Delta$
17     $S \leftarrow S'$
18     $H \leftarrow (H, A, R, S')$
19 end
20 return $Q$
```

If intra-option Differential Q-learning is applied to a simulated experience generated from a model of the world, then it becomes a planning algorithm, which we call *intra-option Differential Q-planning.* Unlike the model used by inter-option Differential Q-learning, which is an option model, the model used by intra-option Differential Q-planning is a standard model that predicts the outcomes of actions. Formally, the model is a function $\tilde{p} : \mathcal{S} \times \mathcal{A} \to \Delta(\mathcal{S} \times \mathcal{R})$, analogous to p, that, like p, sums to 1: $\sum_{s',r} \tilde{p}(s', r \mid s, a) = 1$ for all s, a. A model MDP can be thus constructed using $\tilde{p}$ and $\mathcal{S}, \mathcal{A}, \mathcal{R}, d_0$.

The simulated transitions are generated as follows: at each planning step n, the agent arbitrarily chooses a state $\tilde{S}_n$ and chooses an action $\tilde{A}_n$ according to some history-dependent policy b: $b(\tilde{A}_n \mid \tilde{H}_n)$, where $\tilde{H}_n \in \mathcal{H}_n$, then applies $\tilde{p}$ to generate a simulated resulting state and reward $\tilde{S}'_n, \tilde{R}_n, \sim \tilde{p}\left(\cdot, \cdot \mid \tilde{S}_n, \tilde{A}_n\right)$.

Let $\tilde{r}_*, \tilde{\Pi}^D_*, \tilde{\mathcal{Q}}_\infty$ be defined in the same way as $\hat{r}_*, \hat{\Pi}^D_*, \hat{\mathcal{Q}}_\infty$ but for the SMDP resulting from the model MDP and the set of options.

Like intra-option Differential Q-learning, intra-option Differential Q-planning maintains a table of value estimates $Q_n : \mathcal{S} \times \mathcal{O} \to \mathbb{R}$ and a reward-rate estimate $\bar{R}_n$. At each planning step n, these estimates are updated by (4.47)–(4.49), just as in intra-option Differential Q-learning, except now using $\tilde{S}_n, \tilde{A}_n, \tilde{R}_n, \tilde{S}'_n$ instead of $S_t, A_t, R_{t+1}, S_{t+1}$.

Theorem 4.7. *Given a model MDP and a set of options, if the resulting SMDP is weakly communicating, Assumptions 3.5, 3.6 (with $\mathcal{I} = \mathcal{S} \times \mathcal{O}$) and Assumption 4.2 hold, intra-option Differential Q-planning converges almost surely, 1) $\bar{R}_n$ to $\tilde{r}_*$, 2) Q_n to a sample-path dependent compact connected subset of $\tilde{\mathcal{Q}}_\infty$, and 3) almost surely, $\mu_n \in \tilde{\Pi}^D_*$ for a sufficiently large n, where μ_n is any greedy policy w.r.t. Q_n.*

The prediction learning algorithm, called *intra-option Differential Q-evaluation-learning*, also updates Q and $\bar{R}$ by (4.47) and (4.48), but with the TD error:

$$\delta_t(o) \doteq R_{t+1} - \bar{R}_t + u^{Q_t}_\mu(S_{t+1}, o) - Q_t(S_t, o). \tag{4.50}$$

Theorem 4.8. *Given an MDP and a set of options, if the hierarchical policy μ is unichain in the resulting SMDP, and Assumptions 3.5, 3.6 (with $\mathcal{I} = \mathcal{S} \times \mathcal{O}$) hold, intra-option Differential Q-evaluation-learning (Equations 4.47,4.48,4.50) converges almost surely, $\bar{R}_t$ to $\hat{r}(\mu)$, $Q_t(s, o)$ to $\hat{q}_\infty$.*

Like intra-option Differential Q-evaluation-learning, intra-option Differential Q-evaluation-planning maintains a table of value estimates $Q_n : \mathcal{S} \times \mathcal{O} \to \mathbb{R}$ and a reward-rate estimate $\bar{R}_n$. At each planning step n, these estimates are updated by (4.47),(4.48), and (4.50), just as in intra-option Differential Q-evaluation-learning, except now using the simulated experience $\ldots, \tilde{S}_n, \tilde{O}_n, \tilde{A}_n, \tilde{R}_n, \tilde{S}'_n, \ldots$ instead of the real experience $\ldots, S_t, O_t, A_t, R_{t+1}, S_{t+1}, \ldots$. Let $\tilde{r}(\mu), \tilde{q}_\infty$ be defined the same way as $\hat{r}(\mu), \hat{q}_\infty$ but for the SMDP resulting from the model MDP rather than the real MDP.

```
Algorithm 11: Intra-option Differential Q-evaluation-learning
   Input: Behavioral policy $b$, target policy $\mu$
   Algorithm parameters: step-size sequence $\alpha_n$, parameter $\eta$
 1 Initialize $Q(s,o)\ \forall\, s \in \mathcal{S}, o \in \mathcal{O}, \bar{R}$ arbitrarily (e.g., to zero)
 2 $\nu(s,o) \leftarrow 0\ \forall s, o$
 3 Obtain initial $S$
 4 $H \leftarrow S$
 5 while still time to train do
 6     Sample primitive action $A \sim b(\cdot \mid H)$
 7     Take action $A$, observe $R, S'$
 8     $\Delta \leftarrow 0$
 9     for all options $o$ do
10         $\nu(S,o) \leftarrow \nu(S,o) + 1$
11         $\rho \leftarrow \pi(A \mid S,o)/b(A \mid H)$
12         $\delta \leftarrow R - \bar{R} + \Big(\big(1-\beta(S',o)\big)Q(S',o) + \beta(S',o)\sum_{o'}\mu(o' \mid S')Q(S',o')\Big) - Q(S,o)$
13         $Q(S,o) \leftarrow Q(S,o) + \alpha_{\nu(S,o)}\rho\delta$
14         $\Delta \leftarrow \Delta + \eta\alpha_{\nu(S,o)}\rho\delta$
15     end
16     $\bar{R} \leftarrow \bar{R} + \Delta$
17     $S \leftarrow S'$
18     $H \leftarrow (H, A, S', R)$
19 end
20 return $Q$
```

Theorem 4.9. *Given a model MDP and a set of options, under the same assumptions as in Theorem 4.8, intra-option Differential Q-evaluation-planning converges almost surely, $\bar{R}_t$ to $\tilde{r}(\mu)$, $Q_t(s,o)$ to $\tilde{q}_\infty$.*

I now provide proof for our *control learning* algorithm (Theorem 4.6). The proofs for other algorithms introduced in this section are very similar and are thus omitted.

Proof of Theorems 4.6

Following a similar transformation as for Differential Q-learning, by (4.47)—(4.49) we have $\forall\ s \in \mathcal{S}, o \in \mathcal{O}$:

$$Q_{t+1}(S_t,o) = Q_t(S_t,o)$$
$$+ \alpha_{\nu_t(S_t,o)}\rho_t(o)\Bigg(R_{t+1} - f(Q_t) + u_*^{Q_t}(S_{t+1},o) - Q_t(S_t,o)\Bigg), \forall o \in \mathcal{O}_t, \quad (4.51)$$

where f is defined in (4.43).

We first show that (4.51) is a special case of the General RVI Q's update (3.9). We hypothesize two $|\mathcal{S}\times\mathcal{O}|$-sized i.i.d. random processes $\{S'_t\}, \{R'_t\}$ such that for any t, s, o, $S'_t(s,o), R'_t(s,o) \sim p(\cdot,\cdot \mid s, A_t)$. Now consider the stream of experience $\ldots, S_t, A_t, R_{t+1}, S_{t+1}, \ldots$. Then we have $S_{t+1} = S'_t(S_t, A_t)$, $R_{t+1} = R'_t(S_t, A_t)$. Let $Y_t = \{(S_t, o) | o \in \mathcal{O}_t\}$. Then (4.51) can be rewritten as, for each $s \in \mathcal{S}, o \in \mathcal{O}$,

$$\begin{aligned} Q_{t+1}(s,o) &= Q_t(s,o) + \alpha_{\nu_t(s,o)} \frac{\pi(A_t \mid s,o)}{b(A_t \mid H_t)} \\ &\Bigg(R'_t(s,A_t) - f(Q_t) + \beta(S'_t(s,A_t) \max_{o'} Q_t(S'_t(s,A_t),o') \\ &+ (1-\beta(S'_t(s,A_t),o))Q_t(S'_t(s,A_t),o) - Q_t(s,o)\Bigg) \mathbf{I}\{(s,o)\in Y_t\}. \end{aligned} \tag{4.52}$$

Equation 3.9 reduces to (4.52) by choosing $\mathcal{I} = \mathcal{S}\times\mathcal{O}, Y_n = Y_t, Q_n = Q_t$, and for each $i \in \mathcal{I}$,

$$\begin{aligned} &r(i) = r'(s,o) \doteq \sum_a \pi(a \mid s,o) \sum_{s',r} p(s',r \mid s,a) r, \\ &g(q)(i) = \sum_a \pi(a \mid s,o) \sum_{s'} p(s' \mid s,a) \\ &(\beta(s',o) \max_{o'} q(s',o') + (1-\beta(s',o))q(s',o) - q(s,o)), \\ &M_{t+1}(i) = 0 \text{ if } (s,o) \notin Y_t, \text{ otherwise} \\ &M_{t+1}(i) = \frac{\pi(A_t \mid s,o)}{b(A_t \mid H_t)} \Bigg(R'_t(s,A_t) - f(Q_t) + \beta(S'_t(s,A_t),o) \max_{o'} Q_t(S'_t(s,A_t),o') \\ &+ (1-\beta(S'_t(s,A_t),o))Q_t(S'_t(s,A_t),o) - Q_t(s,o)\Bigg) \\ &- (r'(s,o) - f(Q_t) + g(Q_t)(s,o) - Q_t(s,o)), \\ &\epsilon_{t+1}(i) = 0. \end{aligned}$$

We now show that the assumptions required by General RVI Q are all satisfied.

1. Assumption 3.1 can be verified for $g(q)$ easily.

2. Assumptions 3.2 and 3.4 are satisfied. Note that (3.8) reduces to (4.44) plus (4.46) by our construction and that (4.44) plus (4.46) is equivalent

to (4.29) by Proposition 4.3. Because the MDP $\mathcal{M}$ is weakly communicating, $r_{\#} = \hat{r}_*$ is the only solution of $\bar{r}$ and the solution of q exists in (4.29) and therefore in (4.46). Finally, given that (4.29) is the action-value optimality equation (4.7) for the resulting SMDP given the MDP and the set of options, the assumption of the SMDP being weakly communicating, and Assumption 3.3, Corollary 4.1 shows Assumption 3.4.

3. Assumption 3.3 is easy to be verified for f defined in (4.43),

4. Assumptions 3.5, and 3.6 are given in the theorem statement

5. Assumption 3.7 is satisfied. Note that if $(s, o) \in Y_t$

$$\mathbb{E}\left[M_{t+1}((s,o)) \mid \mathcal{F}_t\right] = \mathbb{E}\Bigg[\frac{\pi(A_t \mid s,o)}{b(A_t \mid H_t)}$$
$$\Bigg(R'_t(s,A_t) - f(Q_t) + \beta(S'_t(s,A_t),o)\max_{o'} Q_t(S'_t(s,A_t),o')$$
$$+ (1-\beta(S'_t(s,A_t),o))Q_t(S'_t(s,A_t),o) - Q_t(s,o)\Bigg) \mid \mathcal{F}_t\Bigg]$$
$$- (r'(s,o) - f(Q_t) + g(Q_t)(s,o) - Q_t(s,o))$$
$$= \sum_{a \in \mathcal{A}_{t,s}} b(a \mid H_t)\frac{\pi(a \mid s,o)}{b(a \mid H_t)} \sum_{s',r} p(s',r \mid s,a)$$
$$\Big(r - f(Q_t) + \beta(s',o)\max_{o'} Q_t(s',o') + (1-\beta(s',o))Q_t(s',o) - Q_t(s,o)\Big)$$
$$- (r'(s,o) - f(Q_t) + g(Q_t)(s,o) - Q_t(s,o))$$
$$= 0,$$

where $\mathcal{A}_{t,s}$ is the set of all $a \in \mathcal{A}$ such that $b(a \mid H_t) > 0$. The second equation holds because H_t is the history up to time step t and is thus $\mathcal{F}_t$ measurable. The last equation holds because of Assumption 4.2. In addition, $\mathbb{E}\left[\|M_{t+1}\|^2 \mid \mathcal{F}_t\right] < K(1 + \|Q_t\|^2)$, because there $\mathcal{S}, \mathcal{O}, \mathcal{A}$ and $\mathcal{R}$ are all finite and Assumption 4.2.

Therefore Theorem 3.1 applies and we conclude that Q_t converges a.s. to $\mathcal{Q}_{\#}$, which is $\hat{\mathcal{Q}}_\infty$ in the current context. The rest of the proof follows Lemma 4.3.

4.8 Intra-Option Experiments

This section presents two sets of experiments that verify Theorem 4.8 and Theorem 4.6 respectively.

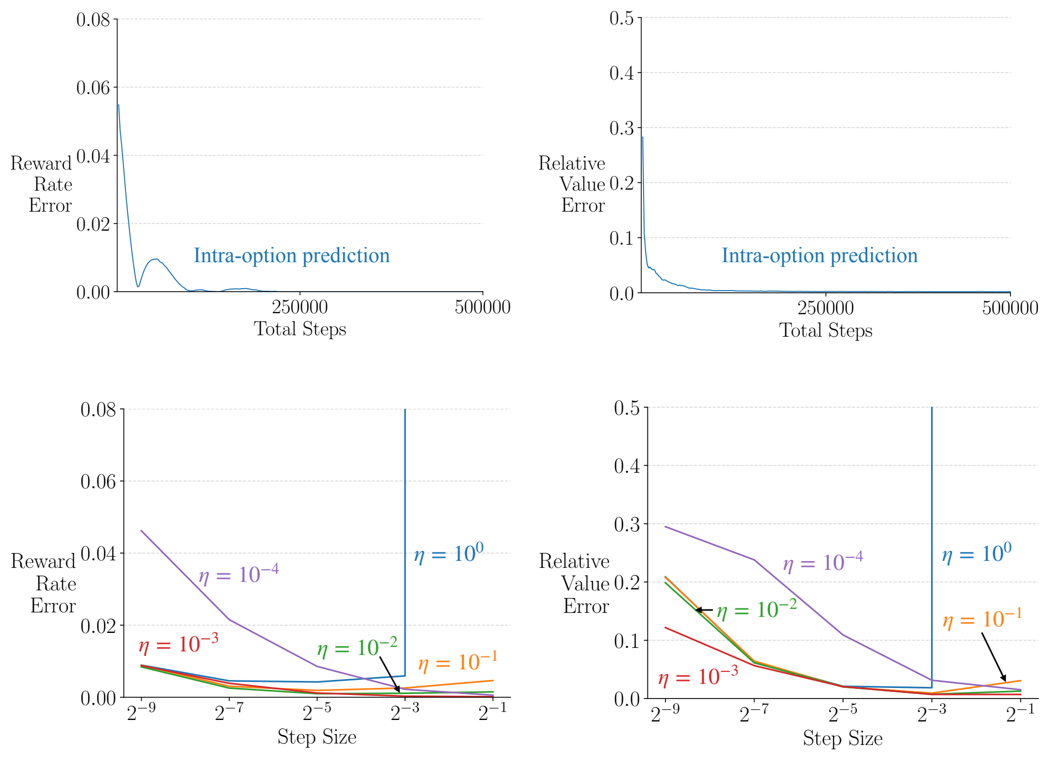
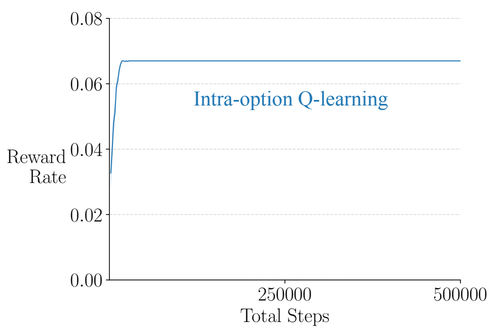
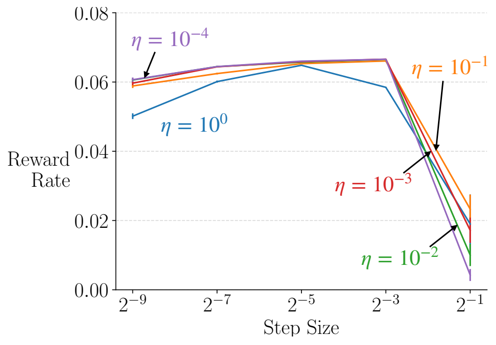


Figure 4.6: Plots showing parameter studies for intra-option Differential Q-evaluation in the continuing four-room domain when the goal was to go to `G1`.

The first set of experiments is to show that intra-option Differential Q-evaluation-learning can learn the reward rate and the relative values well (c.f. Theorem 4.8). The tested world and the setup of the experiment are the same as those in Section 4.6. The learning curves and sensitivity curves are both shown in Figure 4.6. The axes are the same as those in Figure 4.3. The learning curves suggest that intra-option Differential Q-evaluation-learning asymptotically approaches to the true reward rate and relative values, therefore verifying Theorem 4.8. Comparing Figure 4.6 with Figure 4.3 shows that the intra-option algorithm's reward rate and relative value errors were generally better than the inter-option algorithm's unless a large step size like $\alpha = 0.5$ was used.



(a) A learning curve of the intra-option Q-learning algorithm.



(b) Parameter sensitivity curves of the intra-option Q-learning algorithm.

Figure 4.7: A learning curve and parameter sensitivity curves of the intra-option Q-learning algorithm.

I conducted another experiment in the four-room domain to show that intra-option Differential Q-learning can learn the values of hallway options $\mathcal{H}$ using only primitive actions $\mathcal{A}$. As mentioned earlier, there are no baseline intra-option average-reward algorithms, so this is a proof-of-concept experiment.

The rewarding state for this experiment was `G2`, which can be reached using the option that leads to the lower hallway. The optimal reward rate, in this case, is $1/15 \approx 0.666$ with both $\mathcal{O} = \mathcal{H}$ and $\mathcal{O} = \mathcal{A}$. We applied intra-option Differential Q-learning using a behavior policy that chose the four primitive actions with equal probability in all states. This choice of behavior policy and goal `G2` would test if the intra-option algorithm leads to a policy consisting exclusively of options by interacting with the world using only primitive actions. Each parameter setting was run for $500,000$ steps and repeated 30 times. For evaluation, for every 2000 training steps, we report the reward rate averaged over 2000 evaluation steps. During these evaluation steps, I ran the agent's greedy policy w.r.t. its current estimated option-value function and did not modify its parameters.

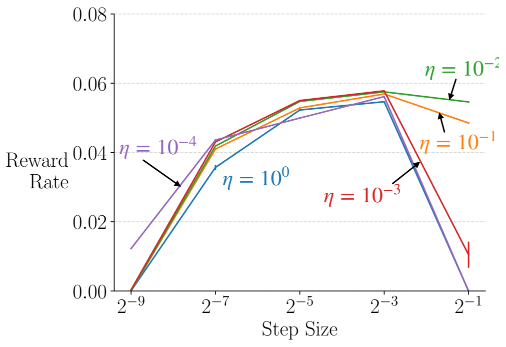
Figure 4.7 shows the learning curve of this average reward across the 30 independent runs for parameters $\alpha = 0.125, \eta = 0.1$. The agent indeed succeeds in learning the option-value function corresponding to the hallway options using a behavior policy consisting only of primitive actions. The parameter study of intra-option Differential Q-learning is presented in the right panel of the same figure. It can be seen that the algorithm is not sensitive to η and works well for a wide range of α.

I performed yet another intra-option experiment in the same domain and experiment setup with the one mentioned in Section 4.6 to demonstrate the differences between inter- and intra-option Differential Q-learning algorithms. The parameter study of intra-option is shown in Figure 4.8. Comparing Figure 4.8 with the left subfigures of Figure 4.5, several observations can be drawn. First, there is a narrower range of α that the intra-option algorithm can achieve a high reward rate rapidly. Second, the intra-option algorithm's reward rate and relative value errors are less sensitive to η but are more sensitive to α. Third, the intra-option algorithm generally achieved a lower relative value error.

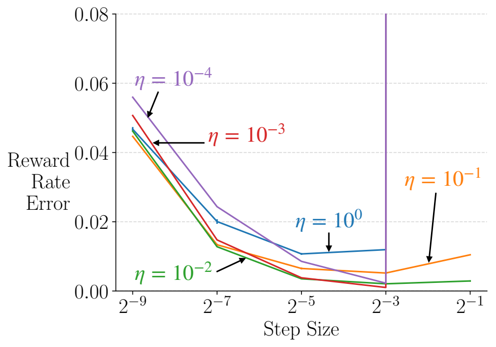
In conclusion, experiments in this section showed that intra-option prediction and control algorithms both converge to the desired points asymptotically, verifying Theorem 4.8 and Theorem 4.6. In addition, the value estimates in both prediction and control intra-option algorithms converged faster. Third, the intra-option control algorithm is more sensitive to the choice of α, but is less sensitive to the choice of η.

4.9 Intra-Option Model Learning and Planning Algorithms

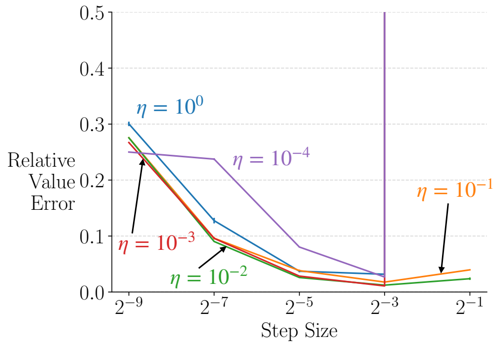
This section introduces methods to obtain option models. Recall that Section 4.5 introduced inter-option planning algorithms, which work with option models, but did not discuss how to obtain such models. This section introduces an algorithm to learn such models in an intra-option fashion. This option-model learning algorithm can be combined with inter-option planning algorithms to obtain a complete model-based average-reward options algorithm that learns option models and plans with them (see Algorithm 12 for the pseudo-code of this combined algorithm). Finally, this section also includes



(a) This plot illustrates the parameter sensitivity curves concerning the reward rate achieved by the learned policy in relation to parameters α and η.



(b) This plot illustrates the parameter sensitivity curves concerning the reward rate error in relation to parameters α and η.



(c) This plot illustrates the parameter sensitivity curves concerning the relative value error in relation to parameters α and η.

Figure 4.8: Parameter sensitivity curves of the intra-option Q-learning algorithm.

the planning version of the model-learning algorithm. This algorithm plans with an action model to obtain an option model.

The average-reward option model is similar to the discounted options model but with key distinctions. Sutton et al.'s (1999) discounted option model has two parts: the dynamics part and the reward part. Given a state and an option, the dynamics part predicts the discounted occupancy of states upon termination, and the reward part predicts the expected (discounted) sum of rewards during the execution of the option. In the average-reward setting, apart from the dynamics and the reward parts, an option model has a third part — the *duration* part — that predicts the duration of execution of the option. In addition, the dynamics part predicts the state distribution upon termination without discounting, and reward part predicts the undiscounted cumulative rewards during the execution of the option.

Formally, the dynamics part approximates $m^p(s' \mid s, o) \doteq \hat{p}(s' \mid s, o)$, the probability that option o terminates in state s' when starting from state s. The reward part approximates $m^r(s, o) \doteq \hat{r}(s, o)$, the expected cumulative reward during the execution of option o when starting from state s. Finally, the duration part approximates $m^l(s, o) \doteq \hat{l}(s, o)$, the expected duration of option o when starting from state s.

We now present a set of recursive equations that are key to our model-learning algorithms. These equations extend the discounted Bellman equations for option models (Sutton et al. 1999) to the average-reward formulation.

$$\bar{m}^p(x \mid s, o) = \sum_a \pi(a \mid s, o) \sum_{s', r} p(s', r \mid s, a) \Big(\beta(s', o)\mathbf{I}\{x = s'\} + \big(1 - \beta(s', o)\big)\bar{m}^p(x \mid s', o)\Big), \tag{4.53}$$

$$\bar{m}^r(s, o) = \sum_a \pi(a \mid s, o) \sum_{s', r} p(s', r \mid s, a)\big(r + (1 - \beta(s', o))\bar{m}^r(s', o)\big), \tag{4.54}$$

$$\bar{m}^l(s, o) = \sum_a \pi(a \mid s, o) \sum_{s', r} p(s', r \mid s, a)\big(1 + (1 - \beta(s', o))\bar{m}^l(s', o)\big). \tag{4.55}$$

The first equation is different from the other two because the total reward and length of the option o are incremented irrespective of whether the option terminates in s' or not. The following theorem shows that (m^p, m^r, m^l) is the unique solution of (4.53–4.55) and therefore the models can be obtained by solving these equations.

Theorem 4.10. *There exist unique* $\bar{m}^p \in \mathbb{R}^{|\mathcal{S}|\times|\mathcal{O}|\times|\mathcal{S}|}$, $\bar{m}^r \in \mathbb{R}^{|\mathcal{S}|\times|\mathcal{O}|}$, *and* $\bar{m}^l \in \mathbb{R}^{|\mathcal{S}|\times|\mathcal{O}|}$ *for which* (4.53), (4.54), *and* (4.55) *hold. Further, if* $\bar{m}^p, \bar{m}^r, \bar{m}^l$ *satisfy* (4.53), (4.54), *and* (4.55), *then* $\bar{m}^p = m^p, \bar{m}^r = m^r, \bar{m}^l = m^l$.

Proof. We will show that there exists a unique solution for (4.53). Results for (4.54) and (4.55) can be shown similarly. To show that (4.53) has a unique solution, we apply a generalized version of the Banach fixed point theorem (see, e.g., Theorem 2.4 by Almezel, Ansari, and Khamsi 2014). Once the unique existence of the solution is shown, we easily verify that m^p is the unique solution by showing that it is one solution to (4.53) as follows. With a little abuse of notation, let $\hat{p}(s', l \mid s, o) \doteq \sum_r \hat{p}(x, r, l \mid s, o)$, we have

$$
\begin{aligned}
& m^p(x \mid s, o) \\
&= \sum_{l=1}^{\infty} \hat{p}(x, l \mid s, o) \\
&= \sum_a \pi(a \mid s, o) \sum_r p(s', r \mid s, a)\beta(s', o)\mathbf{I}\{x = s'\} + \sum_{l=2}^{\infty} \hat{p}(x, l \mid s, o) \\
&= \sum_a \pi(a \mid s, o) \sum_r p(s', r \mid s, a) \\
&\left(\beta(s', o)\mathbf{I}\{x = s'\} + (1 - \beta(s', o)) \sum_{l=1}^{\infty} \hat{p}(x, l \mid s', o)\right) \\
&= \sum_a \pi(a \mid s, o) \sum_r p(s', r \mid s, a) \\
&(\beta(s', o)\mathbf{I}\{x = s'\} + (1 - \beta(s', o))m^p(x \mid s', o)) .
\end{aligned}
$$

To apply the generalized version of the Banach fixed point theorem to show the unique existence of the solution, we first define operator $T : \mathbb{R}^{|\mathcal{S}|\times|\mathcal{S}|\times|\mathcal{O}|} \to \mathbb{R}^{|\mathcal{S}|\times|\mathcal{S}|\times|\mathcal{O}|}$ such that for any $m \in \mathbb{R}^{|\mathcal{S}|\times|\mathcal{S}|\times|\mathcal{O}|}$ and any $x, s \in \mathcal{S}, o \in \mathcal{O}$, $Tm(x \mid s, o) \doteq \sum_a \pi(a \mid s, o) \sum_{s', r} p(s', r \mid s, a)(\beta(s', o)\mathbf{I}\{x = s'\} + (1 - \beta(s', o))m(x \mid s', o)))$. We further define $T^n m \doteq T(T^{n-1}m)$ for any $n \geq 2$ and any $m \in \mathbb{R}^{|\mathcal{S}|\times|\mathcal{S}|\times|\mathcal{O}|}$. The generalized Banach fixed point theorem shows that if T^n is a contraction mapping for any integer $n \geq 1$ (this is called a n-stage contraction), then $Tm = m$ has a unique fixed point.

The only work left is to verify the following contraction property:

$$\left\| T^{|\mathcal{S}|} m_1 - T^{|\mathcal{S}|} m_2 \right\|_\infty \leq \gamma \| m_1 - m_2 \|_\infty, \tag{4.56}$$

where m_1 and m_2 are arbitrary members in $\mathbb{R}^{|\mathcal{S}| \times |\mathcal{S}| \times |\mathcal{O}|}$, and $\gamma < 1$ is some constant.

Consider the difference between $T^{|\mathcal{S}|} m_1$ and $T^{|\mathcal{S}|} m_2$ for arbitrary $m_1, m_2 \in \mathbb{R}^{|\mathcal{S} \times \mathcal{S} \times \mathcal{O}|}$. For any $x, s \in \mathcal{S}, o \in \mathcal{O}$, we have

$$\begin{aligned}
&T^{|\mathcal{S}|} m_1(x \mid s, o) - T^{|\mathcal{S}|} m_2(x \mid s, o) \\
&= \sum_a \pi(a \mid s, o) \sum_{s', r} p(s', r \mid s, a)(1 - \beta(s', o)) \\
&(T^{|\mathcal{S}|-1} m_1(x \mid s', o) - T^{|\mathcal{S}|-1} m_2(x \mid s', o)) \\
&= \sum_{s_1} \Pr(S_{t+1} = s_1 \mid S_t = s, O_t = o)(1 - \beta(s_1, o)) \\
&(T^{|\mathcal{S}|-1} m_1(x \mid s_1, o) - T^{|\mathcal{S}|-1} m_2(x \mid s_1, o)) \\
&= \sum_{s_1} \Pr(S_{t+1} = s_1 \mid S_t = s, O_t = o)(1 - \beta(s_1, o)) \\
&\sum_{s_2} \Pr(S_{t+2} = s_2 \mid S_{t+1} = s_1, O_{t+1} = o)(1 - \beta(s_2, o)) \\
&(T^{|\mathcal{S}|-2} m_1(x \mid s_2, o) - T^{|\mathcal{S}|-2} m_2(x \mid s_2, o)) \\
&\vdots \\
&= \sum_{s_1, \cdots, s_{|\mathcal{S}|}} \Pr\big(S_{t+1} = s_1, \cdots, S_{t+|\mathcal{S}|} = s_{|\mathcal{S}|} \mid S_t = s, O_{t:t+|\mathcal{S}|-1} = o\big) \\
&\prod_{i=1}^{|\mathcal{S}|} (1 - \beta(s_i, o))(m_1(x \mid s_{|\mathcal{S}|}, o) - m_2(x \mid s_{|\mathcal{S}|}, o)) \\
&\leq \sum_{s_1, \cdots, s_{|\mathcal{S}|}} \Pr\big(S_{t+1} = s_1, \cdots, S_{t+|\mathcal{S}|} = s_{|\mathcal{S}|} \mid S_t = s, O_{t:t+|\mathcal{S}|-1} = o\big) \\
&\prod_{i=1}^{|\mathcal{S}|} (1 - \beta(s_i, o)) \| m_1 - m_2 \|_\infty.
\end{aligned}$$

Here

$$\tilde{p}(s,o) \doteq \sum_{s_1,\cdots,s_{|\mathcal{S}|}} \Pr\Big(S_{t+1} = s_1 \cdots , S_{t+|\mathcal{S}|} = s_{|\mathcal{S}|} \mid S_t = s, O_{t:t+|\mathcal{S}|-1} = o\Big)$$
$$\prod_{i=1}^{|\mathcal{S}|}(1 - \beta(s_i, o))$$

is the probability of executing option o for $|\mathcal{S}|$ steps starting from s without termination. If $\tilde{p}(s,o) = 0$, then option o will surely terminate within the first $|\mathcal{S}|$ steps, and if $\tilde{p}(s,o) = 1$, then option o will surely not terminate within the first $|\mathcal{S}|$ steps.

If option o would surely not terminate within the first $|\mathcal{S}|$ steps ($\tilde{p}(s,o) = 1$), then it would surely not terminate forever. This is because there are only $|\mathcal{S}|$ states, and thus an option could visit all states that are possible to be visited by the option within the first $|\mathcal{S}|$ steps. $\tilde{p}(s,o) = 1$ means that option o has a zero probability of terminating in all states that are possible to be visited by option o. But this violates Assumption 4.1. Therefore, $\tilde{p}(s,o) < 1$ for all s, o. So there must exist some $\gamma(s,o) < 1$ such that $\tilde{p}(s,o) \leq \gamma(s,o)$. With $\gamma \doteq \max_{s,o} \gamma(s,o)$, we obtain (4.56). □

Let $S_t, A_t, R_{t+1}, S_{t+1}$ be generated the same way as in intra-option value learning algorithms introduced in Section 4.7. Our *intra-option model-learning* algorithm solves the above recursive equations using the following TD-like update rules for each option o:

$$M^p_{t+1}(x \mid S_t, o) \doteq M^p_t(x \mid S_t, o) + \alpha_{\nu_t(S_t,o)}\rho_t(o)\Big(\beta(S_{t+1}, o)\mathbf{I}\{S_{t+1} = x\} + \big(1 - \beta(S_{t+1}, o)\big)M^p_t(x \mid S_{t+1}, o) - M^p_t(x \mid S_t, o)\Big), \quad \forall\, x \in \mathcal{S}, \tag{4.57}$$

$$M^r_{t+1}(S_t, o) \doteq M^r_t(S_t, o) + \alpha_{\nu_t(S_t,o)}(S_t, o)\rho_t(o) \Big(R_{t+1} + \big(1 - \beta(S_{t+1}, o)\big)M^r_t(S_{t+1}, o) - M^r_t(S_t, o)\Big) \tag{4.58}$$

$$M^l_{t+1}(S_t, o) \doteq M^l_t(S_t, o) + \alpha_{\nu_t(S_t,o)}(S_t, o)\rho_t(o) \Big(1 + \big(1 - \beta(S_{t+1}, o)\big)M^l_t(S_{t+1}, o) - M^l_t(S_t, o)\Big), \tag{4.59}$$

where M^p is a $|\mathcal{S}| \times |\mathcal{O}| \times |\mathcal{S}|$-sized vector of estimates, M^r and M^l are both $|\mathcal{S}| \times |\mathcal{O}|$-sized vectors of estimates, $\{\alpha_n\}_{n \geq 1}$ is a step-size sequence, $\nu_t(S_t, o)$

is the number of times the pair (S_t, o) has been updated up to and including time step t, and $\rho_t(o) \doteq \pi(A_t \mid S_t, o)/b(A_t \mid H_t)$ is the importance sampling ratio.

Theorem 4.11. *If Assumption 3.5 holds for $\{\alpha_n\}$, all state-option pairs are visited an infinite number of times, and Assumption 4.2 holds, then the intra-option model-learning algorithm (4.57–4.59) converges almost surely, M_t^p to m^p, M_t^r to m^r, and M_t^l to m^l.*

Proof. Because the three update rules are independent, we only show the convergence of the first update rule; the other two can be shown in the same way.

We apply a slight generalization of Theorem 3 by Tsitsiklis (1994) to show the first update rule. The generalization replaces Assumption 5 (an assumption for Theorem 3) by:

Assumption 4.3. There exists a vector $x^* \in \mathbb{R}^n$, a positive vector v, a positive integer m, and a scalar $\beta \in [0, 1)$, such that

$$\|F^m(x) - x^*\|_v \leq \beta \|x - x^*\|_v, \quad \forall\, x \in \mathbb{R}^n.$$

That is, we replace the one-stage pseudo contraction assumption with a m-stage contraction assumption. The proof of Tsitsiklis' Theorem 3 also applies to its generalized form and is thus omitted here.

Notice that our update rule (4.57) is a special case of the general update rule considered by Theorem 3 (Equations 1-3), and is thus a special case of its generalized version. Therefore we only need to verify the above $m-$stage contraction assumption, as well as Assumptions 1, 2, and 3 required by Theorem 3. According to the proof of Theorem 4.10, the operator T associated with the update rule (4.57) is a $|\mathcal{S}|-$stage contraction (and thus is a $|\mathcal{S}|-$stage pseudo-contraction). Other assumptions (Assumptions 1, 2, 3) required by Theorem 3 are also satisfied given our step-size sequence, finite MDP assumptions as well as Assumption 4.2. $\square$

The planning version of the intra-option model-learning algorithm, just like this learning algorithm, maintains a table of dynamics estimates M_n^p :

$|\mathcal{S}| \times |\mathcal{O}| \times |\mathcal{S}| \to \mathbb{R}$, a table of reward estimates $M_n^r : |\mathcal{S}| \times |\mathcal{O}| \to \mathbb{R}$ and a table of duration estimates $M_n^l : |\mathcal{S}| \times |\mathcal{O}| \to \mathbb{R}$. At each planning step n, these estimates are updated by (4.57)–(4.59), just as in intra-option model learning, except now using the simulated experience $\ldots, \tilde{S}_n, \tilde{O}_n, \tilde{A}_n, \tilde{R}_n, \tilde{S}'_n, \ldots$ (see Section 4.7 for the generation of the simulated experience) instead of the real experience $\ldots, S_t, O_t, A_t, R_{t+1}, S_{t+1}, \ldots$.

Theorem 4.12. *If Assumption 3.5 holds for $\{\alpha_n\}$, all state-action pairs are visited an infinite number of times, and Assumption 4.2 holds, then the intra-option model-planning algorithm converges almost surely, M_n^p to m^p, M_n^r to m^r, and M_n^l to m^l.*

The proof is essentially the same as that of Theorem 4.11 and is therefore omitted.

4.10 Interruption to Improve the Hierarchical Policy

In all of the algorithms we considered so far, the hierarchical policy would select an option, execute the option's policy till termination, then select a new option. Sutton et al. (1999) showed that the hierarchical policy can be improved by allowing the *interruption* of an option midway through its execution to start a seemingly better option.

We now show that this interruption result applies to average-reward options as well.

Theorem 4.13. *For any MDP, any set of options $\mathcal{O}$, and any hierarchical unichain policy $\mu : \mathcal{S} \to \Delta(\mathcal{O})$, define a new set of options, $\mathcal{O}'$, with a one-to-one mapping between the two option sets as follows: for every $o = (\pi, \beta) \in \mathcal{O}$, define a corresponding $o' = (\pi, \beta') \in \mathcal{O}'$ where $\beta' = \beta$, but for any state s in which $\hat{q}_\mu(s, o) < \hat{v}_\mu(s)$, $\beta'(s, o) = 1$ (where $\hat{v}_\mu(s) \doteq \sum_o \mu(o \mid s)\hat{q}_\mu(s, o)$). Let the interrupted policy μ' be such that for all $s \in \mathcal{S}$ and for all $o' \in \mathcal{O}', \mu'(s, o') = \mu(s, o)$, where o is the option in $\mathcal{O}$ corresponding to o'. Assume that μ' is unichain. Then:*

1. *the new policy over options μ' is not worse than the old one μ, i.e., $\hat{r}(\mu') \geq \hat{r}(\mu)$,*
2. *if there exists a state $s \in \mathcal{S}$ from which there is a non-zero probability of encountering an interruption upon initiating μ' in s, then $\hat{r}(\mu') > \hat{r}(\mu)$.*

In short, the above theorem shows that interruption produces a behavior that achieves a higher reward rate than without interruption. Note that interruption behavior is only applicable to intra-option algorithms; complete option transitions are needed in inter-option algorithms.

Proof. We first show that

$$\begin{aligned}&\sum_{o'} \mu'(o' \mid s) \sum_{s',r,l} \hat{p}(s',r,l \mid s,o')(r - l\hat{r}(\mu) + \hat{v}_\mu(s')) \\ &\geq \sum_{o} \mu(o \mid s) \sum_{s',r,l} \hat{p}(s',r,l \mid s,o)(r - l\hat{r}(\mu) + \hat{v}_\mu(s')) = \hat{v}_\mu(s). \qquad (4.60)\end{aligned}$$

Note that for all s, o and its corresponding o', $\mu(o \mid s) = \mu'(o' \mid s)$. In order to show (4.60), we show $\sum_{s',r,l} \hat{p}(s',r,l \mid s,o')(r - l\hat{r}(\mu) + \hat{v}_\mu(s')) \geq \sum_{s',r,l} \hat{p}(s',r,l \mid s,o)(r - l\hat{r}(\mu) + \hat{v}_\mu(s'))$ for all s, o and corresponding o'.

$$\begin{aligned}&\sum_{s',r,l} \hat{p}(s',r,l \mid s,o')(r - l\hat{r}(\mu) + \hat{v}_\mu(s')) \\ &= \mathbb{E}[\hat{R}_n - \hat{L}_n\hat{r}(\mu) + \hat{v}_\mu(\hat{S}_{n+1}) \mid S_n = s, O_n = o'] \\ &= \mathbb{E}[\hat{R}_n - \hat{L}_n\hat{r}(\mu) + \hat{v}_\mu(\hat{S}_{n+1}) \mid S_n = s, O_n = o', \text{Not encountering an interruption}] \\ &+ \mathbb{E}[\hat{R}_n - \hat{L}_n\hat{r}(\mu) + \hat{v}_\mu(\hat{S}_{n+1}) \mid S_n = s, O_n = o', \text{Encountering an interruption}] \\ &\geq \mathbb{E}[\hat{R}_n - \hat{L}_n\hat{r}(\mu) + \hat{v}_\mu(\hat{S}_{n+1}) \mid S_n = s, O_n = o', \text{Not encountering an interruption}] \\ &+ \mathbb{E}[\beta(s')(\hat{R}_n - \hat{L}_n\hat{r}(\mu) + \hat{v}_\mu(\hat{S}_{n+1})) + (1-\beta(s'))(\hat{R}_n - \hat{L}_n\hat{r}(\mu) + \hat{q}_\mu(\hat{S}_{n+1}, o)) \\ &\mid S_n = s, O_n = o', \text{Encountering an interruption}] \\ &= \sum_{s',r,l} \hat{p}(s',r,l \mid s,o)(r - l\hat{r}(\mu) + \hat{v}_\mu(s')).\end{aligned}$$

The above inequality holds because $\hat{S}_{n+1}$ is the state where termination happens and thus $\hat{q}_\mu(\hat{S}_{n+1}, o) \leq \hat{v}_\mu(\hat{S}_{n+1})$. The last equality holds because $\mathbb{E}[\beta(s')(\hat{R}_n - \hat{L}_n\hat{r}(\mu) + \hat{v}_\mu(\hat{S}_{n+1})) + (1-\beta(s'))(\hat{R}_n - \hat{L}_n\hat{r}(\mu) + \hat{q}_\mu(\hat{S}_{n+1}, o)) \mid S_n = s, O_n = o', \text{Encountering an interruption}]$ is the expected differential return when the agent could interrupt its old option but chooses to stick on the old option. (4.60) is shown.

Now write the l.h.s. of (4.60) in the matrix form

$$
\begin{aligned}
&\sum_{o'} \mu'(o' \mid s) \sum_{s',r,l} \hat{p}(s', r, l \mid s, o')(r - l\hat{r}(\mu) + \hat{v}_\mu(s')) \\
&= r_{\mu'}(s) - l_{\mu'}(s)\hat{r}(\mu) + (P_{\mu'}\hat{v}_\mu)(s),
\end{aligned}
$$

where $r_{\mu'}(s) \doteq \sum_{o'} \mu'(o' \mid s) \sum_{s',r,l} \hat{p}(s', r, l \mid s, o')r$ is the expected one option-transition reward, $l_{\mu'}(s) \doteq \sum_{o'} \mu'(o' \mid s) \sum_{s',r,l} \hat{p}(s', r, l \mid s, o')l$ is the expected one option-transition length, and $P_{\mu'}(s, s') \doteq \sum_{o'} \mu'(o' \mid s) \sum_{r,l} \hat{p}(s', r, l \mid s, o')$ is the probability of terminating at s'.

Combined with the r.h.s. of (4.60), we have

$$
r_{\mu'}(s) - l_{\mu'}(s)\hat{r}(\mu) + (P_{\mu'}\hat{v}_\mu)(s) \geq \hat{v}_\mu(s).
$$

Iterating the above inequality for $K - 1$ times, we have

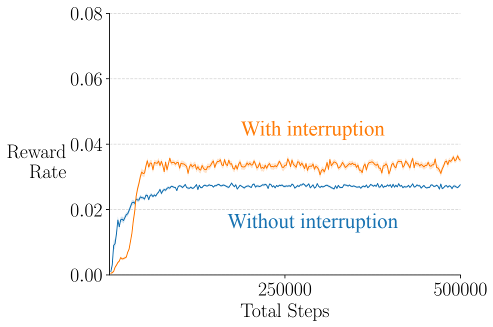
$$
\begin{aligned}
&\sum_{k=0}^{K-1} (P_{\mu'}^k r_{\mu'}(s) - P_{\mu'}^k l_{\mu'}(s)\hat{r}(\mu)) + P_{\mu'}^K \hat{v}_\mu(s) \geq \hat{v}_\mu(s) \\
&\implies \\
&\sum_{k=0}^{K-1} (P_{\mu'}^k r_{\mu'}(s) - P_{\mu'}^k l_{\mu'}(s)\hat{r}(\mu)) \geq \hat{v}_\mu(s) - P_{\mu'}^K \hat{v}_\mu(s).
\end{aligned}
$$

Divide both sides by $\sum_{k=0}^{K-1} P_{\mu'}^k l_{\mu'}(s)$ and take $K \to \infty$:

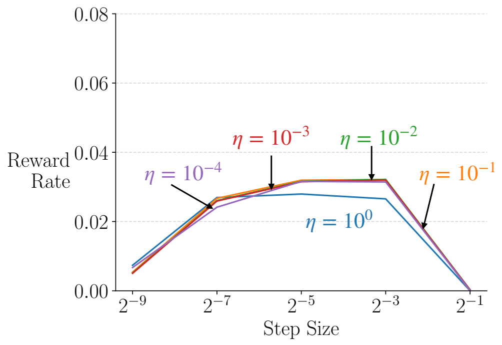
$$
\begin{aligned}
&\lim_{K\to\infty} \frac{1}{\sum_{k=0}^{K-1} P_{\mu'}^k l_{\mu'}(s)} \sum_{k=0}^{K-1} (P_{\mu'}^k r_{\mu'}(s) - P_{\mu'}^k l_{\mu'}(s)\hat{r}(\mu)) \\
&\geq \lim_{K\to\infty} \frac{1}{\sum_{k=0}^{K-1} P_{\mu'}^k l_{\mu'}(s)} (\hat{v}_\mu(s) - P_{\mu'}^K \hat{v}_\mu(s)).
\end{aligned}
$$

For the l.h.s.:

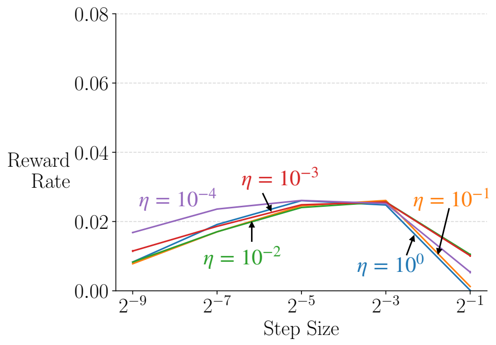
$$
\begin{aligned}
&\lim_{K\to\infty} \frac{1}{\sum_{k=0}^{K-1} P_{\mu'}^k l_{\mu'}(s)} \sum_{k=0}^{K-1} (P_{\mu'}^k r_{\mu'}(s) - P_{\mu'}^k l_{\mu'}(s)\hat{r}(\mu))) \\
&= \lim_{K\to\infty} \frac{\sum_{k=0}^{K-1} P_{\mu'}^k r_{\mu'}(s)}{\sum_{k=0}^{K-1} P_{\mu'}^k l_{\mu'}(s)} - \hat{r}(\mu) \\
&= \hat{r}(\mu') - \hat{r}(\mu),
\end{aligned}
$$



(a) Learning curves with and without interruption.



(b) Parameter sensitivity curves with interruption



(c) Parameter sensitivity curves without interruption

Figure 4.9: Plots showing that executing options with interruptions can achieve a higher reward rate than executing options till termination in the domain described in the adjoining text.

where the last equation holds because μ' is unichain.

For the r.h.s.:

$$\lim_{K\to\infty} \frac{1}{\sum_{k=0}^{K-1} P_{\mu'}^{k} l_{\mu'}(s)} (\hat{v}_{\mu}(s) - P_{\mu'}^{K} \hat{v}_{\mu}(s)) = 0.$$

Therefore $\hat{r}(\mu') - \hat{r}(\mu) \geq 0$.

Finally, note that a strict inequality holds if the probability of interruption when following policy μ' is non-zero. □

4.11 Interruption Experiments

This section tests the intra-option Differential Q-learning algorithm with and without interruption in the four-room domain. The goal is to empirically verify that behaving with interruption can indeed achieve a higher reward rate.

We set the goal as `G3` and allowed the agent to choose and learn only from the set of all hallway options $\mathcal{H}$. With just hallway options, without interruption, the best strategy is to first move to the lower hallway and then try to reach the goal by following options that pick random actions in the states near the hallway and goal. With interruption, the agent can first move to the left hallway, then take the option that moves the agent to the lower hallway but terminate when other options have higher option values. This termination is most likely to occur in the cell just above `G3`. The agent then needs fewer steps in expectation to reach the goal.

The agent followed a random behavior policy to learn option values. For every 2000 training step, there is a testing phase. At the beginning of the testing phase, the agent was reset to the start state. The agent then followed a greedy hierarchical policy w.r.t. learned option values for 2000 steps, either with interruption or without interruption. The average reward rate over the 2000 testing steps is recorded.

Figure 4.9a shows learning curves using two tested algorithms on this problem. Each point is the average-reward rate over 2000 evaluation steps. The parameter being chosen is the one that resulted in the highest reward rate over the last 10000 evaluation steps. As expected, the agent achieved a higher reward rate by using interruptions, verifying Theorem 4.13. The parameter studies of two tested algorithms are shown in Figure 4.9b and Figure 4.9c respectively. The axes are the same as those mentioned in Figure 4.3. The interruption algorithm achieved a higher reward rate for intermediate α and was also not sensitive to η.

4.12 Summary

This chapter introduced a family of average-reward options algorithms and their associated theories.

For both learning and planning, prediction and control problems, this chapter introduced inter- and intra-option algorithms. Among these algorithms, the inter-option planning algorithms need special attention because they plan with option models and are thus potentially much more computationally efficient than planning algorithms that operate with action models. Recall that one of the most important ways that the agent can benefit from using options is to plan faster with option models. To obtain option models, this chapter introduced learning and planning algorithms, both operating in the intra-option fashion. The intra-option algorithms also include learning or planning algorithms to obtain option values. The intra-option learning algorithms use action transitions rather than option transitions, and thus can potentially be more sample efficient than inter-option algorithms, which do not look inside the execution of options.

Note that, while both classes of algorithms can be viewed as extensions of the algorithms introduced in the previous chapter from actions to options, such extensions do not trivially replace actions with options. For example, as shown in Section 4.5, there exist several more straightforward, yet inappropriate ways to extend Differential Q-learning to an inter-option algorithm. For another example, unlike the algorithms introduced in the previous chapter, which only update for one action at each step, the intra-option algorithms introduced in this chapter update for all options at each step.

The theories developed in this chapter are largely based on the convergence theory of the General RVI Q algorithm (Section 3.2) and a new theory concerning SMDPs developed in this chapter (Section 4.3). This new theory characterizes the intersection of the solution set of the SMDP optimality equation and the solution set of a special equation. This special equation involves a class of functions. By choosing a specific function, this intersection reduces to a set to which one of the algorithms introduced in the current and the previous chapters converges.

This chapter also developed a new sub-optimality bound for weakly-communicating SMDPs. This result is independent of the proposed options algorithms. This bound extends Theorem 8.5.5 by Puterman (1994) from unichain MDPs to weakly communicating SMDPs. This result shows a lower bound of the performance of greedy policies of any estimate of the action-value function.

Algorithm 12: Combined algorithm: intra-option model learning + inter-option Differential Q-planning

```
Input: Behavioral policy $b$
Algorithm parameters: step-size sequences $\alpha_n, \beta_n$, parameter $\eta$,
                      number of planning steps per time step $n$
1  Initialize $Q(s,o), P(x \mid s,o), R(s,o)\ \forall\, s,x \in \mathcal{S}, o \in \mathcal{O}, \bar{R}$, arbitrarily
     (e.g., to zero)
2  Initialize $L(s,o)\ \forall\, s \in \mathcal{S}, o \in \mathcal{O}$ to be any value greater than 0
3  $\nu_1(s,o) \leftarrow 0\ \forall s,o$, $\nu_2(s,o) \leftarrow 0\ \forall s,o$
4  $H \leftarrow S$
5  while still time to train do
6      $S \leftarrow$ current state
7      Sample primitive action $A \sim b(\cdot \mid H)$
8      Take action $A$, observe $R', S'$
9      for all options $o$ such that $\pi(A \mid S,o) > 0$ do
10         $\nu_2(S,o) \leftarrow \nu_2(S,o) + 1$
11         $\rho \leftarrow \pi(A \mid S,o)/b(A \mid H)$
12         for all states $x \in \mathcal{S}$ do
13             $P(x \mid S,o) \leftarrow P(x \mid S,o) + \beta_{\nu_2(S,o)}\rho\Big(\beta(S',o)\mathbb{I}(S' =$
                 $x) + \big(1-\beta(S',o)\big)P(x \mid S',o) - P(x \mid S,o)\Big)$
14         end
15         $R(S,o) \leftarrow R(S,o)+\beta_{\nu_2(S,o)}\rho\Big(R'+\big(1-\beta(S',o)\big)R(S',o)-R(S,o)\Big)$
16         $L(S,o) \leftarrow L(S,o)+\beta_{\nu_2(S,o)}\rho\Big(1+\big(1-\beta(S',o)\big)L(S',o)-L(S,o)\Big)$
17     end
18     $H \leftarrow (H,A,S',R')$ for all of the $n$ planning steps do
19         $S \leftarrow$ a random previously observed state
20         $O \leftarrow$ a random option previously taken in $S$
21         $\nu_1(S,O) \leftarrow \nu_1(S,O) + 1$
22         $S' \leftarrow$ a sampled state from $P(\cdot \mid S,O)$
23         $\delta \leftarrow R(S,O) - L(S,O)\bar{R} + \max_o Q(S',o) - Q(S,O)$
24         $Q(S,O) \leftarrow Q(S,O) + \alpha_{\nu_1(S,O)}\rho\delta/L(S,O)$
25         $\bar{R} \leftarrow \bar{R} + \eta\alpha_{\nu_1(S,O)}\rho\delta/L(S,O)$
26     end
27 end
28 return $Q$
```

Algorithm 13: Intra-option Differential Q-learning with interruption

```
   Input: Behavioral policy $b$
   Algorithm parameters: step-size parameters $\alpha, \eta$
1  Initialize $Q(s,o)\ \forall\, s \in \mathcal{S}, o \in \mathcal{O}, \bar{R}$ arbitrarily (e.g., to zero)
2  Obtain initial $S$
3  $\nu(s,o) \leftarrow 0\ \forall s, o$
4  $O \leftarrow$ option sampled from $b(\cdot \mid S)$
5  while still time to train do
6      if $O \notin \operatorname{argmax} Q(S,\cdot)$ then
7          $O \leftarrow$ option sampled from $b(\cdot \mid S)$
8      end
9      Sample primitive action $A \sim \pi(\cdot \mid S, O)$
10     Take action $A$, observe $R, S'$
11     $\Delta \leftarrow 0$
12     for all options $o$ do
13         $\nu(S,o) \leftarrow \nu(S,o) + 1$
14         $\rho \leftarrow \pi(A \mid S,o)/\pi(A \mid S,O)$
15         $\delta \leftarrow R - \bar{R} + \Big(\big(1 - \beta(S',o)\big)Q(S',o) +$
             $\beta(S',o)\max_{o'} Q(S',o')\Big) - Q(S,o)$
16         $Q(S,o) \leftarrow Q(S,o) + \alpha\rho\delta$
17         $\Delta \leftarrow \Delta + \eta\alpha\rho\delta$
18     end
19     $\bar{R} \leftarrow \bar{R} + \Delta$
20     $S \leftarrow S'$
21 end
22 return $Q$
```

Chapter 5

Prediction with Function Approximation

This chapter presents the third area of contribution: an average-reward off-policy prediction learning algorithm with function approximation, its convergence theory, and an error bound for the convergent point.

The previous two chapters present average-reward learning and planning algorithms using a tabular representation. In many real-world problems, the state and/or the action spaces can be very large. It is not possible to represent the action-value function using a table whose size is the order of the number of state-action pairs. For these problems, *function approximation* must be used to represent any object, including the action-value function, that is in order of the number of state-action pairs or more.

This chapter first demonstrates that a direct function approximation extension of the tabular algorithm (Differential TD-learning) introduced in Section 3.3 diverges. It then introduces a new algorithm and its associated convergence proof. Just like all the prediction and control algorithms introduced in the previous two chapters, this new algorithm, called Differential GQ1, also uses the TD error to estimate the reward rate and is therefore, again, a differential method. In addition to the convergence theory of the new algorithm, this chapter further shows an asymptotic error bound of the point that the algorithm converges to due to function approximation. Finally, this chapter uses an experiment to demonstrate the empirical performance of Differential GQ1.

5.1 Problem Setup

This chapter considers the off-policy prediction problem: evaluating a policy using data generated by following another policy. The target policy $\pi \in \Pi$, which is the policy being evaluated, is assumed to be unichain, just as in Section 3.1.

Assumption 5.1. The policy π is unichain in the MDP.

The goal of the off-policy prediction problem is to approximate the reward rate and the differential action-value function of π up to a constant. The differential action-value function q_π can be defined given the differential value function v_π: $q_\pi(s, a) \doteq \sum_{s', r} p(s', r \mid s, a)(r + v_\pi(s'))$.

Unlike Section 3.3, which studies the tabular setting, this chapter considers the linear function approximation (LFA) setting. In this setting, the agent can still observe states and actions, but instead of maintaining tables of size in the order of $|\mathcal{S}|$ or $|\mathcal{A}|$, it typically maintains weight vectors whose sizes are much smaller than $|\mathcal{S}|$ or $|\mathcal{A}|$. An LFA algorithm approximates any function $f : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ that takes as input a state-action pair and produces a real vector (e.g., the action-value function), using a *feature vector* of the pair. Specifically, given a state-action pair (s, a), a feature vector $x(s, a)$ for (s, a) is a real vector of length d, where $x : \mathcal{S} \times \mathcal{A} \to \mathbb{R}^d$. The algorithm then uses $x(s, a)^\top w$ to approximate $f(s, a)$. For the uniqueness of the solution of w, it is typically assumed that

Assumption 5.2. X has linearly independent columns, where $X \in \mathbb{R}^{|\mathcal{S}||\mathcal{A}| \times d}$ is the feature matrix whose (s, a) row is $x(s, a)^\top$.

The LFA setting is mainly for the situation when $|\mathcal{S}|$ or $|\mathcal{A}|$ is large or infinite. But even if $|\mathcal{S}|$ and $|\mathcal{A}|$ are both small, LFA algorithms are still typically faster than their tabular counterparts because of generalization across states and actions. However, LFA algorithms typically achieve a worse asymptotic performance because of the limited representation power of the LFA.

The problem studied in this chapter is also different from the problem studied in Section 3.3 in the way data is generated. In this chapter, the agent learns from i.i.d. transitions sampled from a given unknown sampling distribution, whereas in Chapter 3, the agent learns from data generated by following a known behavior policy. Concretely, at each time step, a sample transition $(S_t, A_t, R_t, S'_t, A'_t)$ from a given sampling distribution $d_{\mu\pi}$, which satisfies the following assumption.

Assumption 5.3. $S_t, A_t \sim d_\mu$, where $d_\mu \in \Delta(\mathcal{S} \times \mathcal{A})$ satisfying $d_\mu(s, a) > 0$ for all (s, a) is some probability distribution over state-action pairs, $S'_t, R_t \sim p(\cdot, \cdot \mid S_t, A_t)$, and $A'_t \sim \pi(\cdot \mid S'_t)$.

Assumption 5.3 requires that $d_\mu > 0$, which just means that every state-action pair is possible to be sampled. This is a necessary condition for learning the differential action-value function accurately for all state-action pairs. However, if one's goal is to only approximate the reward rate, then it is not needed to sample all state-action pairs infinitely often. Instead, reasonable algorithms should only require pairs that are visited infinitely often when following π. In this chapter, this weaker assumption is not considered because my goal includes approximating the action-value function well.

In many cases, the sampling distribution $d_{\mu\pi}$ is not known by the agent. Instead, a large batch of transitions, collected by one or multiple agents, with all agents following possibly different unknown policies in the same MDP, is presented to the agent. One could then draw samples in the following way. First randomly sample (S_t, A_t, R_t, S'_t) from the batch. Then sample $A'_t \sim \pi(\cdot \mid S'_t)$. Assuming that the size of the batch is large enough, then sampling from the batch is approximately equivalent to sampling from some distribution satisfying Assumption 5.3. Of course, as required by Assumption 5.3, the batch of data should cover the state-action space well in order to guarantee that the action-value function can be accurately estimated.

Recall that a common way to obtain the reward rate and the differential action-value function (up to an additive constant) is to solve the action-value evaluation equation, defined by

$$q(s, a) = \sum_{s', r} p(s', r \mid s, a) \left(r - \bar{r} + \sum_{a'} \pi(a' \mid s') q(s', a') \right), \ \forall s \in \mathcal{S}. \tag{5.1}$$

5.2 Differential Semi-Gradient Q-Evaluation

This section presents *Differential Semi-gradient Q-Evaluation* (Differential SGQ), which is a straightforward linear function approximation extension of the Differential TD-learning algorithm (Section 3.3). This straightforward FA extension of the tabular algorithm, however, may diverge as shown in an example presented in this section. This divergence issue motivates searching for a proven-convergent algorithm for the FA setting. The next section presents such an algorithm.

Differential SGQ updates w_t and $\bar{r}_t$ by

$$w_{t+1} \doteq w_t + \alpha_t \delta_t(w_t, \bar{R}_t) x_t, \tag{5.2}$$

$$\bar{r}_{t+1} \doteq \bar{r}_t + \eta \alpha_t \delta_t(w_t, \bar{R}_t), \tag{5.3}$$

where α_t is the step size used at t time step, $x_t \doteq x(S_t, A_t)$, $x'_t \doteq x(S'_t, A'_t)$, $\eta > 0$ is some constant, and $\delta_t(w, \bar{R}) \doteq R_t - \bar{R} + x_t'^\top w - x_t^\top w$ is the TD error.

```
Algorithm 14: Differential SGQ
Input: A sampling distribution $d_{\mu\pi}$ and a feature mapping
       $x : \mathcal{S} \times \mathcal{A} \to \mathbb{R}^K$,
Algorithm parameters: A step-size sequence $\alpha_t$ and a scaling
                      parameter $\eta$
1 Initialize $w \in \mathbb{R}^K$ and $\bar{R} \in \mathbb{R}$ arbitrarily (e.g., to zero)
2 for $t = 0, 1, 2, \ldots$ do
3     Sample a transition $(S, A, R, S', A')$ from $d_{\mu\pi}$
4     $\phi = x(S, A)$, $\phi' = x(S', A')$
5     $\delta \doteq R - \hat{r} + \phi'^\top w - \phi^\top w$
6     $w \leftarrow w + \alpha_t \delta \phi$
7     $\bar{R} \leftarrow \bar{R} + \alpha_t \delta$
8 end
9 return the estimated reward rate $\bar{R}$, differential action-value function
    $Xw$, where $X$ is the feature matrix with $X[(s, a), i] = x(s, a)[i]$ for all
    $s \in \mathcal{S}, a \in \mathcal{A}$ and $i = 0, 1, \ldots, d - 1$
```

Diff-SGQ iteratively solves

$$\mathbb{E}[\delta_t(w, \bar{R}) x_t] = \mathbf{0} \tag{5.4}$$

$$\mathbb{E}[\delta_t(w, \bar{R})] = 0, \tag{5.5}$$

whose solutions, if they exist, are *TD fixed points.* A TD fixed point is an approximate solution to (5.1) using linear function approximation. We consider the quality of the approximation in the next section.

In general, there could be no TD fixed point, one TD fixed point, or infinitely many TD fixed points, as in the discounted setting. To see this, let $y_t \doteq [\eta, x_t^\top]^\top$, $y_t' \doteq [\eta, x_t'^\top]^\top, u \doteq [\bar{R}, w^\top]^\top$, and $e_1 \doteq [1, 0, \cdots, 0]^\top \in \mathbb{R}^{d+1}$. Then combining (5.4) and (5.5) gives

$$\mathbb{E}[\delta_t(u) y_t] = \mathbf{0}, \tag{5.6}$$

where $\delta_t(u) \doteq R_t - e_1^\top u + y_t'^\top u - y_t^\top u$.

Let P_π denote the transition matrix under π (Equation 3.3). Writing (5.6) in vector form, we have $Au + b = \mathbf{0}$, where

$$\begin{aligned} A &\doteq \mathbb{E}[y_t(-e_1 + y_t' - y_t)^\top] \\ &= Y^\top D(P_\pi - I)Y - Y^\top d_\mu e_1^\top \\ &= \begin{bmatrix} -\eta & \mathbf{1}^\top D(P_\pi - I)X \\ -X^\top d_\mu & X^\top D(P_\pi - I)X \end{bmatrix}, \\ b &\doteq \mathbb{E}[y_t R_t] = Y^\top D r, \\ Y &\doteq [\eta \mathbf{1}, X], D \doteq diag(d_\mu). \end{aligned}$$

If and only if A is invertible, there exists a unique TD fixed point

$$u_{\mathrm{TD}} \doteq -A^{-1} b. \tag{5.7}$$

Otherwise, there is either no TD fixed point or there are infinitely many.

Unfortunately, even if there exists a unique TD fixed point, Differential SGQ can still diverge, which exemplifies the deadly triad (Sutton and Barto 2018) in the average-reward setting. The following example confirms this point.

Example 5.1 (The divergence of Differential SGQ**).** Consider a two-state MDP (Figure 5.1). The expected Differential SGQ update per step, when choosing $\eta = 1$, can be written as $\begin{bmatrix} \bar{R}_{t+1} \\ w_{t+1} \end{bmatrix} = \begin{bmatrix} \bar{R}_t \\ w_t \end{bmatrix} + \alpha \left(A \begin{bmatrix} \bar{R}_t \\ w_t \end{bmatrix} + b \right) = \begin{bmatrix} \bar{R}_t \\ w_t \end{bmatrix} + \alpha \begin{bmatrix} -1 & 6 \\ -2 & 6 \end{bmatrix} \begin{bmatrix} \bar{R}_t \\ w_t \end{bmatrix}$. Here, we consider α a constant step size. The eigenvalues of

$A = \begin{bmatrix} -1 & 6 \\ -2 & 6 \end{bmatrix}$ are both positive. Hence, no matter what positive step size is picked, the expected update diverges. The sample updates (5.2) and (5.3) using standard stochastic approximation step sizes, therefore, also diverge. Furthermore, because both eigenvalues are positive, A is an invertible matrix, implying the unique existence of the TD fixed-point.

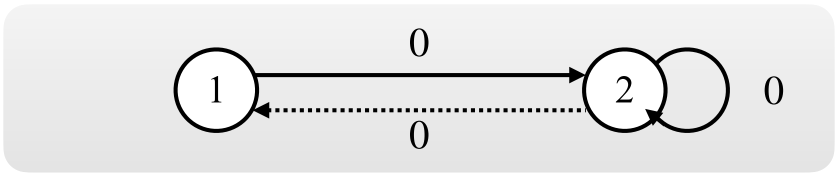


Figure 5.1: An example showing the divergence of Differential SGQ. In this MDP, the target policy always chooses action `solid`. The sampling distribution satisfies that $d_\mu(1, \texttt{solid}) = d_\mu(1, \texttt{dashed}) = 6/13$, $d_\mu(2, \texttt{solid}) = 1/13$. Each state-action pair has a feature vector of size 1. $x(1, \texttt{solid}) = x(2, \texttt{dashed}) = 1$, and $x(2, \texttt{solid}) = 14$.

5.3 One-Stage Differential Gradient Q-Evaluation

We now present Differential GQ1, a proven-convergent off-policy prediction learning algorithm in the LFA setting.

Motivated by the Mean Squared Projected Bellman Error (MSPBE) defined in the discounted setting and used by Gradient TD algorithms, we define the MSPBE in the average-reward setting as follows. Given an approximation of the reward rate $\bar{r}$ and and an approximation of the differential action-value function q, let $u^\top = [\bar{r}, q^\top]$, the MSPBE can be defined as

$$\text{MSPBE}_1(u) \doteq \left\| \Pi_Y \bar{\delta}(u) \right\|_D^2, \tag{5.8}$$

where

$$\Pi_Y \doteq Y(Y^\top D Y)^{-1} Y^\top D$$

is the *projection matrix* onto the column space of Y, and

$$\bar{\delta}(u) \doteq r - e_1^\top u \mathbf{1} + P_\pi Y u - Y u \tag{5.9}$$

is the vector of *Bellman errors* for all state-action pairs. Therefore, the vector $\Pi_Y \bar{\delta}(u)$ is the projection of the vector of Bellman errors on the column space of Y and thus a vector of *projected Bellman errors*. (5.8) is a weighted mean of squared projected Bellman errors, with the weight vector $diag(D) = d_\mu$ and is therefore called an MSPBE.

The existence of the matrix inverse in Π_Y, $(Y^\top DY)^{-1}$, is guaranteed by Assumptions 5.2, 5.3 and the following assumption.

Assumption 5.4. For any $w \in \mathbb{R}^d$, $Xw \neq \mathbf{1}$.

The above assumption guarantees that if w^* is a solution of w in (5.4) and (5.5), then no other solution's approximated action-value function would be identical to Xw^* up to a constant. This assumption is also used by Tsitsiklis and Van Roy (1999) in their on-policy prediction algorithms in average-reward MDPs. Apparently, the assumption does not hold in the tabular setting (i.e., when $X = I$). However, with function approximation, we usually have many more states than features (i.e., $|\mathcal{S}| \gg d$), in which case the above assumption would not be restrictive.

Let $C \doteq Y^\top DY$, we have $\Pi^\top D\Pi = DYC^{-1}Y^\top D$, with which we give a different form for (5.8):

$$\begin{aligned} \text{MSPBE}_1(u) &= \left\| Y^\top D\bar{\delta}(u) \right\|^2_{C^{-1}} = \|Au + b\|^2_{C^{-1}} \\ &= \mathbb{E}[\delta_t(u)y_t]^\top \mathbb{E}[y_t y_t^\top]^{-1} \mathbb{E}[\delta_t(u)y_t]. \end{aligned} \tag{5.10}$$

It can be seen that if (5.6) has a solution, then that solution also minimizes (5.10), in which case solving (5.6) can be converted to minimizing (5.10). However, when (5.6) does not have a unique solution, the set of minimizers of (5.10) could be unbounded, and thus algorithms minimizing MSPBE_1 risk generating unbounded updates. To ensure the stability of the algorithm when (5.6) does not have a unique solution, consider a regularized MSPBE_1 objective:

$$J_{1,c}(u) \doteq \|Au + b\|^2_{C^{-1}} + cu^\top I_0 u,$$

where $I_0 \doteq diag(\mathbf{1} - e_1)$, c is a positive scalar, and $cu^\top I_0 u = c\|w\|^2$ is a ridge regularization term on w. Introducing a regularization term in MSPBE-like objectives is not new though; see, for example, Mahadevan et al. (2014); Yu (2017); Du et al. (2017); Zhang et al. (2020b;c). One could, of course, apply regularization to Differential SGQ directly, similar to Diddigi et al. (2020) in the discounted off-policy linear TD. Unfortunately, the weight for their regularization term has to be sufficiently large to ensure convergence.

To minimize $J_{1,c}(u)$, one could proceed with techniques used in TDC (Sutton et al. 2009), which is left as future work. In this chapter, let's proceed with the saddle-point formulation of GTD2 introduced by Liu et al. (2015), which exploits Fenchel's duality:

$$u^\top M^{-1} u = \max_\nu 2u^\top \nu - \nu^\top M \nu,$$

for any positive definite M, yielding

$$J_{1,c}(u) = \max\nolimits_{\nu \in \mathbb{R}^{d+1}} 2\nu^\top Y^\top D\bar{\delta}(u) - \nu^\top C\nu + cu^\top I_0 u. \tag{5.11}$$

So $\min_u J_{1,c}(u) = \min_u \max_\nu J_{1,c}(u, \nu)$, where

$$J_{1,c}(u, \nu) \doteq 2\nu^\top Y^\top D\bar{\delta}(u) - \nu^\top C\nu + cu^\top I_0 u.$$

As $J_{1,c}(u, \nu)$ is convex in u and concave in ν, we have now reduced the problem to a convex-concave saddle point problem. Applying primal-dual methods to this problem, that is, performing gradient ascent for ν following $\nabla_\nu J_{1,c}(u, \nu)$ and gradient descent for u following $\nabla_u J_{1,c}(u, \nu)$, we arrive at *One-Stage Differential Gradient Q Evaluation*, or *Differential GQ1*. At time step t, with a sample $(S_t, A_t, R_t, S'_t, A'_t)$ from $d_{\mu\pi}$, Differential GQ1 updates u_t and ν_t as

$$\delta_t \doteq R_t - e_1^\top u_t + y_t'^\top u_t - y_t^\top u_t, \tag{5.12}$$

$$\nu_{t+1} \doteq \nu_t + \alpha_t(\delta_t - y_t^\top \nu_t)y_t, \tag{5.13}$$

$$u_{t+1} \doteq u_t + \alpha_t(y_t - y'_t + e_1)y_t^\top \nu_t - \alpha_t c I_0 u_t, \tag{5.14}$$

where $\{\alpha_t\}_{t \geq 0}$ is a sequence of step sizes satisfying the standard assumption Assumption 3.5.

The algorithm is called one-stage because, while there are two weight vectors updated in every iteration, both converge simultaneously.

Algorithm 15: Differential GQ1

```
Input: A sampling distribution $d_{\mu\pi}$ and a feature mapping
       $x : \mathcal{S} \times \mathcal{A} \to \mathbb{R}^d$
Algorithm parameters: A step-size sequence $\alpha_t$, a scaling
       parameter $\eta$, and a regularization weight $c$.
1 Initialize $\nu, u \in \mathbb{R}^{d+1}$ arbitrarily (e.g., to zero)
2 for $t = 0, 1, 2, \ldots$ do
3     Sample a transition $(S, A, R, S', A')$ from $d_{\mu\pi}$
4     $y = \begin{bmatrix} \eta \\ x(S, A) \end{bmatrix}, y' = \begin{bmatrix} \eta \\ x(S', A') \end{bmatrix}$
5     $\delta \doteq R - e_1^\top u + y'^\top u - y^\top u$
6     $\nu \leftarrow \nu + \alpha_t(\delta - y^\top \nu)y$
7     $u \leftarrow u + \alpha_t(y - y' + e_1)y^\top \nu - \alpha c I_0 u$
8 end
9 return the estimated reward rate $u[0]$, differential action-value
    function $Xu[1 :]$, where $X$ is the feature matrix.
```

Theorem 5.1. *If Assumptions 5.1—5.4, and 3.5 hold, then for any $c > 0$, almost surely, the iterate $\{u_t\}$ generated by Differential GQ1 (Equations 5.12–5.14) converges to u_c^*, where $u_c^* \doteq -(cI_0 + A^\top C^{-1} A)^{-1} A^\top C^{-1} b$ is the unique minimizer of $J_{1,c}(u)$. Further, if A is invertible, then for $c = 0$, $\{u_t\}$ converges almost surely to u_{TD} defined in* (5.7).

Proof. The proof of Theorem 5.1 uses a result from Borkar (2009) and mimics the convergence proof of GTD2 in Sutton et al. (2009).

I first state the result from Borkar (2009). Consider updating $\theta \in \mathbb{R}^d$ by

$$\theta_{t+1} \doteq \theta_t + \alpha_t(G_{t+1}\theta_t + h_{t+1}),$$

where $G_{t+1} \in \mathbb{R}^{d \times d}, h_{t+1} \in \mathbb{R}^d$. Assuming

Assumption 5.5. There exist $\bar{G} \in \mathbb{R}^{d \times d}$ and $\bar{h} \in \mathbb{R}^d$ such that

$$M_{t+1} \doteq G_{t+1}\theta_t + h_{t+1} - \bar{G}\theta_t - \bar{h}$$

satisfies

1. $\mathbb{E}[M_{t+1} \mid \mathcal{F}_t] = \mathbf{0}$ a.s.

2. $\mathbb{E}[\|M_{t+1}\|^2 \mid \mathcal{F}_t] \leq C(1 + \|\theta_t\|^2)$ for some constant $C > 0$ a.s.

Here

$$\mathcal{F}_t \doteq \sigma(x_0, M_1, M_2, \ldots, M_t),$$

where $\sigma(\cdot)$ denotes the σ-field.

Assumption 5.6. The real part of every eigenvalue of $\bar{G}$ is strictly negative.

Theorem 5.2. *(Borkar (2009)) Under Assumptions 3.5, 5.5, and 5.6, almost surely,*

$$\lim_{t\to\infty} \theta_t = -\bar{G}^{-1}\bar{h}$$

Theorem 5.2 can be directly obtained from Theorem 2 in Chapter 2 and Theorem 7 in Chapter 3 of Borkar (2009).

I proceed by verifying Assumptions 3.5, 5.5, and 5.6 thus invoking Theorem 5.2. With $\theta_t \doteq [\nu_t^\top, u_t^\top]^\top$, we rewrite Equations 5.12–5.14 as

$$\theta_{t+1} \doteq \theta_t + \alpha_t(G_{t+1}\theta_t + h_{t+1}),$$

where

$$\begin{aligned} G_{t+1} &\doteq \begin{bmatrix} -y_t y_t^\top & y_t(y_t' - y_t)^\top - y_t e_1^\top \\ (y_t - y_t')y_t^\top + e_1 y_t^\top & -cI_0 \end{bmatrix}, \\ h_{t+1} &\doteq \begin{bmatrix} y_t r_t \\ \mathbf{0} \end{bmatrix}, I_0 \doteq \begin{bmatrix} 0 & \mathbf{0}^\top \\ \mathbf{0} & I \end{bmatrix}. \end{aligned}$$

The asymptotic behavior of $\{\theta_t\}$ is governed by

$$\begin{aligned} \bar{G} &\doteq \mathbb{E}[G_{t+1}] = \begin{bmatrix} -C & A \\ -A^\top & -cI_0 \end{bmatrix}, \\ \bar{h} &\doteq \mathbb{E}[h_{t+1}] = \begin{bmatrix} b \\ \mathbf{0} \end{bmatrix}, \end{aligned}$$

where

$$\begin{aligned} A &\doteq Y^\top D(P_\pi - I)Y - Y^\top d_\mu e_1^\top \\ &= \begin{bmatrix} -\eta & \eta \mathbf{1}^\top D(P_\pi - I)X \\ -X^\top d_\mu & X^\top D(P_\pi - I)X \end{bmatrix} \\ b &\doteq \begin{bmatrix} Y^\top D r \\ \mathbf{0} \end{bmatrix}. \end{aligned}$$

Assumption 3.5 is assumed in the theorem being proved. For Assumption 5.5, we define

$$M_{t+1} \doteq G_{t+1}\theta_t + h_{t+1} - \bar{G}\theta_t - \bar{h}.$$

It is easy to see

$$\begin{aligned}\mathbb{E}[M_{t+1} \mid \mathcal{F}_t] &= \mathbb{E}[G_{t+1}]\theta_t + \mathbb{E}[h_{t+1}] - \bar{G}\theta_t - \bar{h} = \mathbf{0} \\ \mathbb{E}[\|M_{t+1}\|^2 \mid \mathcal{F}_t] &\leq \frac{1}{2}\mathbb{E}[\left\|G_{t+1} - \bar{G}\right\|^2 \|\theta_t\|^2 + \left\|h_{t+1} - \bar{h}\right\|^2 |\mathcal{F}_t].\end{aligned}$$

Because our samples are generated in an i.i.d fashion, Assumption 5.5 is guaranteed to hold.

To verify Assumption 5.6, we first show $\det\left(\bar{G}\right) \neq 0$. Using the rule of block matrix determinant, which states that for any square matrix $E = \begin{bmatrix} A & B \\ C & D \end{bmatrix}$, where A, B, C, D are all square matrices, $\det(E) = \det(A)\det(D - CA^{-1}B)$, we have

$$\det\left(\bar{G}\right) = \det(C)\det\left(cI_0 + A^\top C^{-1}A\right).$$

Assumption 5.4 ensures C is positive definite and $A^\top C^{-1}A$ is positive semidefinite, implying $cI_0 + A^\top C^{-1}A$ is positive semidefinite. For any $z \neq \mathbf{0} \in \mathbb{R}^{k+1}$, $z^\top I_0 z = 0$ if and only if z has the form $\begin{bmatrix} c \\ \mathbf{0} \end{bmatrix}$ for some $c \neq 0 \in \mathbb{R}$, implying $A^\top z \neq \mathbf{0}$, i.e., $z^\top A^\top C^{-1}Az \neq 0$. So as long as $c > 0$, $z^\top(cI_0 + A^\top C^{-1}A)z \neq 0$, implying $cI_0 + A^\top C^{-1}A$ is positive definite. It follows easily that $\det\left(\bar{G}\right) \neq 0$. Let $\lambda \in \mathbb{C}$ be an eigenvalue of $\bar{G}$. $\det\left(\bar{G}\right) \neq 0$ implies $\lambda \neq 0$. Let $z \neq \mathbf{0} \in \mathbb{C}^{2K+2}$ be the corresponding normalized eigenvector of λ, i.e., $z^H z = 1$, where z^H is the conjugate transpose of z. Let $z = \begin{bmatrix} z_1 \\ z_2 \end{bmatrix}$, we have

$$\lambda = z^H \bar{G} z = -z_1^H C z_1 - z_2^H A^\top z_1 + z_1^H A z_2 - c z_2^H I_0 z_2.$$

As $(z_2^H A^\top z_1)^H = z_1^H A z_2$, we have $\mathrm{Re}(-z_2^H A^\top z_1 + z_1^H A z_2) = 0$, where $\mathrm{Re}(\cdot)$ denotes the real part. So

$$\mathrm{Re}(\lambda) = -z_1^H C z_1 - c z_2^H I_0 z_2 \leq 0.$$

Because $\lambda \neq 0$, we have $\mathrm{Re}(\lambda) < 0$. Assumption 5.6 then holds. Invoking Theorem 5.2 yields

$$\lim_{t \to \infty} \theta_t = -\bar{G}^{-1}\bar{h} \quad \text{almost surely.}$$

Let u_c^* be the lower half of $-\bar{G}^{-1}\bar{h}$, we have

$$u_c^* \doteq -(cI_0 + A^\top C^{-1} A)^{-1} A^\top C^{-1} b.$$

From (5.11), we can rewrite $J_{1,c}(u)$ as

$$J_{1,c}(w) = \|Au + b\|^2_{C^{-1}} + cu^\top I_0 u$$

It is easy to verify (e.g., using the first order optimality condition of $J_{1,c}(u)$) that u_c^* is the unique minimizer of $J_{1,c}(u)$.

It can also be seen that if $c = 0$ and A is invertible, $\det\big(\bar{G}\big) \neq 0$ as well and $u_{c=0}^* = -A^{-1}b = u_{TD}$. $\square$

Let's now analyze the quality of TD fixed points.

Assumption 5.7. There exists at least one TD fixed point.

Let $u^* = [\bar{r}^*, w^{*\top}]^\top$ be one fixed point (a solution of (5.6)). We are interested in the upper bound of the absolute value of the difference between the estimated reward rate and the true reward rate $|\bar{r}^* - r(\pi)|$ and also the upper bound of the minimum distance between the estimated differential value function $Xw^{*\top}$ to the set $\{q_\pi + c\mathbf{1}\}$. In general, as long as there is a representation error, the TD fixed point can be arbitrarily poor in terms of approximating the value function, even in the discounted case (see Kolter (2011) for more discussion). In light of this, we study the bounds only when d_μ is close to d_π, the stationary state-action distribution of π, in the sense of the following assumption. Let $\xi \in (0, 1)$ be a constant,

Assumption 5.8. F is positive semidefinite, where

$$F \doteq \begin{bmatrix} X^\top D X & X^\top D P_\pi X \\ X^\top P_\pi^\top D X & \xi^2 X^\top D X \end{bmatrix}.$$

A similar assumption about F is also used by Kolter (2011) in the analysis of the performance of the MSPBE minimizer in the discounted setting. Kolter (2011) uses $\xi = 1$ while we use $\xi < 1$ to account for the lack of discounting. Section D.1 of Zhang et al. (2021) shows with a simulation that this assumption holds with reasonable probability in randomly generated MDPs. Furthermore, consider the bounds when all the features have zero mean under the distribution d_μ.

Assumption 5.9. $X^\top d_\mu = \mathbf{0}$.

This can easily be achieved by subtracting each feature vector sampled in our learning algorithm by some estimated mean feature vector, which is the empirical average of all the feature vectors sampled from d_μ. Note that without this mean-centered feature assumption, a looser bound can also be obtained. We are now ready to show a bound of the convergent point. This bound is the first bound for the TD fixed point (up to some regularization if necessary) in the off-policy setting concerning the average-reward formulation.

Proposition 5.1. *Under Assumptions 5.1–5.4, 5.7–5.9,*

$$\inf_{c\in\mathbb{R}} \|Xw^* - q_\pi^c\|_D \leq \frac{\|P_\pi\|_D + 1}{1-\xi} \inf_{c\in\mathbb{R}} \|\Pi_X q_\pi^c - q_\pi^c\|_D,$$
$$|\bar{r}^* - r(\pi)| \leq \frac{\left\|d_\mu^\top(P_\pi - I)\right\|_{D^{-1}}(\|P_\pi\|_D+1)}{1-\xi} \inf\nolimits_{c\in\mathbb{R}} \|\Pi_X q_\pi^c - q_\pi^c\|_D,$$

where $q_\pi^c \doteq q_\pi + c\mathbf{1}$, $\|P_\pi\|_D \doteq \max_{\|x\|_D \leq 1} \|P_\pi x\|_D$, *and* $\Pi_X = X(X^\top DX)^{-1}X^\top D$ *is a projection matrix.*

Proof. That u^* is a TD fixed point implies

$$\mathbb{E}[\delta_t(u^*)y_t] = \mathbf{0},$$

which implies

$$Y^\top D(P_\pi - I)Yu^* - Y^\top d_\mu e_1^\top u^* + Y^\top Dr = \mathbf{0},$$

expanding which yields

$$\bar{r}^* - d_\mu^\top(r + P_\pi Xw^* - Xw^*) = \mathbf{0},$$
$$X^\top D(r - \bar{r}^*\mathbf{1} + P_\pi Xw^* - Xw^*) = \mathbf{0}.$$

So we have

$$\left\|X^\top D(r - \bar{r}^* \mathbf{1} + P_\pi X w^* - X w^*)\right\|^2_{(X^\top D X)^{-1}} = \mathbf{0},$$

implying

$$\|\Pi_X (r - \bar{r}^* \mathbf{1} + P_\pi X w^* - X w^*)\|^2_D = \mathbf{0}.$$

Using the Schur complement, Assumption 5.8 implies (see Kolter (2011) for more details)

$$\|\Pi_X P_\pi X w\|_D \le \xi \|X w\|_D$$

holds for any $w \in \mathbb{R}^K$. We have

$$\begin{aligned}
&\|Xw^* - q^c_\pi\|_D \\
\le&\|Xw^* - \Pi_X q^c_\pi\|_D + \|\Pi_X q^c_\pi - q^c_\pi\|_D \\
=&\|\Pi_X(r + P_\pi X w^* - \bar{r}^*\mathbf{1}) - \Pi_X(r + P_\pi q^c_\pi - r(\pi)\mathbf{1})\|_D + \|\Pi_X q^c_\pi - q^c_\pi\|_D \\
\le&\|\Pi_X P_\pi X w^* - \Pi_X P_\pi q^c_\pi\|_D + \|\Pi_X(\bar{r}^*\mathbf{1} - r(\pi)\mathbf{1})\|_D + \|\Pi_X q^c_\pi - q^c_\pi\|_D \\
=&\|\Pi_X P_\pi X w^* - \Pi_X P_\pi q^c_\pi\|_D + \left\|X(X^\top D X)^{-1}(X^\top D \mathbf{1})(\bar{r}^* - r(\pi))\right\|_D \\
&+ \|\Pi_X q^c_\pi - q^c_\pi\|_D \\
=&\|\Pi_X P_\pi X w^* - \Pi_X P_\pi q^c_\pi\|_D + \|\Pi_X q^c_\pi - q^c_\pi\|_D \qquad\qquad (\text{By } X^\top d_\mu = \mathbf{0}) \\
\le&\|\Pi_X P_\pi X w^* - \Pi_X P_\pi \Pi_X q^c_\pi\|_D + \|\Pi_X P_\pi \Pi_X q^c_\pi - \Pi_X P_\pi q^c_\pi\|_D + \|\Pi_X q^c_\pi - q^c_\pi\|_D \\
\le&\xi\|Xw^* - \Pi_X q^c_\pi\|_D + \|P_\pi\|_D \|\Pi_X q^c_\pi - q^c_\pi\|_D + \|\Pi_X q^c_\pi - q^c_\pi\|_D \\
\le&\xi\|Xw^* - q^c_\pi\|_D + (\|P_\pi\|_D + 1)\|\Pi_X q^c_\pi - q^c_\pi\|_D.
\end{aligned}$$

From the above derivation, we have

$$\|Xw^* - q^c_\pi\|_D \le \frac{\|P_\pi\|_D + 1}{1 - \xi}\|\Pi_X q^c_\pi - q^c_\pi\|_D.$$

Take the infimum

$$\inf_{c\in\mathbb{R}} \|Xw^* - q^c_\pi\|_D \le \inf_{c\in\mathbb{R}} \frac{\|P_\pi\|_D + 1}{1 - \xi}\|\Pi_X q^c_\pi - q^c_\pi\|_D.$$

For the reward rate at the fixed point, we have, for all $c \in \mathbb{R}$,

$$\begin{aligned}
|r(\pi) - \bar{r}^*| &= |d_\mu^\top (P_\pi - I)(Xw^* - q^c_\pi)| \\
&= |d_\mu^\top (P_\pi - I) D^{-\frac{1}{2}} D^{\frac{1}{2}} (Xw^* - q^c_\pi)| \\
&\le \left\|d_\mu^\top (P_\pi - I)\right\|_{D^{-1}} \|Xw^* - q^c_\pi\|_D,
\end{aligned}$$

where the inequality is due to the Cauchy-Schwarz inequality.

□

As a special case, there exists a unique TD fixed point in the on-policy case (i.e., $d_\mu = d_\pi$) under Assumptions 5.1, 5.2, and 5.4. Then $|r(\pi) - \bar{r}^*| = 0$ as $d_\pi^\top (P_\pi - I) = \mathbf{0}$ and a tighter bound for the estimated differential value function can be obtained. See Tsitsiklis and Van Roy (1999) for details.

5.4 Two Related Algorithms

This section discusses two related algorithms for solving the problem considered in this chapter. They are not contributions of this dissertation but are still important to be mentioned to help better understand Differential GQ1. In addition, they are used as baseline algorithms in my experiments in Section 5.5.

The first algorithm, called Differential GQ2 (Zhang et al. 2021), is a sibling algorithm of Differential GQ1. Differential GQ2 was derived from a different MSPBE objective, called MSPBE$_2$. Note that these two MSPBE objectives not only have different forms—they also have different gradients and different minimizers. The fact that we have at least two MSPBEs in the average-reward setting might sound surprising because, in the discounted setting, there is only one MSPBE objective.

MSPBE$_2$ can be derived from MSPBE$_1$. Note that MSPBE$_1$ involves an unknown variable $\bar{r}$ for the reward rate. However, it is not necessary to use such a variable to represent the reward rate because the reward rate of a policy can be obtained directly given action values of the policy using (5.1):

$$r(\pi) = \sum_{s',r} p(s', r \mid s, a) \left(r + \sum_{a'} \pi(a' \mid s') q(s', a') - q(s, a) \right), \forall s, a.$$

Therefore

$$r(\pi) = \sum_{s,a} d(s,a) \sum_{s',r} p(s', r \mid s, a) \left(r + \sum_{a'} \pi(a' \mid s') q(s', a') - q(s, a) \right),$$

for any probability distribution d over the state-action space. One may replace $q(s,a)$ with an approximate action-value function $x(s,a)^\top w$, choose $d = d_\mu$, and obtain an approximation of the reward rate,

$$\sum_{s,a} d_\mu(s,a) \sum_{s',r} p(s', r \mid s, a) \left(r + \sum_{a'} \pi(a' \mid s') x(s', a')^\top w - x(s,a)^\top w \right) \approx r(\pi).$$

Write the r.h.s. of the above equation in the vector form, we have $d_\mu^\top(r + P_\pi X w - Xw)$. Here with a bit of abuse of notation, we use r to denote the one-step reward vector. Replacing $\bar{r}$ in the Bellman errors (5.9) (note that $\bar{r}$ is hidden in u) with $d_\mu^\top(r + P_\pi X w - Xw)$ and note that $P_\pi Y u - Y u = P_\pi X w - Xw$, we have a new Bellman error

$$\bar{\delta}(w) = r - d_\mu^\top(r + P_\pi X w - Xw)\mathbf{1} + P_\pi X w - Xw.$$

A new MSPBE can then be defined using this new Bellman error:

$$\text{MSPBE}_2(w) = \left\| \Pi_X \bar{\delta}(w) \right\|_D^2.$$

There is an interesting fact about these two objectives. While they are different, if the TD fixed point uniquely exists, the minimizers of the two objectives are both the TD fixed point. Readers may refer to Zhang et al. (2021) for more discussion about this fact.

Differential GQ2 was designed to optimize MSPBE_2 (with an extra regularization term just as in Differential GQ1) and has been proved to converge a.s. to the minimum of the objective (Zhang et al. 2021). It maintains two weight vectors $w, \nu \in \mathbb{R}^d$ and a scalar $\bar{R}$. Here w is used to approximate the differential action-value function, $\bar{R}$ is used to approximate the reward rate, and ν is an auxiliary parameter. At each odd step ($t = 1, 3, 5, \ldots$), the algorithm uses the recent two transitions ($(S_t, A_t, R_t, S'_t, A'_t)$ and $(S_{t-1}, A_{t-1}, R_{t-1}, S'_{t-1}, A'_{t-1})$) from $d_{\mu\pi}$ to updates ν_t and w_t:

$$\nu_{t+2} \doteq \nu_t + \alpha_t \Big((R_t + x_t'^\top w_t - x_t^\top w_t) - (R_{t-1} + x_{t-1}'^\top w_t - x_{t-1}^\top w_t) - x_t^\top \nu_t \Big) x_t,$$
$$w_{t+2} \doteq w_t + \alpha_t \Big((x_t - x'_t) + (x_{t-1} - x'_{t-1}) \Big) x_t^\top \nu_t - \alpha_t c w_t,$$

where $x_t \doteq x(S_t, A_t), x_{t-1} \doteq x(S_{t-1}, A_{t-1}), x'_t \doteq x(S'_t, A'_t), x'_{t-1} \doteq x(S'_{t-1}, A'_{t-1}), \{\alpha_t\}$ is a step-size sequence, and updates $\bar{R}_t$ by

$$\bar{R}_{t+2} \doteq \bar{R}_t + \eta\alpha_t\Big(0.5(R_t + x_t'^{\top} w_t - x_t^{\top} w_t) + 0.5(R_{t-1} + x_{t-1}'^{\top} w_t - x_{t-1}^{\top} w_t) - \bar{R}_t\Big),$$

where $c > 0$ is a constant. For convenience, I put the pseudocode of Differential GQ2 in Algorithm 16.

```
Algorithm 16: Differential GQ2
   Input: A sampling distribution $d_{\mu\pi}$ and a feature mapping
          $x : \mathcal{S} \times \mathcal{A} \to \mathbb{R}^K$
   Algorithm parameters: A step-size sequence $\alpha_t$, a scaling
                         parameter $\eta$, and a regularization weight $c$
 1 Initialize $\nu, w \in \mathbb{R}^d$ and $\bar{R} \in \mathbb{R}$ arbitrarily (e.g., to zero)
 2 for $t = 0, 1, 2, \ldots$ do
 3     Sample two transitions $(S_1, A_1, R_1, S'_1, A'_1)$ and $(S_2, A_2, R_2, S'_2, A'_2)$
        from $d_{\mu\pi}$
 4     $x_1 = x(S_1, A_1), x'_1 = x(S'_1, A'_1), x_2 = x(S_2, A_2), x'_2 = x(S'_2, A'_2)$
 5     $\tilde{R}_1 \leftarrow R_1 + x_1'^{\top} w - x_1^{\top} w$
 6     $\tilde{R}_2 \leftarrow R_2 + x_2'^{\top} w - x_2^{\top} w$
 7     $\nu \leftarrow \nu + \alpha_t\Big(\tilde{R}_1 - \tilde{R}_2 - x_1^{\top}\nu\Big)x_1$
 8     $w \leftarrow w + \alpha_t\Big((x_1 - x'_1) + (x_2 - x'_2)\Big)x_1^{\top}\nu - \alpha c w$
 9     $\bar{R} \leftarrow \bar{R} + \eta\alpha_t\Big((\tilde{R}_1 + \tilde{R}_2)/2 - \bar{R}\Big)$
10 end
11 return the estimated reward rate $\bar{R}$, differential action-value function
    $Xw$, where $X$ is the feature matrix.
```

Differential SGQ, Differential GQ1, and Differential GQ2 are the first function approximation methods that estimate the reward rate with the help of an estimated differential action-value function while almost all of the existing approaches (e.g., Liu et al. 2018; Zhang et al. 2020a;b) estimate the reward rate by estimating the ratio between the target policy's stationary distribution and $d_{\mu\pi}$. An estimate of the reward rate can be directly easily using an estimated density ratio. Algorithms estimating such a ratio are called density-ratio-based methods. Only two prior algorithms do not follow this approach. To compute the reward rate, the algorithm by Mousavi et al. (2020) uses an estimated stationary distribution of the target policy, and the algorithm by Lazic et al. (2020) uses an estimated world model.

To see how to estimate the reward given an estimate of the density ratio. let $d_\pi(s,a)$ be the unique stationary state-action distribution under the target policy π. The uniqueness is guaranteed given the unichain assumption on π. Note that with the density ratio $\tau(s,a) \doteq d_\pi(s,a)/d_{\mu\pi}(s,a)$, the reward rate of π,

$$r(\pi) = \mathbb{E}\left[\frac{d_\pi(S_t,A_t)}{d_\mu(S_t,A_t)} R_t \mid S_t, A_t \sim d_\mu\right].$$

Once τ is approximated accurately using some $\bar{\tau}$, the agent can simply estimate $r(\pi)$ using $\bar{R}_t$, updated by

$$\bar{R}_{t+1} = \bar{R}_t + \alpha_t(\bar{\tau}(S_t,A_t)R_t - \bar{R}_t)$$

where α_t is a step-size sequence.

GradientDICE is a competitive density-ratio-based baseline approach. In fact, by the time our paper (Zhang et al. 2021) was published, Gradient-DICE was the only density-ratio-based approach for off-policy prediction in average-reward MDPs that is provably convergent with general LFA and has $\mathcal{O}(d)$ computational complexity per step. The pseudocode of GradientDICE is shown in Algorithm 17.

Algorithm 17: GradientDICE

```
   Input: A sampling distribution $d_{\mu\pi}$ and a feature mapping
          $x : \mathcal{S} \times \mathcal{A} \to \mathbb{R}^K$
   Algorithm parameters: A step-size sequence $\alpha_t$ and two
                         regularization parameters $\lambda, c$
1  Initialize $\nu, w \in \mathbb{R}^d$ and $K, \bar{R} \in \mathbb{R}$ arbitrarily (e.g., to zero)
2  for $t = 0, 1, 2, \ldots$ do
3      Sample a transitions $(S, A, R, S', A')$ from $d_{\mu\pi}$
4      $x = x(S,A), x' = x(S',A')$
5      $w \leftarrow w - \alpha_t(\nu^\top(x'-x)x + \lambda K x_t + cw)$
6      $\nu \leftarrow \nu + \alpha_t(w^\top x(x'-x) - \nu^\top x \nu)$
7      $K \leftarrow K + \alpha_t \lambda(w^\top x - 1 - K)$
8      $\bar{R} \leftarrow \bar{R} + \alpha_t(w^\top x R - \bar{R})$
9  end
10 return the estimated reward rate $\bar{R}$, differential action-value function
     $Xw$, where $X$ is the feature matrix.
```

The LFA version of GradientDICE, with an additional reward rate estimation part, maintains two weight vectors $w, \nu \in \mathbb{R}^d$ and two scalars K and $\bar{R}$.

Here w is used to approximate the density ratio, with $w^\top x(s,a) \approx \tau(s,a)$, and $\bar{R}$ is used to approximate the reward rate. ν and K are auxiliary parameters. At each time step, using the transition $(S_t, A_t, R_t, S'_t, A'_t)$ sampled from $d_{\mu\pi}$, GradientDICE updates its parameters using the following rules:

$$\begin{aligned}
w_{t+1} &= w_t - \alpha_t(\nu_t^\top (x'_t - x_t)x_t + \lambda K_t x_t + c w_t)\\
\nu_{t+1} &= \nu_t + \alpha_t(w_t^\top x_t(x'_t - x_t) - \nu_t^\top x_t \nu_t)\\
K_{t+1} &= K_t + \alpha_t \lambda (w_t^\top x_t - 1 - K_t)\\
\bar{R}_{t+1} &= \bar{R}_t + \alpha_t(w_t^\top x_t R_t - \bar{R}_t),
\end{aligned}$$

where $x_t = x(S_t, A_t), x'_t = x(S'_t, A'_t)$, and λ is a parameter that pushes the sum of $d_\mu^\top \tau$, which is an estimate of d_π, to one. Note that the $\bar{R}_t$ update does not appear in the original GradientDICE algorithm by Zhang et al., (2020b) as the original algorithm was only intended to estimate a density ratio, not a reward rate.

5.5 Experiments

This section empirically studies Differential GQ1. The baseline algorithms are Differential SGQ (see Section 5.2), Differential GQ2, and GradientDICE, all with linear function approximation.

Just as I did in previous chapters, I used the four-room domain (Figure 3.1) as the test domain. The target policy is an optimal one. Unlike previous experiments, where the agent interacts with the world following some behavior policy, here the agent's experience is generated by a fixed probability distribution d_μ in light of the problem setup described in Section 5.1. To ensure that the data sampled from $d_{\mu\pi}$ comes from an off-policy distribution, $d_{\mu\pi}$ should be different from the stationary distribution under π. In this experiment, $d_{\mu\pi}$ was set as follows.

$$d_{\mu\pi}(s,a) = \begin{cases} \left(1 - \epsilon + \frac{\epsilon}{4}\right) d_\mu(s) & a = \pi(s) \\ \frac{\epsilon}{4} d_\mu(s) & a \neq \pi(s) \end{cases},$$

where

$$d_\mu(s) \doteq \begin{cases} \frac{1-\epsilon}{17} + \frac{\epsilon}{104} & s \text{ is visited by } \pi. \\ \frac{\epsilon}{104} & \text{otherwise} \end{cases}$$

state-action pairs were then sampled from $d_{\mu\pi}$. To understand $d_{\mu\pi}$, note that sampling a pair from it is equivalent to the following sampling process. With probability $1-\epsilon$, a state s is sampled from states that are visited by the target policy (there are 17 such states). Note that both the target policy and the world dynamics are deterministic, and therefore these states are visited equally often under the target policy. With probability ϵ, a random state s is sampled from the entire set of state space (there are 104 states in total). Given the sampled state s, with probability $1-\epsilon$, an action a is chosen to be the optimal policy's action at s, and with probability ϵ, a is chosen to be a random action. It is clear that when $\epsilon = 0$, $d_{\mu\pi}$ is the stationary state-action pair distribution under policy π and evaluating π using data sampled from $d_{\mu\pi}$ is an on-policy problem. As ϵ gets higher, $d_{\mu\pi}$ becomes more different from π's stationary state-action distribution and the learning problem becomes more off-policy. As required by Assumption 5.3, resulting states and rewards of state-action pairs sampled from $d_{\mu\pi}$ were generated by the transition function p, and next actions were chosen following the target policy. Note that in the tested domain, as long as $\epsilon > 0$, $d_{\mu\pi}(s,a) > 0, \forall s \in \mathcal{S}, a \in \mathcal{A}$. If $\epsilon = 0$, only state-action pairs visited by the target policy are sampled. The algorithm may still learn the reward rate and the differential action-value function for these pairs correctly, up to an additive constant. But there is no hope that other pairs are learned well. In experiments, ϵ was chosen from $\{0, 0.2, 0.4\}$. This finishes the description of the sampling process that generates transitions used by tested algorithms.

For each state-action pair (s,a), the corresponding feature vector $x(s,a)$ was constructed as follows. First, a state feature vector $x(s)$ was computed. Here,

$$x(s) \doteq \text{concatenate}(\phi(i), \phi(j)),$$

where i, j are the x, y coordinates of the cell corresponding to state s (e.g., the yellow cell at the upper left corner has coordinates $(1, 1)$ and the right hallway cell's coordinates are $(7, 9)$), and,

$$\begin{aligned}
\phi(0) &= [1, 0, 0, 0] \\
\phi(1) &= [0.75, 0.25, 0, 0] \\
\phi(2) &= [0.5, 0.5, 0, 0] \\
\phi(3) &= [0.25, 0.75, 0, 0] \\
\phi(4) &= [0, 1, 0, 0] \\
\phi(5) &= [0, 0.75, 0.25, 0] \\
\phi(6) &= [0, 0.5, 0.5, 0] \\
\phi(7) &= [0, 0.25, 0.75, 0] \\
\phi(8) &= [0, 0, 1, 0] \\
\phi(9) &= [0, 0, 0.75, 0.25] \\
\phi(10) &= [0, 0, 0.5, 0.5] \\
\phi(11) &= [0, 0, 0.25, 0.75] \\
\phi(12) &= [0, 0, 0, 1].
\end{aligned}$$

I then set $x(s, a)$'s elements ranging from the $(aK)^{\text{th}}$ one to the $((a+1)K-1)^{\text{th}}$ one with $x(s)$. All other elements in $x(s, a)$ were set to zero. In this way, generalization only happens across states but not across actions. Each feature vector has length $4 \times 2 \times 4 = 32$.

The reward rate error metric used in Section 3.4 is again used here. For Differential SGQ, Differential GQ1, and Differential GQ2, a relative value error metric was used to determine if the action values were properly learned. Note that the relative value error metric is not pertinent to GradientDICE because the algorithm does not estimate action values. The relative value error metric used here has been defined in Section 3.6. A reference state-action pair is needed in the relative value error metric. The reference state s_0 was chosen to be the yellow cell, just as in previous experiments. The reference action was chosen to be `down`, which is different from the previous choice `up`. This was indeed on purpose—experiments in this section tested different degrees of off-policyness, ranging from the on-policy case to a very off-policy case. In the on-policy case, the action `up` was never sampled from d_μ. Thus the value of action `up` was not learned at any state and was not suitable as a reference value.

The parameters were set as follows. For GradientDICE, tested choices of α were $\{0.1 \times 2^{-1}, 0.1 \times 2^{-3}, 0.1 \times 2^{-5}, 0.1 \times 2^{-7}, 0.1 \times 2^{-9}\}$. For the other three algorithms, tested choices of α were $\{2^{-1}, 2^{-3}, 2^{-5}, 2^{-7}, 2^{-9}\}$. Note that GradientDICE's step sizes were chosen to be only one-tenth of other algorithms' because GradientDICE diverges with a large step size. Differential SGQ, Differential GQ1, and Differential GQ2 all have a parameter η, which has been chosen from $\{10^0, 10^{-1}, 10^{-2}, 10^{-3}, 10^{-4}\}$. Differential GQ1, Differential GQ2, and GradientDICE all use a regularization parameter c to guarantee the uniqueness of their solutions. As discussed before, as long as this term is positive, convergence is always guaranteed for all three algorithms. Therefore, I fixed its value to be a small value 10^{-8}. Finally, GradientDICE uses a parameter λ, for which I tested $\{2^{-4}, 2^{-2}, 2^0, 2^2, 2^4\}$.

For each parameter setting, I performed 30 independent runs, each of which contains $50,000$ time steps. The left subfigure of Figure 5.2 shows learning curves of the four tested algorithms for the reward rate error when $\epsilon = 0.2$. Remember by construction, $\epsilon = 0.2$ results in slight off-policyness. All of the experiment results shown in this chapter use this ϵ. Experiment results in the more off-policy case ($\epsilon = 0.4$) and the on-policy case ($\epsilon = 0$) are deferred to Section B.2 because the same observations can be made for $\epsilon = 0.2$ and $\epsilon = 0.0$ or $\epsilon = 0.4$. Each learning curve is produced with the parameter setting that minimizes the reward rate error averaged over the last 5000 steps and averaged over the 30 runs. Therefore the learning curves show the algorithms' asymptotic performance. The shading region indicates one standard error. Similar learning curves for the relative value error are shown in the right subfigure of Figure 5.2.

First, none of them converge to zero errors eventually. This is unsurprising because they all use LFA. The learning curves show that the three Differential methods converge to points that have similar errors. GradientDICE's error at the end of training is much higher than the other three algorithms. Although GradientDICE may still improve its performance after $50,000$ steps, it is clear that the other three algorithms are faster. Also, note that the learning curves of the three differential methods are more stable than the learning curve of the GradientDICE method.

Figures 5.3–5.6 show sensitivity curves of the tested algorithms, respectively. As usual, each point in a sensitivity curve is an error (either reward rate error or relative value error) averaged over the entire $50,000$ training steps.

Several observations can be drawn from these curves. First, Figure 5.4 shows that Differential GQ1 was almost not sensitive to its parameter η and was only slightly sensitive to its step size α. Comparing Figure 5.4 and Figure 5.2 shows that Differential GQ1's reward rate and relative value errors were close to their asymptotic counterparts for appropriate choices of parameter setting, suggesting that the algorithm learned very fast with these parameters. Second, Figure 5.3 shows that Differential SGQ was more sensitive to η and to α. Just like Differential GQ1, Differential SGQ with appropriate parameter settings also had its average errors being close to the asymptotic errors. Third, the estimated reward rate of Differential GQ2 was sensitive to η while the estimated differential action-value function was not. This should not be surprising given that η only influenced the reward rate estimate. Unlike the other two differential methods, Differential GQ2 was not stable with a large step size. And just like the other two differential methods, GQ2 learned fast with the best parameter setting. Finally, GradientDICE generally performed badly in the experiment regardless of the chosen parameters. And even with the best parameter setting, GradientDICE's reward rate error was still much higher than the asymptotic reward rate error, suggesting that learning was pretty slow.

In summary, in my experiments, Differential GQ1 was not sensitive to its parameter η and was only slightly sensitive to its step-size parameter α. In addition, Differential GQ1 performed better than the three competitors in terms of speed of convergence and sensitivity to hyper-parameters.

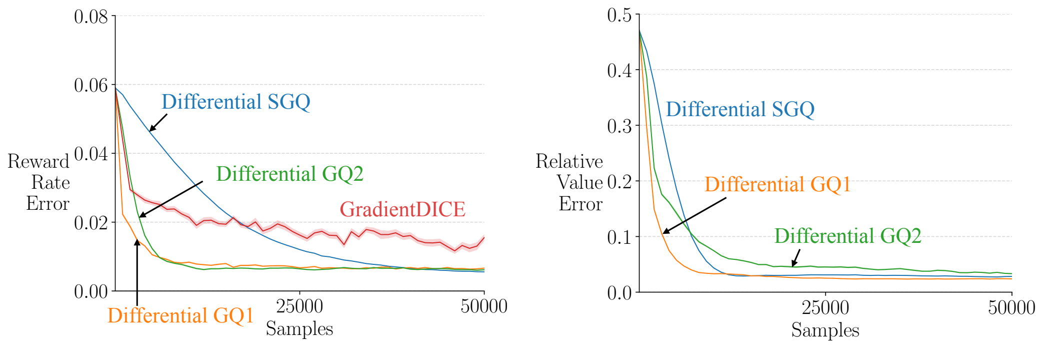


Figure 5.2: Learning curves of the four tested algorithms when $\epsilon = 0.2$. The parameter setting was chosen to minimize the error over the last 5000 steps. The axes have the same meaning as in Figure 3.2a and in Figure 3.2b.

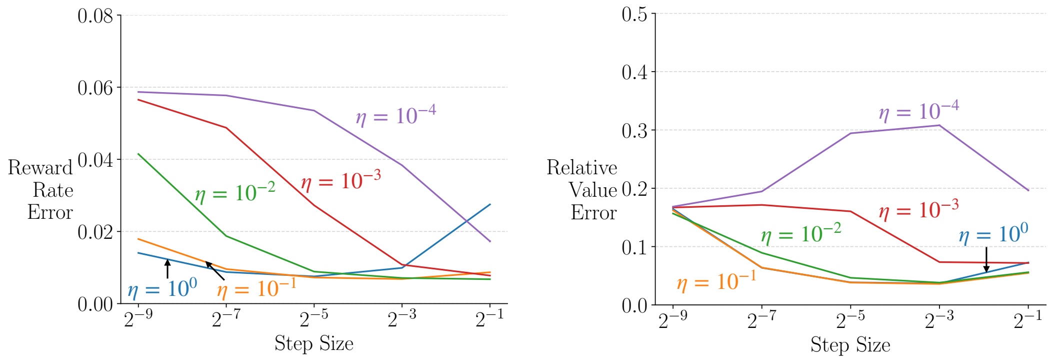


Figure 5.3: Sensitivity curves of Differential SGQ when $\epsilon = 0.2$. The parameter setting was chosen to minimize the error over the entire 50,000 steps. The axes have the same meaning as in Figure 3.2c and in Figure 3.2d.

5.6 Summary

This chapter developed an average-reward algorithm for off-policy prediction learning with function approximation and developed theories for the algorithm. Specifically, this chapter considered the i.i.d. learning setting, where the agent's experience consists of transitions sampled from a certain transition distribution, which can be empirically constructed by collecting a batch of data following the behavior policy. Under this learning setting, this chapter showed, using a counterexample, that Differential Semi-Gradient Q, which is a straightforward linear function approximation extension of Differential TD-learning, diverges. Inspired by the proven-convergent Gradient-TD family of

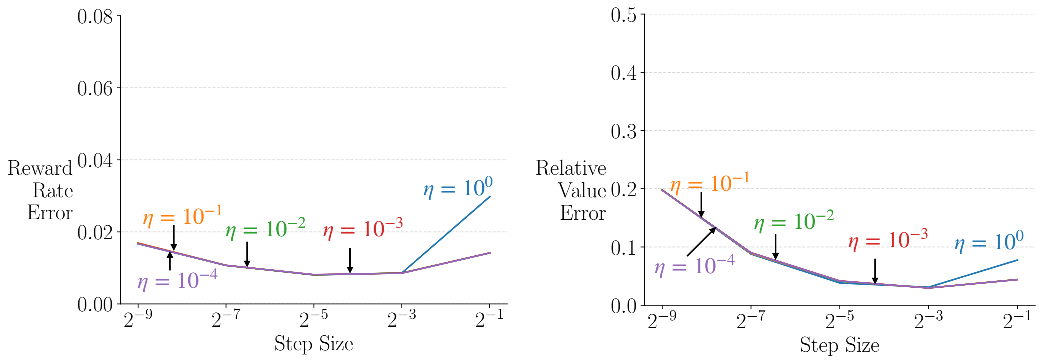


Figure 5.4: Sensitivity curves of Differential GQ1 when $\epsilon = 0.2$. The axes are the same as in Figure 5.3.

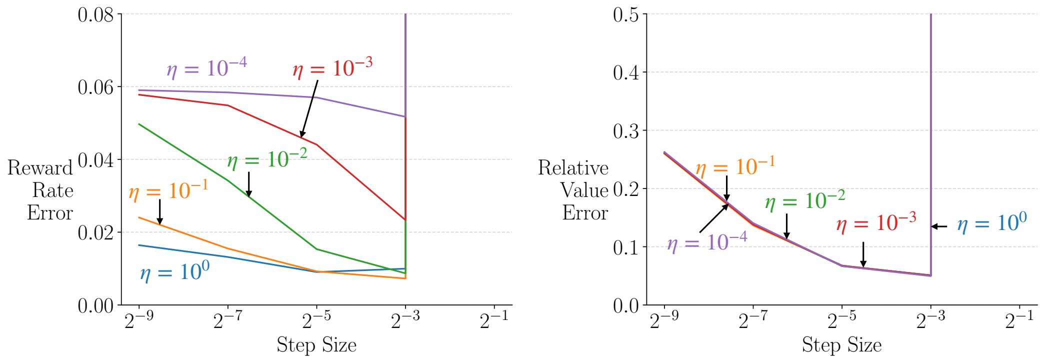


Figure 5.5: Sensitivity curves of Differential GQ2 when $\epsilon = 0.2$. The axes are the same as in Figure 5.3.

algorithms (Sutton et al. 2009), which were originally developed under the discounted setting, this chapter introduced the One-Stage Differential Gradient-Q (Or Differential GQ1) algorithm for the average-reward formulation. For this new algorithm, this chapter showed a convergence result of the new algorithm and presented an error bound of the point that the algorithm converges to. The convergence result only requires very standard assumptions and the proof builds upon the proof of the GTD2 algorithm shown by Sutton et al. (2009). The result characterizing the error bound requires more complicated assumptions. This bound, however, is the first bound for the TD fixed point (up to some regularization if necessary) in the off-policy setting concerning the average-reward formulation. The proof of the theorem extends Kolter's (2011) result from the discounted formulation to the average-reward formu-

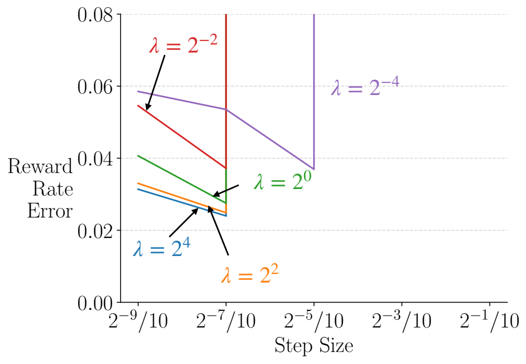


Figure 5.6: Sensitivity curves of GradientDICE when $\epsilon = 0.2$. The axes are the same as in the left subfigure of Figure 5.3.

lation. Finally, this chapter showed that, when the TD fixed point uniquely exists, by removing the regularization term in the algorithm, the Differential GQ1 algorithm converges to the TD fixed point.

This chapter also compared Differential GQ1 with Differential SGQ, its sibling algorithm Differential GQ2 and a competitive density-ratio-based algorithm, GradientDICE, in experiments. The empirical results demonstrated that Differential GQ1 performed better than the competitors in terms of speed of convergence and sensitivity to hyper-parameters in the tested domain.

Chapter 6

Discussion

In this last chapter, I would like to discuss the limitations of this dissertation. These limitations naturally lead to potential future work. I would like to close this chapter with a discussion of some final remarks on the average-reward formulation.

A major limitation of this dissertation is that it primarily focuses on the tabular setting and the FA setting is studied to a limited extent. Function approximation is necessary for large-scale applications and the larger ambitions of artificial intelligence. Chapter 3 and Chapter 4 were both developed in the tabular setting. Chapter 5 was developed under the linear function approximation setting but was limited to the prediction learning case. While it is straightforward to extend the tabular algorithms presented in Chapter 3 and Chapter 4 to the FA setting (see e.g, Appendix E in Wan, Naik, and Sutton (2021a)), the extensions may not be stable as suggested by Section 5.2. Chapter 5 provided a stable LFA prediction algorithm (i.e., Differential GQ1). A natural idea is to extend this algorithm to the control setting, potentially using the technique developed by Maei et al. (2010). It also seems possible to extend this algorithm to the non-linear FA setting, potentially using the technique developed by Maei et al. (2009) or the technique developed by Patterson, White, and White (2022). In addition to the gradient TD approach used to develop Differential GQ1, it is also possible to develop off-policy prediction algorithms following the emphatic approach (He, Wan, and Mahmood 2022), which was primarily studied concerning the discounted formulation (e.g., Sutton, Mahmood, and White 2016; He et al. 2023). Finally, in the FA setting considered here, the feature vector is a function of the current environment (MDP) state. In the real world, environment states are not observable, the agent must construct its own state using the history of observations. It is also possible to develop and analyze average-reward algorithms in this more general agent-state setting (see e.g., Dong, Van Roy, and Zhou 2022).

This dissertation only covers off-policy model-free algorithms and planning algorithms and does not cover two other classes of algorithms: on-policy model-free algorithms and model-based learning algorithms.

On-policy model-free learning algorithms (examples were provided in Chapter 1) need to specify how to choose actions and deal with the exploration-exploitation challenge while off-policy algorithms do not have to. For example, the Differential Q-learning algorithm introduced in Section 3.5 is an off-policy algorithm, and can be applied to a data stream generated by following any fixed policy that assigns a positive probability to each action. A natural question that arises is whether it is possible to develop an on-policy version of Differential Q-learning. Although the derivation of such an on-policy version is not immediately apparent, it does not seem implausible to achieve. In the context of the discounted setting, successful examples have been demonstrated for extending discounted Q-learning (Watkins and Dayan 1992) to on-policy versions (Strehl et al. 2006; Dong et al. 2019). This suggests that due to the strong connection between the discounted approach with a large discount factor and the average-reward formulations, it may be feasible to derive on-policy algorithms using similar techniques.

Model-based learning methods learn a world model and plan with it. These methods could possibly be more sample-efficient and adapt faster to non-stationary environments, as suggested by Wan et al. (2022). Nevertheless, there are also challenges that are unique to model-based methods, especially in the FA setting. For example, the design of the model with FA is not a stand-alone problem—it depends on and influences many other components including feature learning, planning, and option discovery. For more discussion about learning a model with FA, readers may refer to, for example, Sutton et al. (2012), Chua et al. (2018), Wan et al. (2018;2019), Kudashkina et al. (2021), Wan et al. (2022), Gottesman et al. (2019), Yu et al. (2020), and Kidambi et al. (2020).

The empirical studies of all of the proposed algorithms were mainly used to verify theories, even though some of them also helped gain insights that are unrelated to the theories. More extensive study is needed to understand the behaviors of the proposed algorithms, both in pedagogical small-scale problems and challenging large-scale problems.

All of the methods studied in this dissertation are one-step methods. Extensions of these methods to n-step, λ-return, or sophisticated eligibility-trace methods (e.g., van Seijen et al. 2016; Sutton and Barto 2018) are also possible. Recently, Naik and Sutton (2022) studied extensions of the Differential TD-learning algorithm to the multi-step and eligibility-trace settings.

This dissertation considers policies that result in the same average-reward rate as equally good. It is possible to seek more selective sets of policies, which are those achieving better short-term performance among all policies that achieve the optimal reward rate. These policies include bias-optimal policies and the more selective Blackwell optimal policies (Blackwell 1962). The biases (also known as differential value function) of these policies are the highest among all policies achieving the optimal reward rate (for more details, see, e.g. Chapter 10 by Puterman 1994). The Blackwell optimal policies are optimal for not only the average-reward formulation but also the discounted formulation for all discount factors that are close to one. Algorithms seeking these more selective policies have been quite limited. To my knowledge, all existing algorithms that guarantee to obtain such policies are planning algorithms (see Bibliographic Remark of Section 10 in Puterman 1994 for a list of them). Learning algorithms (e.g., Mahadevan 1996b; Dewanto and Gallagher 2021) have not been proven to produce such policies.

Chapter 4 considers average-reward sub-problems that involve temporal abstraction with options. While the options framework (Sutton, Precup, and Singh 1999) provides a simple and general way to formulate the idea of temporal abstraction, there also exist other ways such as Hierarchies of Abstract

Machines (Parr and Russell 1997) and MAXQ (Dietterich 2000). The average-reward formulation has been studied together with HAM by Ghavamzadeh and Mahadevan (2007).

Chapter 4 assumes that a fixed set of options is provided and the agent then learns or plans using them. One of the most important challenges in the options framework is the *discovery* of options. I think the discovery problem is orthogonal to the problem formulation. Hence, another line of future work is to extend existing option-discovery algorithms developed for the discounted and episodic formulations to the average-reward formulation (e.g., algorithms by Wan and Sutton 2022, McGovern and Barto 2001, Menache et al. 2002, Simsek and Barto 2004, Singh et al. 2004, Machado et al. 2017, Brunskill and Li 2014, Jinnai et al. 2019). Relatively more work might be required in extending approaches that couple the problems of option discovery and learning (e.g., Gregor et al. 2016, Bacon et al. 2017, Eysenbach et al. 2018, Achiam et al. 2018, Veeriah et al. 2021).

The Differential GQ1 algorithm (Chapter 5) needs to use i.i.d. samples. These samples can be obtained by sampling from a batch of data, which is potentially stored in a buffer. In large-scale problems, this batch learning approach may need a large buffer to store transitions, which requires a big memory space. Further, if the world is non-stationary, this approach would need to tackle the additional challenge of determining whether or not to remove an old transition in the buffer. Alternatively, the agent may follow an online learning approach, where the old transitions are discarded and only the current transition is used for learning. Transitions collected in this way are not i.i.d. Convergence of Differential GQ1 or its variant in this non-i.i.d. setting might be derived based on Yu's (2017) analysis.

I now discuss some final remarks concerning the average-reward formulation.

Firstly, it appears to me that some researchers in our field believe that the average-reward formulation only cares about the performance in the very far future and does not care about the near future. I hold a more optimistic view and there is a nice theoretical result that supports my view. Specifically, it has been shown in Proposition 6.4 by Meyn (2022) that, given any policy, the average of rewards observed in a sequence of n transitions can be lower bounded by the difference between the reward rate of the policy and a term that diminishes to zero very fast (in particular, with speed $1/n$, where n is the number of transitions). This result shows that the average-reward formulation also cares about the near future.

Secondly, it also appears to me that some researchers within our field believe that, in order to favor policies with high long-term performance, one does not have to apply the average-reward formulation. Instead, one could simply apply the discounted formulation with a large discount factor. In fact, there are several works showing the tight connection between the two formulations when the discount factor is large (Blackwell 1962; Kakade 2001; Tsitsiklis and Van Roy 2002; Devraj and Meyn 2021). Although this approach is doable, it has its own challenges, and this approach is probably not as easy as one would expect. Firstly, how large the discount factor needs to vary across different worlds and is thus not straightforward to choose without prior knowledge of the world. Furthermore, when this discount factor is large, the resulting discounted total reward could have an unduly large magnitude and varies significantly across different policies, making algorithms maximizing it prone to be unstable. Finally, it has been argued that the discounted formulation might be inappropriate for control problems with function approximation (Section 10, Sutton and Barto 2018). They have provided a result showing that the discount factor does not play a role in these problems for special-case MDPs. Specifically, varying this factor does not change the ranks of policies. I show that their result also holds for general MDPs in Chapter A.

Thirdly, the development of average-reward algorithms indeed requires a bit more imagination and creativity compared to episodic or discounted formulations. The average-reward formulation poses a unique challenge of estimating the reward rate, which is not present in episodic or discounted settings. To obtain the reward rate, the naive way is to simply estimate it using the average of all the observed rewards. But this approach can not produce the reward rate in the off-policy-learning or incremental-planning settings. We need alternative non-trivial ways to estimate the reward rate. In this dissertation, we have seen three such ways. The first way is to directly apply the TD error to update both the value estimate and the reward rate estimate as in our Differential Q-learning. The second way is to choose a reference state-action pair, whose associated action value will become the reward rate as in RVI Q-learning by Abounadi, Bertsekas, and Borkar (2001). And the third way uses an estimate of the density ratio between the stationary distributions of the behavior and target policies, as in GradientDICE by (Zhang et al. 2020b). It would be valuable to study the advantages and disadvantages of these three ways in theory and practice.

Fourthly, people usually find that the theoretical analyses of the average-reward algorithms are more challenging than those of the discounted or episodic algorithms. I think this is true. The average-reward algorithms need a different, more involved technical tool to analyze. For example, the technical tool used to analyze both Differential Q-learning and RVI Q-learning is more involved than the technical tool used to analyze discounted Q-learning and episodic Q-learning (Tsitsiklis 1994). The other example is the analysis for value iteration. The discounted and episodic value iteration algorithms (Bertsekas 2007) are much easier to analyze than the average-reward value iteration algorithm (Schweitzer and Federgruen 1977). One needs to consider the structure of Markov chains for the average-reward value iteration algorithm but not for the discounted or episodic algorithms. To advance the theory of average-reward algorithms further, we might still need to discover new theoretical properties of this class of algorithms or new technical tools.

Fifthly, it is a bit surprising to me that most of the RL empirical work has been on episodic problems, and empirical studies of continuing problems have been quite limited. Creating testbeds for continuing problems and benchmarking continuing algorithms hold immense value in our field, as it addresses a significant gap and helps advance research and development.

References

Abounadi, J., Bertsekas, D., Borkar, V. S. (2001). Learning algorithms for Markov decision processes with average cost. *SIAM Journal on Control and Optimization, 40*(3), 681–698.

Abbasi-Yadkori, Y., Bartlett, P., Bhatia, K., Lazic, N., Szepesvari, C., Weisz, G. (2019a). POLITEX: regret bounds for policy iteration using expert prediction. In *Proceedings of the 36th International Conference on Machine Learning*, 3692–3702.

Abbasi-Yadkori, Y., Lazic, N., Szepesvari, C., Weisz, G. (2019b). Exploration-enhanced POLITEX. *ArXiv:1908.10479.*

Achiam, J., Edwards, H., Amodei, D., Abbeel, P. (2018). Variational option discovery algorithms. *ArXiv:1807.10299.*

Almezel, S., Ansari, Q. H., Khamsi, M. A. (2014). *Topics in fixed point theory, 5.* Springer.

Auer, R., Ortner, P. (2006). Logarithmic online regret bounds for undiscounted reinforcement learning. In *Advances in Neural Information Processing Systems, 19*, 49–56.

Bacon, P. L., Harb, J., Precup, D. (2017). The Option-critic architecture. In *Proceedings of the 31st AAAI Conference on Artificial Intelligence*, 1726–1734.

Bartlett, P. L., Tewari, A. (2009). REGAL: a regularization based algorithm for reinforcement learning in weakly communicating MDPs. In *Proceedings of the 25th Conference on Uncertainty in Artificial Intelligence*, 35–42.

Barto, A. G., Sutton, R. S., Anderson, C. W. (1983). Neuronlike adaptive elements that can solve difficult learning control problems. *IEEE Transactions on Systems, Man, and Cybernetics*(5), 834–846.

Bellman, R. E. (1957). *Dynamic programming.* Princeton University Press.

Berner, C., Brockman, G., Chan, B., Cheung, V., Debiak, P., Dennison, C., Farhi, D., Fischer, Q., Hashme, S., Hesse, C., Józefowicz, J., Petrov, M., Pondé de Oliveira Pinto, H., Raiman, J., Salimans, T., Schlatter, J., Schneider, J., Sidor, S., Sutskever, I., Tang, J., Wolski, F., Zhang, S. (2019). Dota 2 with large scale deep reinforcement learning. *ArXiv:1912.06680*.

Bertsekas, D. P. (1998). A new value iteration method for the average cost dynamic programming problem. *SIAM Journal on Control and Optimization, 36*(2), 742–759.

Bertsekas, D. P. (2007). *Dynamic programming and optimal control* (3rd ed., Vol. II). Athena Scientific.

Bertsekas, D. P., Tsitsiklis, J. N. (1996). *Neuro-dynamic programming.* Athena Scientific.

Blackwell, D. (1962). Discrete dynamic programming. *The Annals of Mathematical Statistics*, 719–726.

Blum, J. R. (1954). Approximation methods which converge with probability one. *The Annals of Mathematical Statistics*, 382–386.

Borkar, V. S. (1997). Stochastic approximation with two time scales. *Systems & Control Letters, 29*(5), 291–294.

Borkar, V. S. (1998). Asynchronous stochastic approximations. *SIAM Journal on Control and Optimization, 36*(3), 840–851.

Borkar, Vivek S. (2000). Erratum: Asynchronous stochastic approximations. *SIAM Journal on Control and Optimization, 38*(2), 662–663.

Borkar, V. S. (2009). *Stochastic approximation: A dynamical systems viewpoint.* Springer.

Borkar, V. S., Soumyanatha, K. (1997). An analog scheme for fixed point computation. I. Theory. *IEEE Transactions on Circuits and Systems I: Fundamental Theory and Applications, 44*(4), 351–355.

Borkar, V. S., Meyn, S. P. (2000). The ODE method for convergence of stochastic approximation and reinforcement learning. *SIAM Journal on Control and Optimization, 38*(2):447–469.

Brafman, R. I., Tennenholtz, M. (2002). R-MAX — a general polynomial time algorithm for near-optimal reinforcement learning. *Journal of Machine Learning Research, 3*(10), 213–231.

Brunskill, E., Li, L. (2014). PAC-inspired option discovery in lifelong reinforcement learning. In *Proceedings of the 31st International Conference on Machine Learning*, 316–324.

Chua, K., Calandra, R., McAllister, R., Levine, S. (2018). Deep reinforcement learning in a handful of trials using probabilistic dynamics models. In *Advances in Neural Information Processing Systems, 31*, 4759–4770.

Das, T. K., Gosavi, A., Mahadevan, S. Marchalleck, N. (1999). Solving semi-Markov decision problems using average reward reinforcement learning. *Management Science, 45*(4), 560–574.

Denardo, Eric V. (1971). Markov renewal programs with small interest rates. *The Annals of Mathematical Statistics 42*(2), 477–496.

Diddigi, R. B., Kamanchi, C., Bhatnagar, S. (2020). A convergent off-policy temporal difference algorithm. In *Proceedings of the 24th European Conference on Artificial Intelligence*, 1103–1110.

Dietterich, T. G. (2000). Hierarchical reinforcement learning with the MAXQ value function decomposition. *Journal of Artificial Intelligence Research, 13*, 227–303.

Dewanto, V., Dunn, G., Eshragh, A., Gallagher, M., Roosta, F. (2020). Average-reward model-free reinforcement learning: a systematic review and literature mapping. *ArXiv:2010.08920.*

Dewanto, V., Gallagher, M. (2021). A nearly Blackwell-optimal policy gradient method. *ArXiv:2105.13609.*

Devraj, A. M., Meyn, S. P. (2021). Q-learning with uniformly bounded variance. *IEEE Transactions on Automatic Control.*

Dong, K., Wang, Y., Chen, X., Wang, L. (2019). Q-learning with ucb exploration is sample efficient for infinite-horizon mdp. *ArXiv:1901.09311.*

Dong, S., Van Roy, B., Zhou, Z. (2022). Simple agent, complex environment: Efficient reinforcement learning with agent states. *Journal of Machine Learning Research, 23*(255), 1–54.

Du, S. S., Chen, J., Li, L., Xiao, L., Zhou, D. (2017). Stochastic variance reduction methods for policy evaluation. In *Proceedings of the 34th International Conference on Machine Learning*, 1049–1058.

Eysenbach, B., Gupta, A., Ibarz, J., Levine, S. (2018). Diversity is all you need: Learning skills without a reward function. In *Proceedings of the 7th International Conference on Learning Representations.*

Ghavamzadeh, M., Mahadevan, S. (2007). Hierarchical average reward reinforcement learning. *Journal of Machine Learning Research, 8*(11).

Gosavi, A. (2004). Reinforcement learning for long-run average cost. *European Journal of Operational Research, 155*(3), 654–674.

Gottesman, O., Liu, Y., Sussex, S., Brunskill, E., Doshi-Velez, F. (2019). Combining parametric and nonparametric models for off-policy evaluation. In *Proceedings of the 36th International Conference on Machine Learning*, 2366–2375.

Gregor, K., Rezende, D. J., Wierstra, D. (2016). Variational intrinsic control. *ArXiv:1611.07507.*

Hao, B., Lazic, N., Abbasi-Yadkori, Y., Joulani, P., Szepesvari, C. (2021). Adaptive approximate policy iteration. In *Proceedings of the 24th International Conference on Artificial Intelligence and Statistics*, 523–531.

He, J., Wan, Y., Mahmood, R. (2022). The emphatic approach to average-reward policy evaluation. In *NeurIPS 2022 Workshop on DeepRL.*

He, J., Che, F., Wan, Y., Mahmood, R. (2023), Consistent Emphatic Temporal-Difference Learning. Accepted by *the 39th Conference on Uncertainty in Artificial Intelligence.*

Howard, R. A. (1960). *Dynamic programming and Markov processes.* MIT Press.

Jalali, A., Ferguson, M. J. (1989). Computationally efficient adaptive control algorithms for Markov chains. In *Proceedings of the 28th IEEE Conference on Decision and Control*, 1283–1288.

Jalali, A., Ferguson, M. J. (1990). A distributed asynchronous algorithm for expected average cost dynamic programming. In *Proceedings of the 29th IEEE Conference on Decision and Control*, 1394–1395.

Jaksch, T., Ortner, R., Auer, P. (2010). Near-optimal regret bounds for reinforcement learning. *Journal of Machine Learning Research, 11*(4), 1563–1600.

Jin, Y., Sidford, A. (2020). Efficiently solving MDPs with stochastic mirror descent. In *Proceedings of the 37th International Conference on Machine Learning*, 4890–4900.

Jinnai, Y., Abel, D., Hershkowitz, D., Littman, M., Konidaris, G. (2019). Finding options that minimize planning time. In *Proceedings of the 36th International Conference on Machine Learning*, 3120–3129.

Kakade, S. (2001). Optimizing average reward using discounted rewards. In *International Conference on Computational Learning Theory*, 605–615.

Kearns, M., Singh, S. (2002). Near-optimal reinforcement learning in polynomial time. *Machine Learning, 49*(2), 209–232.

Kidambi, R., Rajeswaran, A., Netrapalli, P., Joachims, T. (2020). Morel: Model-based offline reinforcement learning. In *Advances in Neural Information Processing Systems, 33*, 21810–21823.

Kim, H., Jordan, M., Sastry, S., Ng, A. (2003). Autonomous helicopter flight via reinforcement learning. Advances in *Neural Information Processing Systems, 16*, 799-806.

Kolter, J. (2011). The fixed points of off-policy TD. In *Advances in Neural Information Processing Systems, 24*, 2169–2177.

Konda, V. R., (2002). *Actor-critic algorithms.* Doctoral dissertation, MIT.

Kudashkina, K., Wan, Y., Naik, A., Sutton, R. S. (2021). Planning with expectation models for control. *ArXiv:2104.08543*.

Lawler, G. F. (2018). *Introduction to stochastic processes.* Chapman and Hall/CRC.

Lazic, N., Boutilier, C., Lu, T., Wong, E., Roy, B., Ryu, M. K., Imwalle, G. (2018). Data center cooling using model-predictive control. In *Advances in Neural Information Processing Systems, 31*.

Lazic, N., Yin, D., Farajtabar, M., Levine, N., Gorur, D., Harris, C., Schuurmans, D. (2020). A maximum-entropy approach to off-policy evaluation in average-reward mdps. In *Advances in Neural Information Processing Systems, 33*, 12461–12471.

Li, Y., Cao, F. (2010). RVI Reinforcement learning for semi-Markov decision processes with average reward. In *Proceedings of the 8th World Congress on Intelligent Control and Automation*, 1674–1679.

Liu, B., Liu, J., Ghavamzadeh, M., Mahadevan, S., Petrik, M. (2015). Finite-sample analysis of proximal gradient TD algorithms. In *Proceedings of the 31st Conference on Uncertainty in Artificial Intelligence*, 504–513.

Liu, Q., Li, L., Tang, Z., Zhou, D. (2018). Breaking the curse of horizon: Infinite-horizon off-policy estimation. In *Advances in Neural Information Processing Systems, 31*, 5361–5371.

Machado, M. C., Bellemare, M. G., Bowling, M. (2017). A Laplacian framework for option discovery in reinforcement learning. In *Proceedings of the 34th International Conference on Machine Learning*, 2295–2304.

Maei, H., Szepesvari, C., Bhatnagar, S., Precup, D., Silver, D., Sutton, R. S. (2009). Convergent temporal-difference learning with arbitrary smooth function approximation. In *Advances in neural information processing systems, 22*, 1204–1212.

Maei, H. R., Szepesvari, C., Bhatnagar, S., Sutton, R. S. (2010). Toward off-policy learning control with function approximation. In *Proceedings of the 27th International Conference on Machine Learning*, 719–726.

Mahadevan, S. (1996a). Average reward reinforcement learning: Foundations, algorithms, and empirical results. *Machine Learning, 22*(1–3), 159–195.

Mahadevan, S. (1996b). An average-reward reinforcement learning algorithm for computing bias-optimal policies. In *Proceedings of the 13th National Conference on Artificial Intelligence*, 875–880.

Mahadevan, S., Liu, B., Thomas, P., Dabney, W., Giguere, S., Jacek, N., Gemp, I., Liu, J. (2014) Proximal reinforcement learning: A new theory of sequential decision making in primal-dual spaces. *ArXiv:1405.6757*.

Marbach, P., Mihatsch, O., Tsitsiklis, J. N. (2000). Call admission control and routing in integrated services networks using neuro-dynamic programming. *IEEE Journal on Selected Areas in Communications, 18*(2), 197–208.

Marbach, P., Tsitsiklis, J. N. (2001). Simulation-based optimization of Markov reward processes. *IEEE Transactions on Automatic Control, 46*(2), 191–209.

McCarthy, J. 2007. What is Artificial Intelligence? Available electronically at http://www-formal.stanford.edu/jmc/whatisai/whatisai.html

McGovern, A., Barto, A. G. (2001). Automatic discovery of subgoals in reinforcement learning using diverse density. In *Proceedings of the 18th International Conference on Machine Learning*, 361–368.

Menache, I., Mannor, S., Shimkin, N. (2002). Q-cut—dynamic discovery of sub-goals in reinforcement learning. In *Proceedings of the 13th European Conference on Machine Learning*, 295–306.

Meyn, S. (2022). *Control systems and reinforcement learning.* Cambridge University Press.

Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., Graves, A., Riedmiller, M., Fidjeland, A. K., Ostrovski, G., Petersen, S. , Beattie, C. , Sadik, A. , Antonoglou, I. , King, H. , Kumaran, D. , Wierstra, D. , Legg, S. , Hassabis, D. (2015). Human-level control through deep reinforcement learning. *Nature 518*(7540), 529–533.

Mousavi, A., Li, L., Liu, Q., Zhou, D. (2020). Black-box off-policy estimation for infinite-horizon reinforcement learning. In *Proceedings of the 8th International Conference on Learning Representations.*

Naik, A., Sutton, R. S. Multi-step average-reward prediction via differential TD (λ) (2022). In *Conference on Reinforcement Learning and Decision Making (RLDM).*

OpenAI (2023). GPT-4 Technical Report. *ArXiv:2303.08774.*

Parr, R., Russell, S. (1997). Reinforcement learning with hierarchies of machines. In *Advances in Neural Information Processing Systems, 10*, 1043–1049.

Pateria, S., Subagdja, B., Tan, A., Quek, C. (2021). Hierarchical reinforcement learning: A comprehensive survey. *ACM Computing Surveys, 54*(5), 1–35.

Patterson, A., White, A., White, M. (2022). A generalized projected bellman error for off-policy value estimation in reinforcement learning. The *Journal of Machine Learning Research, 23*(1), 6463–6523.

Puterman, M. L. (1994). *Markov decision processes: Discrete stochastic dynamic programming.* John Wiley & Sons.

Ren, Z., Krogh, B. H. (2001). Adaptive control of Markov chains with average cost. *IEEE Transactions on Automatic Control, 46*(4), 613–617.

Schulman, J., Wolski, F., Dhariwal, P., Radford, A., Klimov, O. (2017). Proximal policy optimization algorithms. *ArXiv:1707.06347.*

Schwartz, A. (1993). A reinforcement learning method for maximizing undiscounted rewards. In *Proceedings of the 10th International Conference on Machine Learning*, 298–305.

Schweitzer, P. J. (1971). Iterative solution of the functional equations of undiscounted Markov renewal programming. *Journal of Mathematical Analysis and Applications, 34*(3), 495–501.

Schweitzer, P. J., Federgruen, A. (1977). The asymptotic behavior of undiscounted value iteration in Markov decision problems. *Mathematics of Op-*

erations Research, 2(4), 360–381.

Schweitzer, P. J., Federgruen, A. (1978). The functional equations of undiscounted Markov renewal programming. *Mathematics of Operations Research, 3*(4), 308–321.

Silver, D., Huang, A., Maddison, C. J., Guez, A., Sifre, L., van den Driessche, G., Schrittwieser, J., Antonoglou, I., Panneershelvam, V., Lanctot, M., Dieleman, S., Grewe, D., Nham, J., Kalchbrenner, N., Sutskever, I., Lillicrap, T., Leach, M., Kavukcuoglu, K., Graepel, T., Hassabis, D. (2016). Mastering the game of Go with deep neural networks and tree search. *Nature, 529*(7587), 484-–489.

Simsek, O., Barto, A. G. (2004). Using relative novelty to identify useful temporal abstractions in reinforcement learning. In *Proceedings of the 21st International Conference on Machine Learning.*

Singh, S. P. (1994). Reinforcement learning algorithms for average-payoff Markovian decision processes. In *Proceedings of the 12th AAAI Conference on Artificial Intelligence*, 700–705.

Singh, S., Barto, A. G., Chentanez, N. (2004). Intrinsically motivated reinforcement learning. In *Advances in Neural Information Processing Systems 17*, 1281–1288.

Strehl, A. L., Li, L., Wiewiora, E., Langford, J., Littman, M. L. (2006). PAC model-free reinforcement learning. In *Proceedings of the 23rd International Conference on Machine learning*, 881–888.

Sutton, R. S. (1988). Learning to predict by the methods of temporal differences. *Machine learning, 3*(1), 9–44.

Sutton, R. S. (1990). Integrated architectures for learning, planning, and reacting based on approximating dynamic programming. In *Proceedings of the 7th International Conference on Machine Learning*, 216–224.

Sutton, R. S., Precup, D., Singh, S. (1999). Between MDPs and semi-MDPs: A framework for temporal abstraction in reinforcement learning. *Artificial*

Intelligence, 112(1–2), 181–211.

Sutton, R. S., Barto, A. G. (2018). *Reinforcement learning: An introduction.* MIT Press.

Sutton, R. S., Maei, H. R., Precup, D., Bhatnagar, S., Silver, D., Szepesvari, C., Wiewiora, E. (2009). Fast gradient-descent methods for temporal-difference learning with linear function approximation. In *Proceedings of the 26th International Conference on Machine Learning*, 993–1000.

Sutton, R. S., Szepesvari, C., Geramifard, A., Bowling, M. P. (2012). Dyna-style planning with linear function approximation and prioritized sweeping. In *Proceedings of the 24th Conference on Uncertainty in Artificial Intelligence*, 528–536.

Sutton, R. S., Mahmood, A. R., White, M. (2016). An emphatic approach to the problem of off-policy temporal-difference learning. *The Journal of Machine Learning Research, 17*(1), 2603–2631.

Sutton, R. S., Modayil, J., Delp, M., Degris, T., Pilarski, P. M., White, A., Precup, D. (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*, 761–768.

Szepesvari, C. (2010). Algorithms for reinforcement learning. *Synthesis Lectures on Artificial Intelligence and Machine Learning*, Morgan and Claypool Publishers.

Tang, Z., Feng, Y., Li, L., Zhou, D., Liu, Q. (2019). Doubly robust bias reduction in infinite horizon off-policy estimation. In *Proceedings of the 8th International Conference on Learning Representations.*

Tsitsiklis, J. N. (1994). Asynchronous stochastic approximation and Q-learning. *Machine Learning, 16*(3), 185–202.

Tsitsiklis, J. N., Van Roy, B. (1999). Average cost temporal-difference learning. *Automatica, 35*(11), 1799–1808.

Tsitsiklis, J. N., Van Roy, B. (2002). On average versus discounted reward temporal-difference learning. *Machine Learning, 49*(2), 179–191.

van Seijen, H., Mahmood, A. R., Pilarski, P. M., Machado, M. C., Sutton, R. S. (2016). True online temporal-difference learning. *Journal of Machine Learning Research, 17*(145), 1–40.

Veeriah, V., Zahavy, T., Hessel, M., Xu, Z., Oh, J., Kemaev, I., van Hasselt, H., Silver, D., Singh S. (2021). Discovery of options via meta-learned subgoals. In *Advances in Neural Information Processing Systems 34*, 29861–29873.

Vien, N. A., Chung, T. (2008). Policy gradient semi-Markov decision process. In *Proceedings of the 20th IEEE International Conference on Tools with Artificial Intelligence, 2*, 11–18.

Wan, Y., Zaheer, M., White, M., Sutton, R. S. (2018). Model-based reinforcement learning with non-linear expectation models and stochastic environments. In *The Joint IJCAI/ECAI/AAMAS/ICML Conference Workshop on Prediction and Generative Modeling in Reinforcement Learning.*

Wan, Y., Abbas, Z., White, A., White, M., Sutton, R. S. (2019). Planning with expectation models. In *Proceedings of the 28th International Joint Conference on Artificial Intelligence*, 3649–3655.

Wan, Y., Naik, A., Sutton, R. S. (2021a). Learning and planning in average-reward Markov decision processes. In *Proceedings of the 38th International Conference on Machine Learning*, 10653–10662.

Wan, Y., Naik, A., Sutton, R. S. (2021b). Average-reward learning and planning with options. In *Advances in Neural Information Processing Systems, 34*, 22758–22769.

Wan, Y., Sutton, R. S. (2022). Toward discovering options that achieve faster planning. *ArXiv:2205.12515.*

Wan, Y., Yu, H., Sutton, R. S. (2023). On convergence of average-reward off-policy control algorithms in weakly communicating MDPs. *To Be Submitted. An Earlier Version in ArXiv:2209.15141.*

Wan, Y., Rahimi-Kalahroudi, A., Rajendran, J., Momennejad, I., Chandar, S., van Seijen, H. (2022). Towards evaluating adaptivity of model-based reinforcement learning methods. In *Proceedings of the 39th International Conference on Machine Learning*, 22536–22561.

Wang, M. (2017). Primal-dual π learning: Sample complexity and sublinear run time for ergodic Markov decision problems. *ArXiv:1710.06100.*

Warlop, R., Lazaric, A., Mary, J. (2018). Fighting boredom in recommender systems with linear reinforcement learning. In *Advances in Neural Information Processing Systems, 31*, 1757–1768.

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

Wei, C. Y., Jahromi, M. J., Luo, H., Sharma, H., Jain, R. (2020). Model-free reinforcement learning in infinite-horizon average-reward markov decision processes. In *Proceedings of the 37th International Conference on Machine Learning*, 10170–10180.

Wen, J., Dai, B., Li, L., Schuurmans, D. (2020). Batch stationary distribution estimation. In *Proceedings of the 37th International Conference on Machine Learning*, 10203–10213.

Wheeler, R., Narendra, K. (1986). Decentralized learning in finite Markov chains. *IEEE Transactions on Automatic Control, 31*(6), 519–526.

White, D. J. (1963). Dynamic programming, Markov chains, and the method of successive approximations. *Journal of Mathematical Analysis and Applications, 6*(3), 373–376.

Yang, S., Gao, Y., An, B., Wang, H., Chen, X. (2016). Efficient average reward reinforcement learning using constant shifting values. In *Proceedings of the 30th AAAI Conference on Artificial Intelligence*, 2258–2264.

Yu, H. (2017). On convergence of some gradient-based temporal-differences algorithms for off-policy learning. *ArXiv:1712.09652.*

Yu, H., Bertsekas, D. P. (2008). New error bounds for approximations from projected linear equations. In *European Workshop on Reinforcement Learning*, 253–267.

Yu, H., Bertsekas, D. P. (2009). Convergence results for some temporal difference methods based on least squares. *IEEE Transactions on Automatic Control, 54*(7), 1515–1531.

Yu, T., Thomas, G., Yu, L., Ermon, S., Zou, J. Y., Levine, S., Finn, C., Ma, T. (2020). Mopo: Model-based offline policy optimization. In *Advances in Neural Information Processing Systems, 33*, 14129–14142.

Zhang, R., Dai, B., Li, L., Schuurmans, D. (2020a). GenDICE: Generalized offline estimation of stationary values. In *Proceedings of the 8th International Conference on Learning Representations.*

Zhang, S., Liu, B., Whiteson, S. (2020b). GradientDICE: Rethinking generalized offline estimation of stationary values. In *Proceedings of the 37th International Conference on Machine Learning*, 11194–11203.

Zhang, S., Wan, Y., Sutton, R. S., Whiteson, S. (2021). Average-reward off-policy policy evaluation with function approximation. In *Proceedings of the 37th International Conference on Machine Learning*, 12578–12588.

Zhang, S., Zhang, Z., Maguluri, S. T. (2021). Finite Sample Analysis of Average-Reward TD Learning and Q-Learning. *Advances in Neural Information Processing Systems, 34*, 1230–1242.

Appendix A

The Discounted Factor Deprecates with Function Approximation

It has been shown (Section 10.4 in Sutton & Barto 2018) that, with the discounted formulation, the discounted factor deprecates in control problems with function approximation. Their result was developed for ergodic finite MDPs. In this section, I show that their result applies to general finite MDPs.

The discounted formulation of MDPs seeks a policy that maximizes the discounted total reward, or the *discounted return*, with the discount factor given as a problem parameter. The discounted discounted return is $R_1 + \gamma R_2 + \cdots$, where $0 \leq \gamma < 1$ is the discount factor. Define the value of a state s given a policy π to be the expected discounted return:

$$v_\pi^\gamma(s) \doteq \mathbb{E}_\pi[R_1 + \gamma R_2 + \cdots \mid S_0 = s].$$

The discounted formulation seeks a policy $\pi \in \Pi$ that maximizes the value for all states. Such a π always exists.

It is well-known that for both the discounted and the average-reward formulations, there exists a deterministic optimal policy. It is therefore common for the agent to maintain a $\mathcal{S} \times \mathcal{A}$ table that can represent every deterministic policy. However, if the state and action spaces are large, such a table would be large and the agent would have to resort to some function approximation (FA) that only represents a much smaller subset of Π.

When policies are represented by function approximation, both the discounted and the average-reward formulations need to be modified because there may not be a representable policy that maximizes expected return/discounted return/reward rate for each state and there must be a different weight for different states.

If the weights are set to be the start state distribution, the best policy would then be optimized for the start states and can be useless for other states. Such a policy performs well in the early stages but can become useless eventually if the start states are not visited again. Such a policy is not desirable in continuing problems because the agent is expected to perform well continuously, not only in the early stages. A more reasonable way is to weigh states according to the long-run fraction of time that the agent spends in each state. The rigorous definition of such a fraction for policy π is

$$d_\pi(s) \doteq \lim_{n\to\infty} \frac{1}{n} \sum_{t=0}^{n-1} \Pr(S_t = s \mid S_0 \sim d_0, A_{0:t-1} \sim \pi), \tag{A.1}$$

where the limit always exists. This limit is not necessarily the limiting distribution of π with initial distribution d_0. The limiting distribution does not necessarily exist. The limit is one stationary distribution of π (note that a policy may have multiple stationary distributions).

The discounted formulation is modified to

$$\sum_s d_\pi(s)\mathbb{E}_\pi[R_1 + \gamma R_2 + \cdots \mid S_0 = s]. \tag{A.2}$$

The average-reward formulation is thus modified to

$$\sum_s d_\pi(s) r(\pi, s). \tag{A.3}$$

The following proposition shows that maximizing the average-reward formulation (A.3) is equivalent to maximizing the discounted formulation (A.2). This proposition extends what is presented in the box on Page 254 by Sutton & Barto (2018) by considering general MDPs rather than ergodic MDPs.

Proposition A.1. *For any $\pi \in \Pi$ and $s \in \mathcal{S}$,*

$$\sum_s d_\pi(s) r(\pi, s) = \sum_s d_0(s) r(\pi, s),$$
$$\frac{1}{1-\gamma} \sum_s d_0(s) r(\pi, s) = \sum_s d_\pi(s) v_\pi^\gamma(s).$$

Proof.

$$\begin{aligned}
\sum_s d_\pi(s) r(\pi, s) &= \sum_s d_0(s) \sum_{s'} \lim_{n\to\infty} \frac{1}{n} \sum_{t=0}^{n-1} \Pr(S_t = s' \mid S_0 = s, \pi) r(\pi, s') \\
&= \sum_s d_0(s) \lim_{n\to\infty} \frac{1}{n} \sum_{t=0}^{n-1} \sum_{s'} \Pr(S_t = s' \mid S_0 = s, \pi) r(\pi, s') \\
&= \sum_s d_0(s) \lim_{n\to\infty} \frac{1}{n} \sum_{t=0}^{n-1} r(\pi, s) \\
&= \sum_s d_0(s) r(\pi, s)
\end{aligned}$$

The third equation holds because

$$\begin{aligned}
&\sum_{s'} \Pr(S_t = s' \mid S_0 = s, \pi) r(\pi, s') \\
&= \sum_{s'} \Pr(S_t = s' \mid S_0 = s, \pi) \lim_{n\to\infty} \frac{1}{n} \sum_{i=t+1}^{n+t} \mathbb{E}[R_i \mid S_t = s', \pi] \\
&= \lim_{n\to\infty} \frac{1}{n} \sum_{i=t+1}^{n+t} \mathbb{E}[R_i \mid S_0 = s, \pi] \\
&= \lim_{n\to\infty} \frac{1}{n} \left(\sum_{i=1}^{n+t} \mathbb{E}[R_i \mid S_0 = s, \pi] - \sum_{i=1}^{t} \mathbb{E}[R_i \mid S_0 = s, \pi] \right) \\
&= r(\pi, s).
\end{aligned}$$

$$
\begin{aligned}
&\sum_s d_\pi(s) v_\pi^\gamma(s) \\
&= \sum_s d_\pi(s) \mathbb{E}_\pi[R_1 + \gamma R_2 + \cdots \mid S_0 = s] \\
&= \sum_s d_\pi(s) \mathbb{E}_\pi[R_1 \mid S_0 = s] + \gamma \sum_s d_\pi(s) \mathbb{E}_\pi[R_2 \mid S_0 = s] + \cdots \\
&= \sum_s d_0(s) r(\pi, s) + \gamma \sum_s d_\pi(s) \mathbb{E}_\pi[R_2 \mid S_0 = s] + \cdots \\
&= \sum_s d_0(s) r(\pi, s) + \gamma \sum_s d_0(s) r(\pi, s) + \cdots \\
&= \frac{1}{1-\gamma} \sum_s d_0(s) r(\pi, s).
\end{aligned}
$$

The third equation holds from the definition of d_π and $r(\pi, s)$:

$$
\begin{aligned}
&\sum_s d_\pi(s) \mathbb{E}_\pi[R_1 \mid S_0 = s] \\
&= \sum_s \lim_{n\to\infty} \frac{1}{n} \sum_{t=0}^{n-1} \Pr(S_t = s \mid S_0 \sim d_0, \pi) r_\pi(s) \\
&= \lim_{n\to\infty} \frac{1}{n} \sum_{t=0}^{n-1} \mathbb{E}[R_{t+1} \mid S_0 \sim d_0, \pi] \\
&= \sum_s d_0(s) r(\pi, s).
\end{aligned}
$$

Here $r_\pi(s) \doteq \sum_a \pi(a \mid s) \sum_{s',r} p(s', r \mid s, a) r$ is the one-step expected reward under policy π.

The fourth equation holds for a similar reason

$$
\begin{aligned}
&\sum_s d_\pi(s)\mathbb{E}_\pi[R_2 \mid S_0 = s] \\
&= \sum_s \lim_{n\to\infty} \frac{1}{n} \sum_{t=0}^{n-1} \Pr(S_t = s \mid S_0 \sim d_0, \pi) \sum_a \pi(a \mid s) \sum_{s',r} p(s', r \mid s, a) r_\pi(s') \\
&= \lim_{n\to\infty} \frac{1}{n} \sum_{t=0}^{n-1} \sum_{s'} \Pr(S_{t+1} = s' \mid S_0 \sim d_0, \pi) r_\pi(s') \\
&= \lim_{n\to\infty} \frac{1}{n} \sum_{t=0}^{n-1} \mathbb{E}[R_{t+2} \mid S_0 \sim d_0, \pi] \\
&= \lim_{n\to\infty} \frac{1}{n} \sum_{t=0}^{n-1} \mathbb{E}[R_{t+1} \mid S_0 \sim d_0, \pi] - \lim_{n\to\infty} \frac{1}{n}\mathbb{E}[R_1 \mid S_0 \sim d_0, \pi] \\
&+ \lim_{n\to\infty} \frac{1}{n}\mathbb{E}[R_{n+1} \mid S_0 \sim d_0, \pi] \\
&= \sum_s d_0(s) r(\pi, s).
\end{aligned}
$$

□

The above proposition suggests that γ is not playing a role in ranking policies and is thus unnecessary. Therefore, the discount factor deprecates, and the discounted formulation is equivalent to the average-reward formulation when function approximation is used to represent policies. One might wonder why not optimize the discounted formulation anyway with an arbitrary discount factor, given that it is equivalent to the average-reward formulation. Could that be easier than the average-reward formulation? It turns out that no known algorithm maximizes the discounted formulation with different states weighted by the stationary distribution. If there is such an algorithm then its choice of the discount factor should not influence its solutions. All existing algorithms that I know of do not have this property.

Appendix B

Additional Experiments

B.1 Options Experiments

Section 4.6 shows sensitivity curves of inter-option prediction and control algorithms in the four-room domain with the second step size parameter $\beta = 2^{-1}$. This section shows the same results but with other choices of β $(2^{-3}, 2^{-5}, 2^{-7}, 2^{-9})$. Figure B.1 and Figure B.2 show sensitivity curves of the prediction and the control algorithms, respectively. These sensitivity curves are similar to those with $\beta = 2^{-1}$, suggesting that the algorithm is not sensitive to the choice of β.

B.2 Function Approximation Experiments

In this section, I present additional experiment results for the experiment described in Section 5.5. In Section 5.5, I presented learning curves and sensitivities curves for Differential SGQ, Differential GQ1, Differential GQ2, and GradientDICE when a parameter controlling the off-policyness of the problem $\epsilon = 0.2$. In this section, I show results with two other choices of ϵs. The first choice of ϵ is zero, in which case the problem becomes an on-policy one. The second choice of ϵ is 0.4, in which case the behavior policy is more different from the target policy than the case when ϵ is 0.2. The most extreme case is $\epsilon = 1$. In this case, state-action pairs are randomly sampled from the entire state-action space, whose size is $104 \times 4 = 416$, while the target policy is optimal and only visits 17 state-action pairs equally often. Figure B.3 and

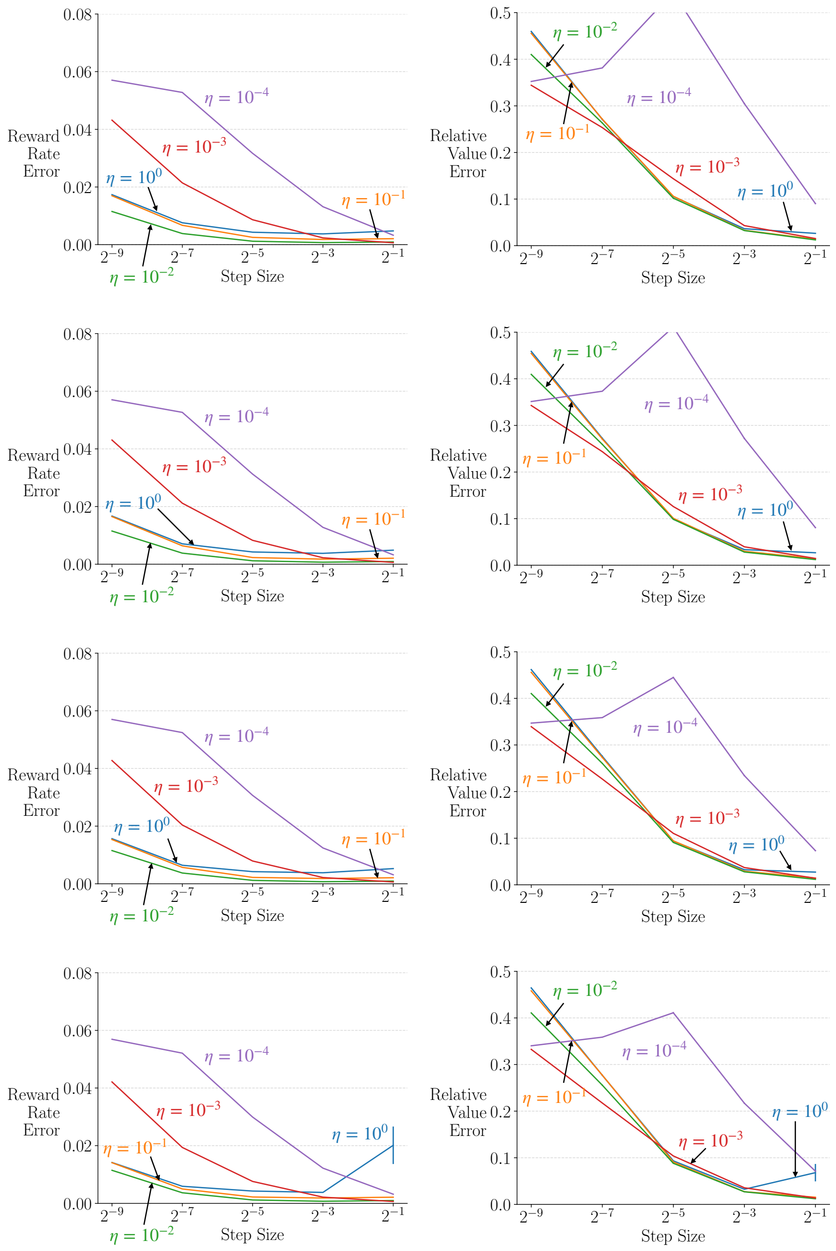


Figure B.1: Plots showing learning curves and parameter studies for inter-option Differential Q-evaluation-learning in the continuing Four-Room domain when the goal was to go to `G1`. $\mathcal{O} = \mathcal{A} + \mathcal{H}$. $\beta = 2^{-3}$ (first row), $\beta = 2^{-5}$ (second row), $\beta = 2^{-7}$ (third row), $\beta = 2^{-9}$ (fourth row). The first and second columns show the reward rate error, and the relative value error achieved by the algorithm, respectively. The experiment setting and the plot axes are the same as those in Figure 4.3.

Figure B.8 shows learning curves for $\epsilon = 0$ and $\epsilon = 0.4$ respectively. Figure B.4 – Figure B.7 show sensitivity curves for the four tested algorithms when $\epsilon = 0$.

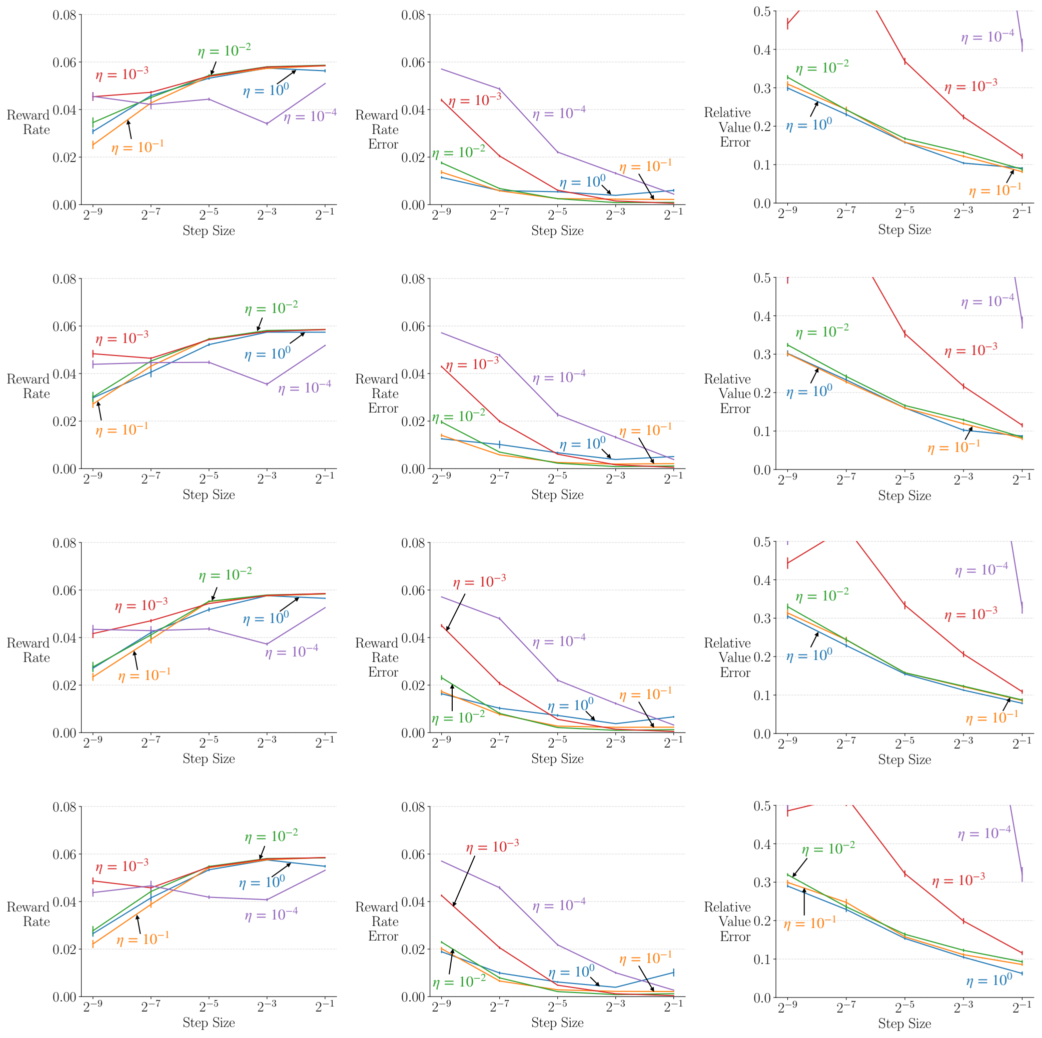


Figure B.2: Parameter studies of inter-option Differential Q-learning. $\mathcal{O} = \mathcal{A} + \mathcal{H}$ and $\beta = 2^{-3}$ (first row), $\beta = 2^{-5}$ (second row), $\beta = 2^{-7}$ (third row), $\beta = 2^{-9}$ (fourth row). The first, second, and third columns show the reward rate, the reward rate error, and the relative value error achieved by the algorithm, respectively. The experiment setting and the plot axes are the same as mentioned in Figure 4.3's caption.

Figure B.9 – Figure B.12 shows sensitivity curves for the four tested algorithms when $\epsilon = 0.4$. The observations I made in Section 5.5 for $\epsilon = 0.2$ also apply here for $\epsilon = 0$ and $\epsilon = 0.4$.

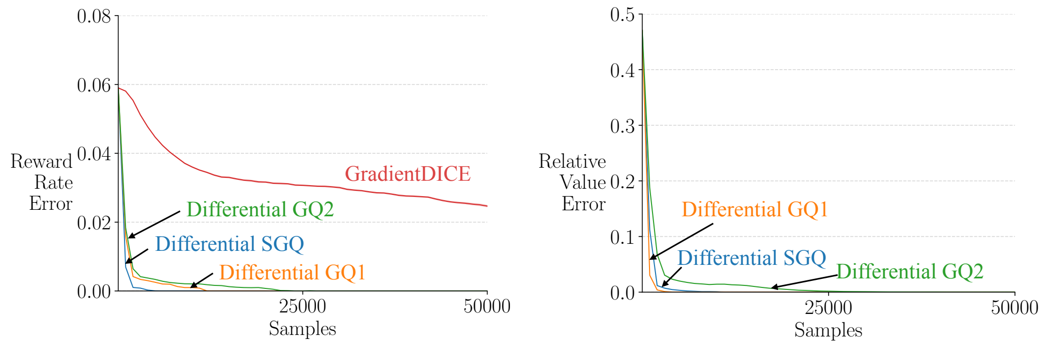


Figure B.3: Learning curves of the four tested algorithms when $\epsilon = 0$. The parameter setting was chosen to minimize the error over the last 5000 steps. The axes have the same meaning as in Figure 3.2a and in Figure 3.2b.

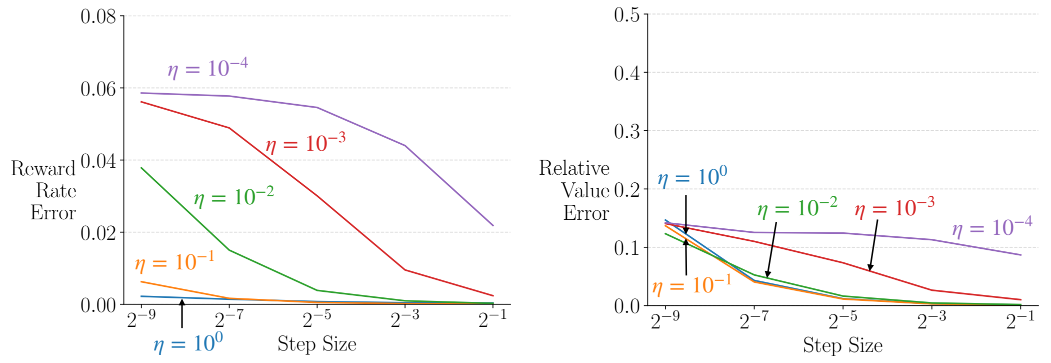


Figure B.4: Sensitivity curves of Differential SGQ when $\epsilon = 0$. The parameter setting was chosen to minimize the error over the entire 50,000 steps. The axes have the same meaning as in Figure 3.2a and in Figure 3.2b.

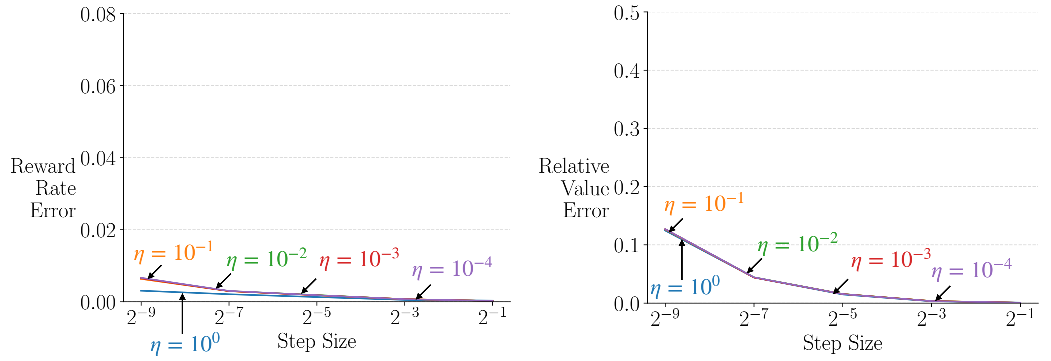


Figure B.5: Sensitivity curves of Differential GQ1 when $\epsilon = 0$. The axes are the same as in Figure 5.3.

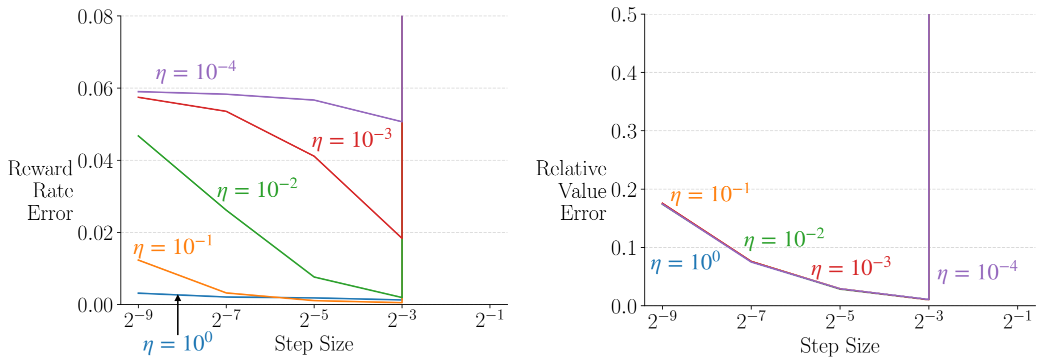


Figure B.6: Sensitivity curves of Differential GQ2 when $\epsilon = 0$. The axes are the same as in Figure 5.3.

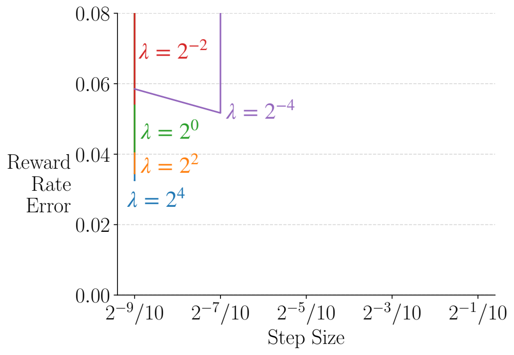


Figure B.7: Sensitivity curves of GradientDICE when $\epsilon = 0$. The axes are the same as in the left subfigure of Figure 5.3.

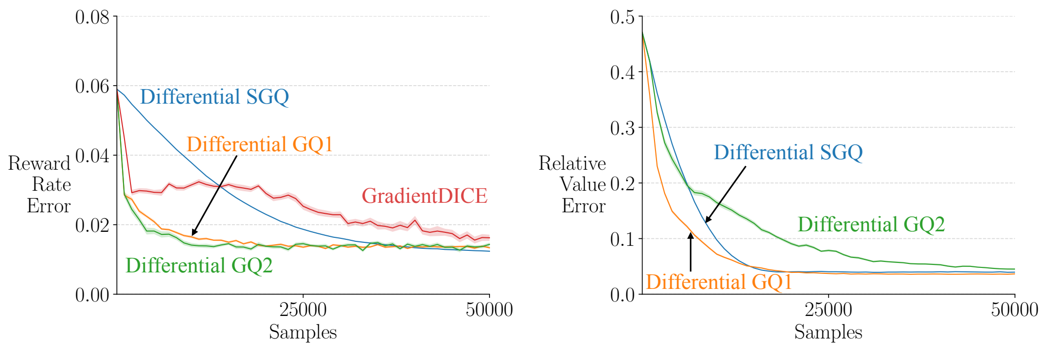


Figure B.8: Learning curves of the four tested algorithms when $\epsilon = 0.4$. The axes are the same as in Figure 5.2.

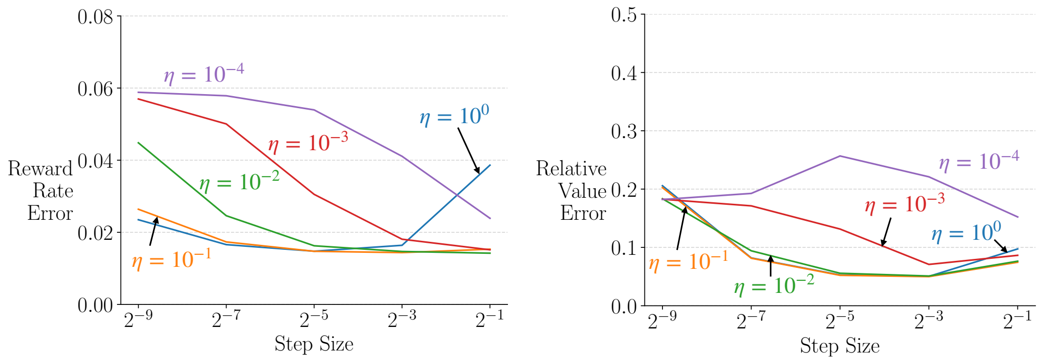


Figure B.9: Sensitivity curves of Differential SGQ when $\epsilon = 0.4$. The axes are the same as in Figure 5.3.

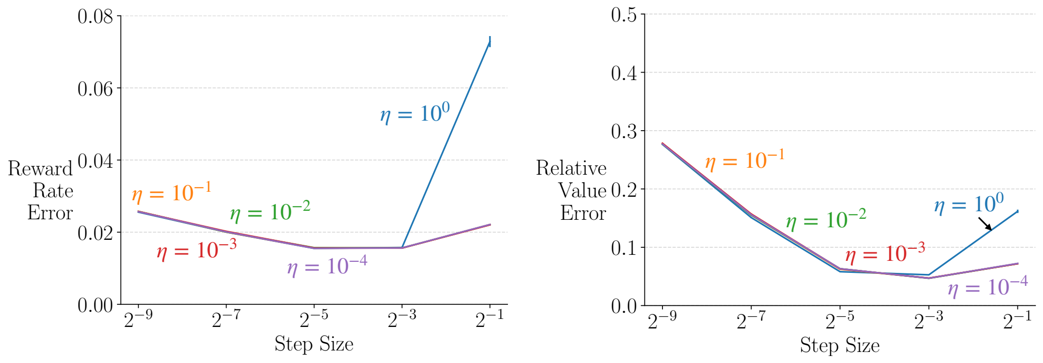


Figure B.10: Sensitivity curves of Differential GQ1 when $\epsilon = 0.4$. The axes are the same as in Figure 5.3.

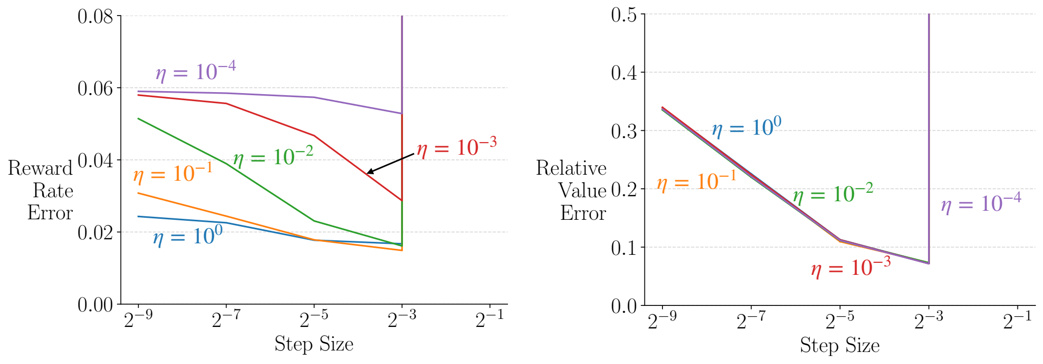


Figure B.11: Sensitivity curves of Differential GQ2 when $\epsilon = 0.4$. The axes are the same as in Figure 5.3.

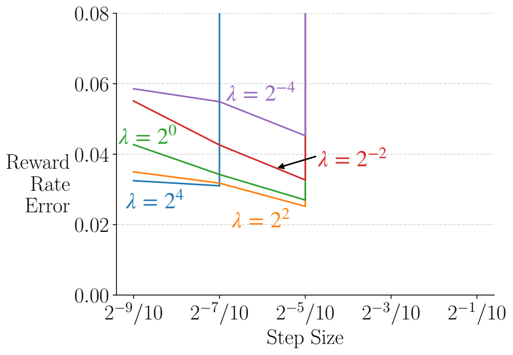


Figure B.12: Sensitivity curves of GradientDICE when $\epsilon = 0.4$. The axes are the same as in the left subfigure of Figure 5.3.