MODEL-BASED REINFORCEMENT LEARNING WITH SMALL SAMPLE SIZE

by

LIANGPENG ZHANG

A thesis submitted to
The University of Birmingham
for the degree of
DOCTOR OF PHILOSOPHY

School of Computer Science
College of Engineering and Physical Sciences
The University of Birmingham
September 2020

Abstract

State-of-the-art reinforcement learning (RL) algorithms generally require a large sample of interaction data to learn sufficiently well, which makes it difficult to apply them to the problems where data is expensive. This thesis studies exploration, transformation bias, and policy selection of model-based RL in finite MDPs, all of which has strong impact on sample efficiency.

Exploration has previously been studied under the setting where learning-process cumulative reward needs to be maximised. When learning-process cumulative reward is irrelevant and sample efficiency is of primary concern, existing strategies become inefficient and analyses become unsuitable. This thesis formulates the planning for exploration problem, and shows that the efficiency of exploration strategies can be better analysed by comparing their behaviours and exploration costs with the optimal exploration scheme of the planning problem. The weaknesses of existing strategies and the advantages of conducting explicit planning for exploration are presented through an exploration behaviour analysis in tower MDPs.

Transformation bias of value estimates in model-based RL has previously been considered insignificant and has not gained much attention. This thesis presents a systematic empirical study to show that when the sample size is small, the transformation bias is not only significant, it can even lead to disastrous effect on the accuracy of value estimates and overall learning performance in some cases. The novel Bootstrap-based Transformation Bias Correction method is proposed to reduce the transformation bias without requiring any additional data. It can work well even when sample size per state-action is very small, which is not possible with the existing method.

Policy selection is rarely studied and has been conducted naively by directly comparing two estimated values in most model-based algorithms, which increases the risk of selecting inferior policies due to the asymmetry of the value estimate distributions. To better study the effectiveness of policy selection, two novel family-wise metrics are proposed and analysed in this thesis. The novel Bootstrap-based Policy Voting method is proposed for policy selection, which can significantly reduce the risk of policy selection failures. Then, two novel tournament-based policy refinement methods are proposed, which can improve general RL performance without needing more data.

ACKNOWLEDGEMENTS

I would like to express my deepest appreciation to my supervisor Prof. Xin Yao. The completion of this thesis would not have been possible without his guidance and support. His encouragement has been invaluable and kept me motivated through this long journey. His questions were like a marine sextant which helped me locate myself in this ocean of knowledge and uncertainty.

I would like to express my gratitude to my thesis group members, Prof. Ata Kaban, Dr. Leandro L. Minku, and former member Dr. Jeremy Wyatt, for their valuable advice and expertise. Their comments and suggestions gave me a wider vision on both my research topics and my PhD study.

I would like to thank Prof. Ke Tang from SUSTech, for his constructive advice on my research and future career. Many thanks to my external examiner, Prof. Richard Everson from University of Exeter, for his helpful comments and suggestions. I would also like to thank Dr. Rami Bahsoon, Dr. Peter Hancox, Dr. Shan He, Dr. Manfred Kerber, Dr. Miqing Li, Prof. Uday Reddy, Prof. Aaron Sloman, Prof. Peter Tino, Dr. Shuo Wang, Sarah Brookes, and all my PhD colleagues for their kind helps with my research and my student life in the School of Computer Science.

Many thanks to my friends Wenjing Hong, Guiying Li, Ning Shi, Haiyang Chen, Malcolm Dick, Gan Zhang, Xiaofen Lu, Liyan Song, Xiaofeng Jiang, Ling Tang, Wanqing Tai, Weifeng Li, Shengcai Liu, Chao Qian, Yaxin Xu, Kyle, Ben, Jeremy, James, John, Rona, Gelc, and Tom, who have always been a great source of support.

Thanks also to the City of Birmingham Symphony Orchestra and the Birmingham Philharmonic Orchestra for their splendid concerts. Music is my source of courage.

Finally, I would like to sincerely thank my parents for their love and encouragement. They themselves are outstanding researchers in their own fields, and their experiences have greatly inspired me throughout my PhD study.

CONTENTS

LIST OF FIGURES

LIST OF ALGORITHMS

LIST OF SYMBOLS

M	Markov decision process
$\mathcal{S}$, $\mathcal{A}$	sets of states and actions
S_t, A_t	state and action at time t
s, a	some state and action
s_i, a_i	the i-th state and action, subscripts indicate the indices rather than the time
P	transition function
$\mathbb{P}$	probability
R	reward function
$R_{\min}, R_{\max}$	minimum and maximum rewards of an MDP
γ	discount factor
ψ	trajectory (i.e. the interaction history and the dataset of RL)
π	policy
Π	set of all policies
G	cumulative reward (aka return)
V^π, Q^π	state and action value functions (aka utility functions) of policy π
π^*, V^*, Q^*	optimal policy and optimal state/action value functions
$\hat{M}$	model MDP
$\hat{X}, \tilde{X}$	estimates of X, e.g. $\hat{P}, \hat{R}, \hat{V}, \hat{Q}$ are estimates of P, R, V, Q
$N_{s,a}, N_{s,a,s'}$	sample size at (s, a) and transition count (i.e. number of observations) of (s, a, s')

ε, δ	when used together, ε and δ indicates the degree of near-optimality and the probability of being non-near-optimal, respectively
$\mathbb{E}[X]$	mathematical expectation of X
D	data demand
$H(D; s, a)$	demand reduction function
D_{init}	initial demand
$\mathcal{D}$	demand space
ϕ	exploration scheme
$M^{\Delta}, \mathcal{S}^{\Delta}, P^{\Delta}, R^{\Delta}$	augmented MDP, state space, transition function, reward function of planning for exploration problem
U^{ϕ}, C^{ϕ}	state/action exploration costs of exploration scheme ϕ
Φ	set of all possible exploration scheme
ϕ^{*}, U^{*}, C^{*}	optimal exploration scheme and optimal state/action exploration costs
$\text{Bias}(\hat{X})$	bias of estimator $\hat{X}$
η	relative bias
$\mathbf{N}, \mathbf{N}_{*,*}$	collections of all $N_{s,a,s'}$ and $N_{s,a}$ respectively
$\mathbf{N}_i^{+}$	the i-th bootstrap data
$\text{avg}[X]$	sample mean of X
b	bootstrap size parameter
d	bootstrap depth parameter
W	policy selector
W_{Ran}, W_{MB}, W^{*}	random guesser, naive model-based selector, oracle selector
W_{BC}, W_{PV}	bias-corrected selector (based on BTBC) and policy voting selector (based on BPV)
ξ	policy selection problem

$\mathrm{Risk}(W;\xi)$	instance-wise policy selection risk of selector W in policy selection problem ξ
$\{M_{(x)}\}$	a family of MDPs with parameter x
$\mathrm{Risk}_{\{M_{(x)}\}}(W)$	family-wise policy selection risk of selector W in MDP family $\{M_{(x)}\}$
$\mathrm{Unfairness}_{\{M_{(x)}\}}(W)$	family-wise unfairness of selector W in MDP family $\{M_{(x)}\}$

CHAPTER 1

INTRODUCTION

1.1 Background

Reinforcement learning (RL) is a branch of machine learning which focuses on learning desirable behaviours by interacting with the environment [Sutton and Barto, 2018, Szepesvári, 2010]. The learning agent observes the environment, takes some actions, see what happens, and tries to improve the utility of its behaviour based on these observations. Such a trial-and-error style learning process occurs frequently in our daily life. For example, when we take showers, we often play with the control valve, turning the temperature of the water up and down several times, to find out the most comfortable setting. This is different from the approach of supervised learning, by which we find out the best temperature setting of the shower by, say, referring to the instruction manual of the heater.

There have been many successes in applying RL to real-world problems in recent years. RL has widely been used in robotics [Kober et al., 2013, Riedmiller et al., 2009, Gu et al., 2017, Zhu et al., 2017, Tan et al., 2018, Hwangbo et al., 2019] to deal with various problems such as motor control, locomotion, and navigation. RL agents can play board, card, and video games at a near-human level, and it is not even too surprising in recent days to hear that in yet another game the deep-RL-based agents have beaten the best human players in the world [Mnih et al., 2015, Silver et al., 2016, Silver et al., 2018, Vinyals et al., 2019, OpenAI et al., 2019]. RL has also been applied to many other domains such as dialogue generation and chatbot [Li et al., 2016, Serban et al., 2017], analysing biological data [Mahmud et al., 2018], autonomous driving [Shalev-Shwartz et al., 2016, Sallab et al., 2017], industrial process control [Nian et al., 2020], and communications and networking [Luong et al., 2019], just to name a few.

A common characteristic of these successes is that they utilise a very large sample of interaction data. For example, in training AlphaGo [Silver et al., 2016] which plays the board game Go, not only a set of 30 million human expert data was used in pre-training, but also another set of 30 million self-play matches was used in the RL phase. Assume that a human player can have 30 matches per day, that 30 million matches need 1 million days, or 2738 years, to finish. Another example is OpenAI Five [OpenAI et al., 2019], a game playing agent designed for the video game Dota 2. The agent was trained for 10 months, each day playing 180 years worth of game, resulting in a gigantic sample size of 54,000 years worth of game in total.

Apparently, it is difficult (if not impossible) to replicate the successes of AlphaGo and the like in domains where obtaining data is more expensive. For example, in autonomous driving, obtaining non-simulated data requires a real car in good condition, some specially designed devices that allow computers to manipulate the car, and letting the car run in real streets. Furthermore, for safety concerns, a human backup driver is usually needed, so that when the autonomous driving system fails, the human driver can take over control. Even this is not sufficient to prevent fatal accidents such as the one caused by an Uber autonomous car in 2018 [BBC, 2018], and thus even more security measures must be taken, further increasing the cost of obtaining real data. Although there are some driving simulators that can provide cheaper data, most of them are inaccurate and sometimes very different from the real driving environment (e.g. in [Sallab et al., 2017], a simulator with a top-down perspective rather than an in-car one was used for experiments). All these factors make it tremendously difficult in autonomous driving to obtain as much data as in the cases of AlphaGo and OpenAI Five. In such cases, having a learning algorithm that works reasonably well with only a small sample of data becomes highly important.

The quality of being able to learn well with small sample size is called *sample efficient*. There are many factors that decide the sample efficiency of an reinforcement learning algorithm. A common factor nowadays is the utilisation of deep neural networks [LeCun et al., 2015] as the function approximation module of the RL algorithm. It is well-known that deep neural networks by themselves are already highly data-hungry, and combining RL with it worsens the problem. Thus improving the neural network optimisation method has been one main focus of the RL research community in recent years, many have been quite successful and widely applied (e.g. TRPO [Schulman et al., 2015], DDPG [Lillicrap et al., 2016], and PPO [Schulman et al., 2017]).

This thesis, on the other hand, investigates several factors that lie in the more foundational part of RL and are independent of function approximation. The first factor is *exploration*. In reinforcement learning, exploration encourages the agent to actively visit states and try actions that have higher uncertainty. If exploration is not carried out properly, the learning agent may either prematurely converge to an inferior policy due to lack of key information, or waste a lot of time collecting useless data.

There has been rich literature regarding exploration, many claiming to be significantly improving the sample efficiency, some even theoretically near-optimal. However, benchmarking has shown such claims to be questionable, where complex systematic exploration strategies were outperformed by simple random strategies [Kuleshov and Precup, 2014, Tijsma et al., 2016, Taiga et al., 2020]. We consider this mismatch between theory and practice as a twofold problem. The first is that the cumulative-reward-based metrics of exploration efficiency used in existing analysis do not reflect practical performance in terms of sample size. The second is that, due to being designed under the cumulative-reward-based metrics, existing strategies do not actually care to reduce useless data collection as long as such behaviour yield sufficient rewards. These make a theoretically sound exploration strategy practically terrible.

The second factor been studied is the *transformation bias* of policy evaluation in model-based reinforcement learning. The agent estimates the utility (i.e. the state and action values) of different policies and identifies the best one according to the utility, and thus the accuracy of the utility estimates has direct impact on the performance of learning algorithms. In model-based RL, the utility is obtained by solving the Bellman equations. The solution of the equations is a nonlinear mapping from the estimated transition probabilities of the environment to the state/action values. Such nonlinearity introduces bias to the value estimates, which we call transformation bias.

Transformation bias has historically been considered insignificant and unimportant compared to the variance of value estimates [Mannor et al., 2004]. However, our empirical results show that the transformation bias can be significant and sometimes even as large as the true values to be estimated, especially when the sample size is small. Without proper correction, the only way to reduce the bias is to collect a large amount of data, making the algorithm less sample efficient.

The third factor is the *value comparison and policy selection*. The execution of the learning process can be seen as an infinitely repeatable experiment in probability theory sense, and thus the estimated utilities are random variables following some distributions decided by the environment and the algorithm. The distributions of the estimated utilities are rarely symmetric, which makes it possible in a single learning task that some policies are more likely to be overestimated while the others are more likely to be underestimated. Consequently, naive policy selection that simply choose the policy with higher estimated utility can be biased towards inferior policies that are more likely to be overestimated. This problem is called unfairness by [Doroudi et al., 2017].

When the sample size is small, the distribution of estimated utilities tend to be more skewed, which may increase the chance of selecting the wrong policies. Existing algorithms without proper policy selection method have to rely on large sample size to negate such effect, resulting in low sample efficiency.

This thesis aims to improve sample efficiency of model-based RL by investigating these three factors. The remainder of this chapter is organised as follows. Section 1.2 specifies the scope of this thesis. Section 1.3 poses the research questions to be answered in the thesis. Section 1.4 outlines the contributions of the thesis. Section 1.5 describes the organisation of the following chapters of this thesis.

1.2 Scope of the thesis

Although since about 2014 the majority of RL research community has shifted the focus to model-free deep reinforcement learning in continuous Markov decision processes (MDPs), this thesis limits its own scope to the sample efficiency of model-based RL in finite (discrete) MDPs. The value of the researches presented in this thesis has frequently been questioned by the reviewers of our paper submissions on the ground that the results are not instantly applicable to the former setting. Thus, the remainder of this section is dedicated to explain the reasons behind our decision.

First, deep RL is not the best solution to every RL problem (an obvious fact neglected by many). Deep neural networks obtain their strong representation power and model flexibility at the cost of being data-hungry, power consuming, and even sensitive to implementation [Engstrom et al., 2019]. When resources are not sufficient to find a suitable architecture, parameter setting and implementation, or when strong representation power is not necessary at all, using deep RL is not a wise choice. Sometimes even simple random search algorithms can provide competitive results at a much less cost [Mania et al., 2018]. The same argument holds for model-free vs model-based RL (the former has been more popular but has worse sample efficiency than the latter in general [Luo et al., 2019, Łukasz Kaiser et al., 2020]).

Second, the common claim that discrete problems in RL are less challenging than continuous ones and thus are less worthy to be studied is a fallacy. Consider two tasks, where the first one is to draw random curves on a screen, while the second one is to play contract bridge. The former is a continuous problem that can be dealt with without effort (or even without learning at all), while the latter is a discrete problem that requires extensive experience. Thus, the difficulty of a learning task should not be judged solely on the ground whether it is continuous or not. The same argument holds for the scale of MDPs. If the difficulty of learning is positively correlated to the size of state space, then multi-armed bandits (one-state MDPs) should be trivial, which is simply untrue.

Last but not least, not being instantly applicable does not mean irrelevant. On the contrary, the insights gained from studying the problems at more fundamental level can often help improve more sophisticated methods. All three factors discussed in this thesis (exploration, transformation bias, and policy selection) can be potential factors to the low sample efficiency of state-of-the-art RL algorithms as well.

1.3 Research questions

The three factors mentioned in Section 1.1 lead to three works, each presented in a separate chapter of this thesis. The research questions that will be discussed in these chapters are given in the following subsections respectively.

1.3.1 Explicit planning for efficient exploration

Exploration has been studied for more than 30 years, yet state-of-the-art RL algorithms still struggle in exploration. The first work of this thesis points out that the low efficiency is partly due to existing exploration strategies being designed under learning-process cumulative reward based performance metrics rather than sample size based ones. When learning efficiency in terms of sample size is of primary concern, existing strategies reduce to hand-designed heuristics that lack global planning. As a result, existing strategies can waste a lot of time visiting already well-known states and actions, reducing their overall sample efficiency.

To resolve this problem, the following research questions are going to be answered in the first work of this thesis.

Q1.1 How to better analyse the sample efficiency of exploration strategies with a sample-size-based performance metric from a planning-based point of view?

Q1.2 According to this new analysis, what are the weaknesses of existing exploration strategies?

Q1.3 Can explicit planning for exploration significantly improve the sample efficiency?

1.3.2 Transformation bias of model-based RL and its correction

In model-based RL, the state/action values are estimated by solving the Bellman equations with estimated transition probabilities. The solution of the Bellman equations is a nonlinear mapping from the transition probabilities to the state/action values. Even if transition probability estimates are unbiased, such nonlinearity may make value estimates biased. To distinguish it from other types of biases in RL, we call it transformation bias in this thesis. Mannor et al. [2007] suggested that correcting the transformation bias is unnecessary because it reduces quickly as sample size grows, and also is smaller than the variance in general. However, our empirical study presented in this thesis shows that when sample size is small, the transformation bias can be destructively large and thus has profound impact to the sample efficiency of model-based RL if no correction is conducted.

To deal with the transformation bias of model-based RL, the second work presented in this thesis will answer the following research questions.

Q2.1 How does the transformation bias relate to sample efficiency?

Q2.2 In what cases is the transformation bias remarkable and needs correction?

Q2.3 How to correct the transformation bias when sample size is small?

1.3.3 Effectiveness of policy selection in model-based RL and its improvement

Estimated state/action values are essentially statistics computed from interaction experience. Due to the randomness of transitions, the value estimates are random variables that usually follow some asymmetric distributions. The asymmetry may make the agent favour inferior policies that are more likely to be overestimated. There is little research on policy selection of model-based RL, and existing methods have to rely on using larger samples to reduce the chance of wrong comparison, resulting in low sample efficiency.

The third work presented in this thesis will answer the following research questions regarding the effectiveness of policy selection.

Q3.1 How to measure the effectiveness of policy selection methods? Is the naive model-based selector good under this measurement?

Q3.2 How to improve the effectiveness of policy selection?

Q3.3 How to use the improved policy selection methods to further improve overall model-based RL performance?

1.4 Contributions of the thesis

The contributions of this thesis are as follows.

1. We propose the planning for exploration (PFE) problem and formulate it as an augmented MDP. We show that given the exploration demand and true transitions, the optimal exploration scheme can be found by solving the augmented MDP using a modified version of value iteration algorithm. We show that sample efficiency of exploration strategies can be analysed by comparing their behaviours with the optimal exploration scheme.

2. Exploration behaviours in a class of tower MDPs are analysed to illustrate the benefit of explicit planning for exploration, and to expose two weaknesses of existing methods, namely distance traps and reward traps.

3. We present an extensive study on the transformation bias of state/action value estimates, revealing the relationship between the scale of the bias and relevant factors including sample size, scale of MDP, and transition dynamics.

4. We propose the Bootstrap-based Transformation Bias Correction (BTBC) method which can significantly reduce transformation bias of state/action value estimates even if sample size is small.

5. We propose two metrics for measuring the effectiveness of policy selection methods, namely family-wise policy selection risk and family-wise unfairness, and show that they are more suitable for measuring the overall effectiveness than the original strict fairness proposed by [Doroudi et al., 2017] and the instance-wise policy selection risk. We then analyse the naive model-based selector to show that it is not satisfactory in terms of these metrics.

6. We propose the Bootstrap-based Policy Voting (BPV) method for policy selection which has significantly better family-wise effectiveness than the naive model-based selector, and also has less family-wise unfairness than the bias-corrected selector based on our BTBC. We then propose two policy refinement methods, Bias-corrected Tournament (BCT) and Policy Voting Tournament (PVT), to improve overall model-based RL performance.

1.5 Organisation of the thesis

The remainder of this thesis is organised as follows.

Chapter 2 provides more details on the background of this thesis, including a short introduction to model-based reinforcement learning and a review of existing researches on exploration, bias correction, and policy selection. Importantly, it discusses the crucial differences between the learning-while-serving workflow commonly assumed in literature and the learning-*then*-serving workflow commonly used in industry, as well as their relation to this thesis.

Chapter 3 introduces our work on explicit planning for exploration, which answers research questions Q1.1-1.3 and leads to contributions 1 and 2.

Chapter 4 presents our study on the transformation bias of value estimates in model-based RL, which answers research questions Q2.1-2.3 and leads to contributions 3 and 4.

Chapter 5 presents our research on the effectiveness of policy selection, answering research questions Q3.1-3.3 and leading to contributions 5 and 6.

Finally, Chapter 6 summarises the thesis and suggests some potential future work.

CHAPTER 2

BACKGROUND AND LITERATURE REVIEW

This chapter provides detailed background for the later chapters. Section 2.1 gives a short introduction to model-based reinforcement learning. Section 2.2 discusses an important deviation of researches from practice in RL, that is, the learning-while-serving workflow followed by most RL researches vs the learning-then-serving workflow more often seen in industry. Sections 2.3, 2.4, 2.5 respectively review the literature on exploration, bias correction and policy selection, and point out their shortcomings. Section 2.6 shortly introduces the statistical technique frequently used in this thesis called bootstrap. Section 2.7 summaries this chapter.

2.1 A brief introduction to model-based reinforcement learning

This section shortly introduces the essential concepts of model-based reinforcement learning that are closely related to this thesis. A more comprehensive introduction can be found in the textbooks [Sutton and Barto, 2018, Szepesvári, 2010, Wiering and Van Otterlo, 2012].

2.1.1 Markov decision processes

In reinforcement learning, the learning agent interacts with the environment by taking actions and observing the changes of the state of the environment. The interaction process is often formulated by a (discounted, infinite horizon) Markov decision process (MDP) M, which is a tuple $(\mathcal{S}, \mathcal{A}, P, R, \gamma)$, where $\mathcal{S}$ is the set of states, $\mathcal{A}$ is the set of actions, $P : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \mapsto [0, 1]$ is the transition probability function, $R : \mathcal{S} \times \mathcal{A} \mapsto [R_{\min}, R_{\max}]$ or $R : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \mapsto [R_{\min}, R_{\max}]$ is the reward function, and $\gamma \in (0, 1)$ is a constant called discount factor. There also exist several less common formulations such as the undiscounted and finite-horizon MDPs, but these will not be the main focus of this thesis.

The current state represents the circumstance of the environment the agent is in. At each time step $t = 1, 2, 3, \ldots$, the agent observes the current state $S_t \in \mathcal{S}$, and picks an action $A_t \in \mathcal{A}$ to execute. The state of the environment then changes to S_{t+1} following the probability distribution given by the transition function P, i.e. $\mathbb{P}(S_{t+1} = s'|S_t = s, A_t = a) = P(s'|s, a)$. By definition, $\sum_{s' \in \mathcal{S}} P(s'|s, a) = 1$ for all $s \in \mathcal{S}$ and $a \in \mathcal{A}$. It is called *Markov* decision process because the next state S_{t+1} is only dependent on the current state S_t and action A_t, but not the ones in the past (S_{t-1}, A_{t-1}, S_{t-2}, A_{t-2},...). For convenience, selecting action a at state s is denoted (s, a) while the transition from state s to s' under action a is denoted (s, a, s').

The reward function R indirectly provides information about the target behaviour to be learnt by the agent. The basic idea of reinforcement is to reward the agent when it does something right and to punish it when it does something wrong. This is implemented as providing reward $R(s, a)$ or $R(s, a, s')$ to the agent when transition (s, a, s') takes place. With a properly set reward function and learning algorithm, the agent seeking to maximise total reward eventually learns the behaviour wanted by the RL user.

The interaction history with n transitions is represented as a trajectory $\psi_n = (S_1, A_1, R_1, S_2, A_2, R_2, ..., S_n, A_n, R_n, S_{n+1})$. A data point (S_i, A_i, R_i, S_{i+1}) indicates that at time i the transition (S_i, A_i, S_{i+1}) happened and yielded reward R_i. A trajectory of length n has n such data points, ending at S_{n+1} without action A_{n+1}.

A (deterministic) policy $\pi : \mathcal{S} \mapsto \mathcal{A}$ specifies which action is going to be taken by the agent at each state. The set of all possible policies for a given MDP is denoted Π. Policy represents the learnt behaviour of the agent and is the output of RL.

The utility of a policy is indicated by its value functions. The discounted cumulative reward (sometimes referred to as the return) of a trajectory $\psi_n = S_1, A_1, R_1, S_2, A_2, R_2, ..., S_{n+1}$ is defined as $G_n = R_1 + \gamma R_2 + \gamma^2 R_3 + ... + \gamma^{n-1} R_n$. In infinite horizon setting, the state value $V^\pi(s)$ of policy π is defined as the limit of the expected return $\lim_{n\to\infty} \mathbb{E}[G_n]$ the agent obtains when starting from state s and following policy π forever, i.e. $V^\pi(s) = \lim_{n\to\infty} \mathbb{E}[G_n | S_1 = s, A_t = \pi(S_t)\ (t \le n)]$. The action value $Q^\pi(s,a)$ is the limit of the expected return when starting from taking action a at state s, then following π thereafter, i.e. $Q^\pi(s,a) = \lim_{n\to\infty} \mathbb{E}[G_n | S_1 = s, A_1 = a, A_t = \pi(S_t)\ (2 \le t \le n)]$. With $0 < \gamma < 1$ and R_t bounded by some $[R_{\min}, R_{\max}]$ for all t, we have the return $G \in [\frac{R_{\min}}{1-\gamma}, \frac{R_{\max}}{1-\gamma}]$, and thus the state/action values are always bounded despite the infinite summation. Note that using an infinite horizon in the utility definition does not mean that the agent has to actually interact infinitely. Rather, it is more of a choice for theoretical concern and mathematical convenience. By definition, it holds that $V^\pi(s) = Q^\pi(s, \pi(s))$.

The relationship between the utility of different states and actions are given by the Bellman equations. For any policy π and state s, the Bellman equation for state value function V^π is:

$$V^\pi(s) = \sum_{s' \in \mathcal{S}} P(s'|s,\pi(s))\big(R(s,\pi(s),s') + \gamma V^\pi(s')\big), \tag{2.1}$$

and for action value function Q^π is:

$$Q^\pi(s,a) = \sum_{s' \in \mathcal{S}} P(s'|s,a)\big(R(s,a,s') + \gamma Q^\pi(s',\pi(s'))\big). \tag{2.2}$$

These equations can be proved by their definition.

A good policy should lead to high expected return. The optimal policy, denoted π^*, is the one that can lead to maximum expected return, i.e. $V^{\pi^*}(s) \ge V^\pi(s)$ and

$Q^{\pi^*}(s,a) \geq Q^{\pi}(s,a)$ for all policies $\pi \in \Pi$, states $s \in \mathcal{S}$, and actions $a \in \mathcal{A}$. For convenience, V^{π^*} and Q^{π^*} are denoted V^* and Q^* respectively. It can be proved that the following the Bellman optimality equations hold:

$$V^*(s) = \max_{a \in \mathcal{A}} \sum_{s' \in \mathcal{S}} P(s'|s,a)\big(R(s,a,s') + \gamma V^*(s')\big), \tag{2.3}$$

$$Q^*(s,a) = \sum_{s' \in \mathcal{S}} P(s'|s,a)\big(R(s,a,s') + \gamma \max_{a' \in \mathcal{A}} Q^*(s',a')\big). \tag{2.4}$$

In addition, it holds that $V^*(s) = \max_{a \in \mathcal{A}} Q^*(s,a)$. It is possible that more than one policy has the same state/action values at all states, thus the optimal policy might not be unique, but the optimal state/action value function is always unique (see Section 3.6 of [Sutton and Barto, 2018]).

2.1.2 Solving true MDPs via policy planning

If the true P and R are known, then the problem of computing utility for a given policy reduces to solving a system of linear equations given by Equations 2.1 or 2.1, which can be done by straightforward methods such as matrix inversion and Gaussian elimination. Equations 2.3 and 2.4 involve maximisation operations and thus are slightly more complicated. Policy planning algorithms such as Policy Iteration and Value Iteration [Puterman, 1994] can be used to compute V^* and Q^* by iteratively solving these equations. It has been proved that these algorithms converge to the true optimal values in the limit, or to the near-optimal ones in polynomial time under some assumptions [Littman et al., 1995]. The optimal policy π^* can easily be obtained by $\pi^*(s) = \text{argmax}_a Q^*(s,a)$ or equivalently $\pi^*(s) = \text{argmax}_a \sum_{s' \in \mathcal{S}} P(s'|s,a)\big(R(s,a,s') + \gamma V^*(s')\big)$.

2.1.3 Model-based and model-free RL

Since in most RL settings the transition probability function P and reward function R are unknown to the learning agent, V and Q must be estimated from the observations. Existing methods can be put into two categories, namely model-based RL and model-free RL, based on whether the algorithm maintains a model of the unknown MDP or not. As discussed in Section 1.2, in this thesis we focus on model-based RL, though we will also discuss briefly the transformation bias in model-free RL in Section 4.6.

Model-based RL creates a model MDP $\hat{M} = (\mathcal{S}, \mathcal{A}, \hat{P}, \hat{R}, \gamma)$ of the unknown true MDP $M = (\mathcal{S}, \mathcal{A}, P, R, \gamma)$, and updates it with collected transition and reward information. The most straightforward way to update the model transition function is to use $\hat{P}(s'|s,a) = N_{s,a,s'}/N_{s,a}$, where $N_{s,a}$ is the sample size at (s, a) (that is, the number of times action a was taken at state s), and $N_{s,a,s'}$ is the number of transitions (s, a, s') occurred in that sample. As for the rewards, $\hat{R}(s,a) = \sigma(s,a)/N_{s,a}$ or $\hat{R}(s,a,s') = \sigma(s,a,s')/N_{s,a,s'}$ can be used, where $\sigma(s,a)$ and $\sigma(s,a,s')$ are the sum of the reward gathered along with state-action pair (s,a) and transition (s,a,s'), respectively. In many RL settings the reward function R is deterministic, and the estimation of R becomes trivial (i.e. $\hat{R} = R$).

After constructing a model MDP $\hat{M}$, model-based RL use it in place of the unknown true MDP M as the input of the policy planning algorithms mentioned in the previous section to compute estimated value functions $\hat{V}$, $\hat{Q}$, $\hat{V}^*$, or $\hat{Q}^*$. The optimal policy of $\hat{M}$ then can be obtained in the same way as introduced in Section 2.1.2. With the sample size $N_{s,a} \to \infty$ at every state-action pair (s,a), the model MDP $\hat{M}$ converges in probability towards the true MDP M, and thus the optimal policy of $\hat{M}$ is a consistent estimator of the true unknown optimal policy π^*.

Historically, model-based RL is considered expensive both in time and space complexity, because it needs to store the whole transition function and iterate all entries when computing state/action values. Model-free RL methods were then proposed to avoid these problems by not estimating transition probabilities directly, but using visit frequencies of each transition in a trajectory to approximate the effect of transition probabilities in value computation. A detailed discussion regarding the relationship between model-based and model-free RL can be found in Chapter 8 of [Sutton and Barto, 2018].

Model-free RL has had much more popularity in the last 10 years because it is easier to implement and to combine with function approximation techniques than model-based RL, and has been successfully adapted to deep RL relatively earlier (e.g. DQN [Mnih et al., 2015] and A3C [Mnih et al., 2016]). Model-based RL has regained attention in recent years for being more sample efficient than model-free RL, and there have also been much efforts to adapt it to deep RL, such as [Nagabandi et al., 2018, Luo et al., 2019, Łukasz Kaiser et al., 2020].

2.2 Learning-while-serving vs learning-then-serving

This section discusses an important deviation of RL academic research from industrial practice, that is, the learning-while-serving and the learning-*then*-serving workflows. This deviation, though very often completely neglected by the RL research community, is crucial for understanding some gaps between theory and practice regarding sample efficiency.

In most researches, the workflow of RL is automatically assumed to be learning-while-serving, where the agent is trained and deployed to the application at the same time, not only synchronously, but also in the same MDP interaction process. That is, the agent is assumed to learn *by* fulfilling its real-world duties. Such a setting closely resembles our own experiences, in which we learn and improve ourselves based on the feedbacks we get in our daily life.

However, the industry more often uses a learning-*then*-serving workflow when applying RL, where the agent is first trained during its production process, then deployed to its actual serving process with the policy produced in the previous process. Learning does not happen in the serving process, and thus the learnt policy does not change after the learning process is finished. Examples of explicit preference to this workflow in industry can be found in [Nian et al., 2020].

The separation of learning and serving processes has three important practical benefits. First, it reduces the risk of undesirable incidents happening during service due to unexpected changes of behaviour. No customer would like to see their autonomous cars running into buildings, service robots damaging furnitures at home, bots acting stupidly in video games, just because the learning agent decides to try something new. Such incidents, if necessary for learning purpose, better happen only under controlled circumstances supervised by experts in the production phase.

Second, the separation allows quality control, where the manufacturer can test the trained agent and decide whether it satisfies the specification of the product or not. Conducting quality control is more difficult in the learning-while-serving workflow because learning is not halted and the policy could change (sometimes drastically) after testing, unless the learning process is completely converged (i.e. no new data could possibly change the policy, which is difficult to guarantee in practice).

Third, the separation significantly reduces the cost of providing services, because running learning algorithms requires far more resources than simply executing a fixed policy. The fast execution of RL policy is also a significant advantage of RL against traditional control methods that relies on online searching and optimisation.

The deviation has crucial impact on both objective and evaluation of RL algorithms. In learning-while-serving setting, the cumulative rewards of the learning process and serving process is exactly the same thing. In learning-then-serving setting, on the other hand, the cumulative rewards of the learning process and serving process are completely separated. Since the cumulative reward in serving process decides the real-world value of the agent as a product/service, this means that in learning-while-serving setting the learning-process cumulative reward is exactly the objective to be maximised during learning. In contrast, in learning-then-serving setting the objective of learning is to maximise expected serving process cumulative reward, preferably with less steps of learning because longer learning process means higher production cost.

Maximising learning-process cumulative reward leads to counter-intuitive scenarios where converging *slower* to the optimal policy could be a better choice. We use the following example to show this point.

Example 2.1. Consider a task in which agents can press two buttons No and Yes, where pressing No yields 0 reward while pressing Yes yields 1 reward. Consider two learning agents Alice and Bob, where Alice presses No 100 times, then decides that Yes is better and presses only Yes thereafter, while Bob presses No and Yes alternately for 150 steps, then presses only Yes thereafter. Since the optimal policy is pressing only Yes, Alice converges to the optimal policy at $t = 101$ while Bob converges at $t = 151$.

In this example, the undiscounted cumulative reward for Alice is $G_t^A = 0$ for $t \leq 100$ and $G_t^A = t - 100$ for $t > 100$, while for Bob it is $G_t^B = \lfloor t/2 \rfloor$ for $t \leq 150$ and $G_t^B = \lfloor 150/2 \rfloor + t - 150 = t - 75$ for $t > 150$. For the discounted case the expressions are slightly more complicated, where for Alice it is $G_t^A = 0$ for $t \leq 100$ and $G_t^A = \frac{\gamma^{100}(1-\gamma^{t-100})}{1-\gamma}$ for $t > 100$, while for Bob it is $G_t^B = \frac{\gamma(1-\gamma^{2\lfloor t/2 \rfloor})}{1-\gamma^2}$ for $t \leq 150$ and $G_t^B = \frac{\gamma(1-\gamma^{150})}{1-\gamma^2} + \frac{\gamma^{150}(1-\gamma^{t-150})}{1-\gamma}$ for $t > 150$.

The curves of G_t^A and G_t^B in both undiscounted and discounted (with $\gamma = 0.97$) cases are shown in Figure 2.1. It can be seen from the curves that, despite Bob converges to the optimal policy slower than Alice, his cumulative reward of learning process is higher than Alice at any time step $t > 0$.

In fact, in discounted setting with $\gamma = 0.97$, $G_t^B \geq \gamma + \gamma^3 \approx 1.8827$ for $t \geq 4$, while $G_t^A \leq \frac{\gamma^{100}}{1-\gamma} \approx 1.5851$, and thus even if Bob never presses the button Yes again (i.e. converges to the wrong policy) after $t > 4$, his cumulative reward is still higher than Alice, no matter how many times Alice presses Yes in the long run. It is thus very clear that learning-process cumulative reward as a performance metric strongly favours early exploiters and in some cases does not even care whether the agent learns a good policy or not.

Such criteria is clearly problematic in the learning-then-serving setting where how fast an agent learns a desirable policy is of primary concern. In the example above, both agents eventually learns the same policy, which in an industrial setting means that two

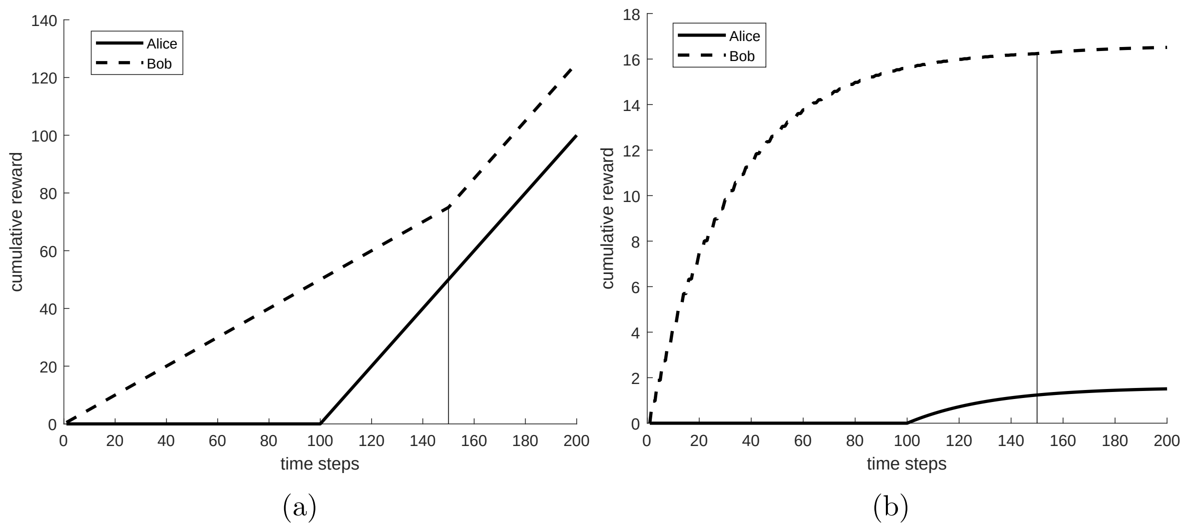

Figure 2.1: Cumulative rewards of two learning processes, where Alice converges to the optimal policy much earlier than Bob, but her learning-process cumulative reward is lower than Bob at every time step. (a) Undiscounted case. (b) Discounted case.

production processes eventually yield the same product. Meanwhile, Alice only uses 2/3 of the steps Bob needs to converge. Since total sample size is equivalent to the number of learning steps, this means the production cost of Alice in terms of data is only 2/3 of Bob. It is clearly inappropriate to still claim that Bob learns more efficiently than Alice.

Having the more practical view of sample efficiency in mind, we follow the learning-then-serving setting throughout this thesis. Consequently, the sample efficiency of an algorithm is evaluated either by the sample size needed to output a desirable policy, or the quality of the output policy under a fixed sample size. It will not consider, say, the cumulative reward collected during learning, or the number of steps taking suboptimal actions during learning, which have been more common choices in RL literature. The difference immediately affects the viewpoint regarding exploration, which is discussed in the next section.

2.3 Exploration

In most RL settings, the agent starts from *tabula rasa* and relies on environment interaction to collect information. In traditional learning-while-serving workflow introduced in Section 2.2, there exists a so-called "exploration vs exploitation" dilemma, where the agent has to decide whether to try actions that are deemed suboptimal by its current knowledge but may bring more information and thus potentially more rewards in the long run (exploration), or to choose actions with the highest estimated utility in order to collect more short-term rewards (exploitation) [Sutton and Barto, 2018].

The dilemma becomes irrelevant in the learning-then-serving workflow followed by this thesis, because the cumulative reward during learning is not the objective to be maximised by the agent. Here the sole purpose of environment interaction during learning is to collect information, and rewards are just part of such information. Thus, no exploitation is needed in learning-then-serving setting. This scenario is called "pure exploration" in literature on multi-armed bandits [Bubeck et al., 2009].

Many exploration strategies proposed for learning-while-serving RL can work adequately in learning-then-serving RL, in the sense that algorithms using them can usually avoid converging to undesirable policies due to lack of key information. However, since they are designed to maximise cumulative reward of learning process rather than sample efficiency directly, there is still a large space for improvement when applied to learning-then-serving RL. The following subsections review common exploration strategies in RL, their theoretical analysis and benchmarking results, then point out their shortcomings.

2.3.1 Exploration strategies

Exploration strategies are part of learning algorithms that manage the exploration vs exploitation balance in learning-while-serving workflow and the scheme for exploration in learning-then-serving workflow. Existing strategies can be broadly put into three categorises, namely random strategies, systematic strategies, and Bayesian approach.

Random exploration strategies

The simplest yet widely applied exploration strategy is ε-greedy. At every learning step, with ε probability the agent selects a random action from uniform distribution, and with $1 - \varepsilon$ probability it selects the action greedily with the highest estimated utility, where $\varepsilon \in (0, 1)$. Parameter ε is often initialised to a small value (e.g. 0.03) and annealed to 0 to let the agent exploit most of the time and eventually converge to the fully greedy behaviour. Due to its simplicity, ε-greedy is widely used in applications, such as Atari 2600 game playing [Bellemare et al., 2013, Mnih et al., 2015] and robotics [Riedmiller et al., 2009]. On the other hand, the exploration efficiency of ε-greedy is terrible due to it relying mostly on luck to discover new information. Whitehead [1991] and Li [2012] have shown that in some chain MDPs with n states, ε-greedy needs $O(2^n)$ steps just to reach the goal state once.

Other random strategies conduct exploration by following some distributions such as Boltzmann and Gaussian distributions, where actions with higher estimated utility are more likely to be selected. Similar to ε-greedy, the degree of randomness is usually set high initially and gradually reduce to 0, or sometimes adjusted adaptively during the learning process [Derhami et al., 2010, Tokic and Palm, 2011]. These strategies are often found in robotics [Whiteson et al., 2007, Kober et al., 2013] because its conservative exploration behaviour (not selecting actions of low estimated utility) may help reduce unwanted incidents that could damage the robots. It has also been adapted for deep RL recently [Haarnoja et al., 2017].

Systematic Exploration

As opposed to the random strategies that rely on luck to discover new information, systematic exploration strategies attempt to reduce uncertainty through directed exploration. The main ideas of systematic exploration, often described as "optimism in the face of uncertainty" principle [Kaelbling et al., 1996], are as follows:

1. State-action pairs with less data have higher uncertainty.

2. The utility of state-action pairs with high uncertainty should be intentionally overestimated to encourage the agent to visit them more often.
3. Collecting data at a state-action pair reduces its uncertainty as well as the corresponding optimistic overestimation.

Early researches implemented the ideas by directly adding a handcrafted exploration bonus $f(s, a)$ to the value estimates $\hat{Q}(s, a)$. Agent greedily maximising $\tilde{Q}(s, a) \overset{\text{def}}{=} \hat{Q}(s, a) + f(s, a)$ thus automatically balances exploration and exploitation by selecting actions with both high estimated values and high uncertainty. The bonuses were based on visit recency [Sutton, 1990], visit frequency [Barto and Singh, 1990], value prediction error [Moore, 1990, Schmidhuber, 1991], estimated modelling precision [Thrun and Möller, 1992], and optimistic initial values [Kaelbling, 1993].

Later, the exploration bonus developed into the interval estimation (IE) method [Kaelbling, 1993]. Based on statistical techniques, the confidence interval (CI), or in particular the upper limit of the CI, of value estimates are maintained and used to select actions. The CI upper limit $\tilde{Q}(s, a)$ of a value estimate $\hat{Q}(s, a)$ is always greater than or equal to the value estimate $\hat{Q}(s, a)$ itself, and gets closer to $\hat{Q}(s, a)$ as the sample size grows. Therefore, the upper limit $\tilde{Q}(s, a)$ of the IE method works as a bonus-infused value estimates like other strategies mentioned above, but has more solid mathematical backing than previous handcrafted bonuses.

Another branch of development in exploration bonus resulted in R-MAX [Brafman and Tennenholtz, 2002, Kakade, 2003], in which all state-action pairs (s, a) with less than m data points are labelled "unknown" and are optimistically assumed to have utility $\tilde{Q}(s, a) = \frac{R_{\max}}{1-\gamma}$, where $m > 0$ is a parameter. Recall that $\frac{R_{\max}}{1-\gamma}$ is the maximum possible utility for any MDP with rewards no greater than $R_{\max}$, the "unknown" state-action pairs thus always have $\tilde{Q}(s, a)$ no less than (and in most cases higher than) the "known" state-action paris (the ones with at least m data points). Thus, by choosing actions with maximum $\tilde{Q}(s, a)$, R-MAX guarantees exploration to the state-action pairs with insufficient data. In addition, different from early bonus-based methods where the bonuses

$f(s,a)$ do not propagate together with the estimated values $\hat{Q}(s,a)$, uncertainty in R-MAX propagates to other states through $\tilde{Q}(s,a)$, which makes it possible for the agent to look ahead and proactively move from the explored areas to the uncertain areas. This idea later re-emerged as "deep exploration" [Osband et al., 2016].

Direct descendants of R-MAX include Delayed Q-learning [Strehl et al., 2009], Optimistic Initial Model (OIM) [Szita and Lőrincz, 2008], MoR-MAX [Szita and Szepesvári, 2010], V-MAX [Rao and Whiteson, 2012], Increasingly Cautious R/V-MAX (ICR/ICV) [Zhang et al., 2015], and Directed Delayed Q-learning [Oh and Iyengar, 2018]. Delayed Q-learning is an adaptation of model-based R-MAX to model-free RL, while the others encourage earlier exploitation during exploration (e.g. by maximising a linear combination of optimistic $\frac{R_{\max}}{1-\gamma}$ and non-optimistic $\hat{V}$ in V-MAX, or by using a much smaller m in MoR-MAX).

The interval estimation method has also been extensively developed. This includes Model-Based Interval Estimation (MBIE) [Strehl and Littman, 2005], Upper Confidence Bound (UCB) [Auer et al., 2002], Upper Confidence Reinforcement Learning (UCRL) [Auer and Ortner, 2006], and their many variants (e.g. UCRL2 [Jaksch et al., 2010], UCCRL [Ortner and Ryabko, 2012], UCRL-Factored [Osband and Van Roy, 2014], UCRLγ [Lattimore and Hutter, 2014], UCBVI [Azar et al., 2017], and UCRL-V [Tossou et al., 2019]).

UCRL is also applied to the Monte-Carlo Tree Search (MCTS) [Sutton and Barto, 2018, Browne et al., 2012] method and leads to the widely-known Upper Confidence Tree (UCT) [Kocsis and Szepesvári, 2006]. UCT is common in Go-playing agents [Silver et al., 2016].

Systematic exploration strategies mentioned above require to keep track of the degree of uncertainty for each state-action pair, and thus had difficulty in applying to problems with large or continuous state/action spaces. Recently, many effort has been put to adapt them for deep RL. Bellemare et al. [2016] proposed the pseudo-count, which combines the count-based exploration of R-MAX with the ideas of interval estimation and intrinsic motivation, and achieved higher efficiency than the simple optimistic initialisation trick by Machado et al. [2015] in playing Montezuma's Revenge, one of the most difficult games of Atari 2600. Tang et al. [2017] proposed #Exploration which uses hash functions for count-based exploration to reduce computational cost. Martin et al. [2017] proposed ϕ-pseudocount which exploits the feature representation of deep neural networks. Pathak et al. [2017] used the prediction error of feature space as exploration bonus. O'Donoghue et al. [2018] proposed the uncertainty Bellman equation which is used to propagate the exploration bonus in deep RL as R-MAX and similar strategies in finite MDPs. Machado et al. [2020] utilised successor representation as exploration bonus.

Bayesian approach

Bayesian RL [Dearden et al., 1999, Guez et al., 2012, Ghavamzadeh et al., 2016] is another approach to deal with exploration vs exploitation. Here the unknown transition function P is assumed to be a random variable following some prior distribution $\mathbb{P}(P)$. After observing a trajectory ψ_t at time t, the posterior belief of P can be updated following Bayes' rule $\mathbb{P}(P|\psi_t) \propto \mathbb{P}(\psi_t|P)\mathbb{P}(P)$. The MDPs are augmented to Bayes-adaptive MDPs with the states being the combination of the original state information and the trajectory of the learning process. The transition of augmented states is an integral of the original transition P multiplied by their posterior $\mathbb{P}(P|\psi)$. Under such formulation, greedily following the optimal policy of a BAMDP yields maximum cumulative reward in its original MDP if the prior distribution of unknown P exactly matches the reality, which automatically removes the exploration vs exploitation dilemma of learning-while-serving RL. If this assumption does not hold, then the greedy policy of BAMDP is still optimal within Bayesian framework, but does not correspond to any non-Bayesian optimality and in general is worse than the true optimal policy in terms of cumulative reward [Ghavamzadeh et al., 2016].

Due to the formulation of optimality being drastically different to traditional RL, this thesis will not further consider the Bayesian approach to exploration, but only provides some pointers here: [Wyatt, 1998, Strens, 2000, Wang et al., 2005, Poupart et al., 2006, Kolter and Ng, 2009, Guez et al., 2013, Osband et al., 2016, Russo et al., 2017, Agrawal and Jia, 2017].

2.3.2 Analysis of exploration strategies

Theoretical efficiency of systematic exploration strategies has received much attention and efforts among the RL research community, and numerous strategies with theoretical guarantees have been proposed. The works on this topic can be put into two categories according to the metric of efficiency being used, namely sample complexity and regret.

Sample complexity analysis

The sample complexity analysis is also known as Probably Approximately Correct (PAC) analysis, which originated from [Valiant, 1984] for supervised learning and later adapted to episodic reinforcement learning in [Fiechter, 1994]. It was then applied to undiscounted fixed horizon RL in [Kearns and Singh, 2002] and the more common discounted infinite horizon RL in [Kakade, 2003].

Sample complexity analysis in RL mainly concerns the number of suboptimal steps in an infinitely long learning process. Step t is considered suboptimal if $V^*(S_t) - V^{\mathrm{algo}_t}(S_t) > \varepsilon$, where V^* is the optimal state value, V^{algo_t} is the state value of the algorithm at step t seen as a non-stationary policy, and $\varepsilon > 0$ is a parameter of near-optimality which specifies the allowance for “approximately correct”. PAC analysis of an algorithm gives an asymptotic bound to its sample complexity that holds for probability at least $1 - \delta$ under certain conditions on the parameter setting of the algorithm, where $\delta > 0$ is also a parameter of near-optimality which specifies the allowance for “probably correct”.

The bounds are usually polynomials of $|\mathcal{S}|$, $|\mathcal{A}|$, $\frac{1}{1-\gamma}$, $\frac{1}{\varepsilon}$, and $\frac{1}{\delta}$, where $|\mathcal{S}|$ and $|\mathcal{A}|$ are the sizes of the state/action sets, and γ is the discount factor of MDP. The conditions on algorithm parameters (directly or indirectly) specify how much exploration is needed. For example, Strehl et al. [2009] proved a sample complexity upper bound $O(\frac{|\mathcal{S}||\mathcal{A}|}{\varepsilon^3(1-\gamma)^6}(|\mathcal{S}| + \ln\frac{|\mathcal{S}||\mathcal{A}|}{\delta})\ln\frac{1}{\delta}\ln\frac{1}{\varepsilon(1-\gamma)})$ for R-MAX under condition $m = O(\frac{1}{\varepsilon^2(1-\gamma)^4}(|\mathcal{S}| + \ln\frac{|\mathcal{S}||\mathcal{A}|}{\delta}))$, where m is the parameter deciding how much data should be collected for each "unknown" state-action pair before it is deemed "known" (see Section 2.3.1).

Sample complexity analysis has been conducted to Explicit Explore or Exploit (E^3) [Kearns and Singh, 2002], R-MAX [Kakade, 2003, Strehl et al., 2006], MBIE and MBIE-EB [Strehl and Littman, 2005, 2008], OIM [Szita and Lőrincz, 2008], Delayed Q-learning [Strehl et al., 2009], MoR-MAX [Szita and Szepesvári, 2010], V-MAX [Rao and Whiteson, 2012], UCRLγ [Lattimore and Hutter, 2014], ICR and ICV [Zhang et al., 2015], and Directed Delayed Q-learning [Oh and Iyengar, 2018]. The best sample complexity upper bounds for finite MDPs so far are achieved by MoR-MAX and UCRLγ.

Sample complexity analysis has also been generalised to cases other than finite MDPs.[1] Strehl et al. [2007] studied the sample complexity in factored-state MDPs. Pazis and Parr [2013] provided a sample complexity analysis in continuous MDP for the C-PACE algorithm. Dann and Brunskill [2015] analysed the sample complexity in episodic fixed-horizon learning problems for Upper Confidence Fixed-Horizon (UCFH) algorithm. Pazis and Parr [2016] studied the case where multiple MDPs are available and can be explored concurrently. Krishnamurthy et al. [2016] provided sample complexity analysis in a contextual decision processes similar to partially observable MDPs. There is also a more general framework called Know What It Knows (KWIK) which applies not only to RL but also supervised learning and active learning [Li et al., 2011]. Kolter and Ng [2009] provided a PAC style analysis for a Bayesian algorithm called Bayesian Exploration Bonus.

Regret analysis

Regret analysis concerns the total loss of rewards the agent could have compared to an optimal policy in undiscounted MDPs ($\gamma = 1$) with finite horizon (i.e. cumulative rewards

[1]There are also researches (e.g. [Zou et al., 2019]) concerning finite sample analysis of RL that looks similar to PAC analysis but take entirely different approach to exploration. Such researches assume that every state can be reached from any other state under any policy being considered, which in fact implies *infinite* sample complexity in the majority of MDPs where good policies do not visit every state.

only consider first t steps). Formally, the regret of an algorithm executed for t steps is defined as $t\rho^* - G_t$, where ρ^* is the average (per-step) reward of the optimal policy, and G_t is the undiscounted cumulative reward of the algorithm till step t. Regret analysis of an algorithm gives an asymptotic bound to its regret that holds for probability at least $1 - \delta$ under certain conditions, where $\delta > 0$ similar to sample complexity analysis. The bounds are expressed in $|\mathcal{S}|$, $|\mathcal{A}|$, t, as well as some other metrics related the dynamics of the MDP (e.g. its diameter [Auer et al., 2009]).

Regret has mostly been studied under the undiscounted fixed-horizon setting. The regret upper bound of UCB1 algorithm for multi-armed bandit problems was studied in [Auer et al., 2002]. It was then generalised to general RL problems in [Auer and Ortner, 2006] for UCRL algorithm. Notable follow-up researches include [Auer et al., 2009, Jaksch et al., 2010] for UCRL2 algorithm, [Bartlett and Tewari, 2009] for REGAL algorithm in weakly communicating MDPs, [Ortner and Ryabko, 2012] for UCCRL algorithm in continuous MDPs, [Ortner et al., 2014] for Colored UCRL2 in restless Markov bandits, [Azar et al., 2017] for UCB-VI, and [Tossou et al., 2019] for UCRL-V which closed the gap between the upper bound and the best lower bound so far given by [Jaksch et al., 2010]. In addition, [Agrawal and Jia, 2017] and [Fruit et al., 2018] analysed the regret bounds of two Thompson sampling algorithms, optimistic PSRL and SCAL, respectively.

There is also a metric called average loss [Strehl and Littman, 2008] which is similar to the regret, but also has strong connection to sample complexity. It considers not $t\rho^* - G_t$ but $\sum_{j=1}^{t}(V^*(S_j) - G_j)$. The differences between the two is that the total optimal reward is trajectory-irrelevant in the former while trajectory-relevant in the latter, and that the former is for undiscounted MDPs while the latter is for discounted MDPs. It was proved in [Strehl and Littman, 2008] that if an algorithm has bounded sample complexity, then it also has small average loss within bounded t.

2.3.3 Benchmarking of exploration strategies

While many papers proposing new exploration strategies claimed theirs to be theoretically or empirically superior, benchmarking of the exploration strategies has repeatedly led to opposite conclusions. For example, in [Rao and Whiteson, 2012], experiments showed that MoR-MAX performed much worse than MBIE in SixArms and Anchor domains, and even worse than R-MAX in SixArms, in terms of learning-process cumulative reward and per-step reward, despite MoR-MAX having much better theoretical sample complexity. In [Kuleshov and Precup, 2014], exploration strategies for multi-armed bandits (single-state MDPs) are extensively tested, and results showed that simple strategies such as ε-greedy and Boltzmann selection significantly outperformed theoretically guaranteed strategies in both learning-process regret and percentage of optimal action in most settings. Empirical results of [Tijsma et al., 2016] showed that the theoretically guaranteed UCB-1 was outperformed in learning-process cumulative reward by Boltzmann selection in stochastic mazes. Recent exploration strategies inspired by the ones with theoretical guarantees and adapted to deep RL were benchmarked by Taiga et al. [2020] in the Atari 2600 environment. Their results showed that the new strategies were not significantly better in learning-process average score (episodic cumulative reward) than ε-greedy in games considered difficult, and performed even worse than ε-greedy in easier games.

2.3.4 Shortcomings of existing researches on exploration

It is interesting to observe such a remarkable gap between what is guaranteed by the theoretical analysis of exploration strategies and what is observed from empirical results. Our understanding about the reasons behind the gap is as follows.

The principal problem here is that the learning-process cumulative reward is not a good metric of sample efficiency. As shown in Figure 2.1 of Section 2.2, cumulative reward as a performance metric is biased towards myopic learners that exploits frequently since the early phase of learning, and is indifferent to how fast the agent finds out a desirable policy.

Random strategies such as ε-greedy and Boltzmann selection are more myopic and exploit earlier than systematic strategies, and thus are at an inherent advantage, especially when the task is sufficiently easy such that even random strategies can find out desirable policies without much effort.

What exactly, then, is guaranteed for the systematic strategies that follow the explore-sufficiently-before-exploiting idea? The answer is still a better learning-process cumulative reward, but the advantage is visible only in sufficiently long run against a sufficiently bad competitor, such that discovering a better policy faster actually pays off. Let us consider a variant of Example 2.1 presented in Section 2.2, given as follows:

Example 2.2. Consider a variant of Example 2.1 where Bob does not converge to the optimal policy (selecting only Yes) at $t = 151$, but selects Yes 9 times and No once in every 10 steps since $t = 1$, mimicking ε-greedy strategy with $\varepsilon = 0.1$. Alice still converges to the optimal policy at $t = 101$ as in the original example.

The undiscounted learning-process cumulative reward of the agents in this case is shown in Figure 2.2. Convergence to the optimal policy eventually pays off for the fast-learner Alice, but it takes a much longer time ($t \geq 1001$) compared to the time of convergence ($t = 101$), and the difference at $t = 1500$ is not even visually significant, where Alice wins by 1400 vs 1350, a mere 3.57% relative advantage. Note that in the discounted case with $\gamma = 0.97$, Alice still cannot beat Bob in learning-process cumulative reward, just as pointed out in Section 2.3.4. Considering that Alice has bounded sample complexity and regret (both are 100) while Bob has unbounded ones, it is clear how poor such theoretical guarantees translate into learning-process cumulative reward in average cases.

This puts the researches on sample efficiency in learning-while-serving workflow an awkward position. While RL users hope their algorithms learn fast, the ultimate objective of the learning-while-serving workflow not only does not care about how fast an algorithm finds a good policy, it sometimes even *punishes* algorithms for learning fast through active exploration. It is thus a misplaced idea to seek for high sample efficiency in learning-while-serving setting.

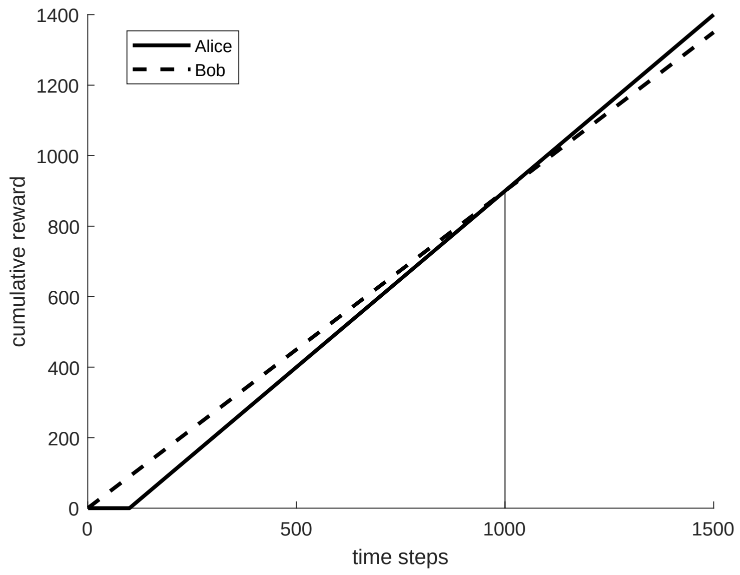

Figure 2.2: Cumulative rewards of two learning processes, where Alice converges to the optimal policy at $t = 101$ while Bob never does. Convergence to optimal does not pay off until $t \geq 1001$.

For the reasons discussed above and in Section 2.2, when sample efficiency is of primary concern, it is arguably better to consider a learning-then-serving workflow and avoid using learning-process cumulative reward as performance metric. In the learning-then-serving workflow, neither sample complexity analysis nor regret analysis are immediately usable in anticipating the exploration efficiency of algorithms due to their direct connection to the learning-process cumulative reward. Therefore, we need new tools for analysing sample efficiency of exploration strategies in the learning-then-serving setting. On the other hand, existing systematic exploration strategies can still work adequately as heuristics because they at least demand sufficient exploration before convergence. Meanwhile, since these heuristics are not designed to maximise learning-then-serving sample efficiency, there is still much room for improvement, especially when in tasks where rewards do not indicate good directions of exploration.

2.4 Biases of state/action value estimates

2.4.1 Bias (and variance) in statistics

Bias of a point estimator is defined as the difference between the expected value of the estimate and the true value being estimated [Casella and Berger, 2002]. Let $\hat{\theta}$ be an estimator of some true value θ, then $\text{Bias}(\hat{\theta}) \overset{\text{def}}{=} \mathbb{E}[\hat{\theta}] - \theta$ is the bias of estimator $\hat{\theta}$. An estimator is said to be unbiased if its bias is 0 (and thus $\mathbb{E}[\hat{\theta}] = \theta$) for all θ. Biases can occur due to various reasons, such as problematic estimator designs, lack of data, wrong sampling methods, and sometimes deliberate injection of biases for certain purposes.

Bias is sometimes discussed together with variance of estimators, which is defined as $\text{Var}(\hat{\theta}) \overset{\text{def}}{=} \mathbb{E}[(\hat{\theta} - \mathbb{E}[\hat{\theta}])^2]$. Bias describes the accuracy of an estimator, while variance describes the precision of an estimator.

Since the values being estimated can vary in magnitude and sign, it is difficult to compare the quality of estimators directly by their biases. Therefore, in this thesis, we also use the (signed) relative error $\frac{\hat{\theta}-\theta}{\theta}$ and the (absolute) relative bias $|\frac{\mathbb{E}(\hat{\theta})-\theta}{\theta}|$ ($\times 100\%$) to measure the magnitude of the estimation error and the bias, respectively.

2.4.2 Transformation bias of model-based RL

Although the Bellman equations 2.1,2.2 of state/action values are linear equations, the expressions of the solutions are nonlinear mappings from transition probabilities and rewards to state/action values. For example, given a certain policy, Equation 2.1 can be rewritten in matrix form as:

$$\mathbf{V} = (\mathbf{P} \odot \mathbf{R})\mathbf{1} + \gamma \mathbf{P}\mathbf{V},$$

where $\mathbf{V}$ is the column vector of state values under that policy, $\mathbf{P}$ and $\mathbf{R}$ are transition probability matrix and reward matrix under that policy, $\odot$ denotes Hadamard product, and $\mathbf{1}$ denotes an all-1 column vector. Then the expression of the solution is:

$$\mathbf{V} = (\mathbf{I} - \gamma \mathbf{P})^{-1} (\mathbf{P} \odot \mathbf{R}) \mathbf{1}.$$

This expression, seen as a function $V = g(P)$, is clearly nonlinear due to matrix inversion.

Generally, equation $\mathbb{E}[f(X)] = f(\mathbb{E}[X])$ holds if $f(X)$ is a linear function of random variable X. However, when $f(X)$ is nonlinear, very often $\mathbb{E}[f(X)] \neq f(\mathbb{E}[X])$. Specifically, by Jensen's inequality, if f is convex, then $\mathbb{E}[f(X)] \geq f(\mathbb{E}[X])$, and vice versa.

Now consider model-based RL where values are estimated by $\hat{V} = g(\hat{P})$. The transition probability estimator $\hat{P}$ defined as $\hat{P}(s'|s, a) \overset{\text{def}}{=} \frac{N_{s,a,s'}}{N_{s,a}}$ (see Section 2.1) is unbiased. Thus we have $\mathbb{E}[\hat{P}] = P$, which means $g(\mathbb{E}[\hat{P}]) = g(P) = V$. Therefore, the nonlinearity of g leads to $\mathbb{E}[\hat{V}] = \mathbb{E}[g(\hat{P})] \neq g(\mathbb{E}[\hat{P}]) = V$, which means that $\hat{V}$ is a biased estimator of V. To stress that the bias is due to the nonlinearity of transformation (used synonymously with "function"), we call it *transformation bias* of model-based RL in this thesis.

Transformation bias has been extensively discussed in [Mannor et al., 2004, 2007, Grünewälder and Obermayer, 2011]. Mannor et al. [2004, 2007] provided the expressions for the second order approximation to the transformation bias and parametric variance of model-based estimator. Their analysis suggested that the error caused by the variance is typically much larger than the bias, and thus bias reduction is relatively unimportant in general. The empirical result in a real-world mail catalogue problem with data from 1.72 million customers, reported in [Mannor et al., 2007], also showed that the bias of the value estimate was small, both in relative value and compared to the standard deviation (about 2.38% relative bias vs 5.26% standard deviation).

From a practical point of view, the results of [Mannor et al., 2004, 2007] have two major problems. First, their expression for second order approximation to the transformation bias relies on full knowledge of the true transition function P and reward function R. In most RL settings these are not available to the agent. Thus, to use the expression to compute the bias during learning, we have to use the estimated $\hat{P}$ and $\hat{R}$ to replace P

and R, just as what we do in computing $\hat{V}$. The expression of the bias is also a nonlinear function, which introduces *additional* bias to the transformation bias estimates, and thus is not suitable for bias correction purpose.

Second, their conclusion that the transformation bias is less important than the variance becomes invalid when sample size is small. As our empirical results presented in Chapter 4 show, the transformation bias can become very large when sample size is small. As will be elaborated later in Section 4.2, when the bias is sufficiently large to change the relation between two estimated values, variance reduction become useless or even harmful for value comparison. To make variance reduction useful, one should conduct bias correction first in such cases.

Grünewälder and Obermayer [2011] also analytically studied the transformation bias. They showed that any estimator satisfying the Bellman equations cannot be unbiased, unless the MDP is acyclic (i.e. there is no way to start from a state and travel back to it later), or the discount factor $\gamma = 1$ and an additional full information criterion (FI) is fulfilled. Notably, since absorbing states (states that lead to themselves under any action) also result in cycles in MDPs, being acyclic means any path in this MDP must eventually lead to a state that has no action available (otherwise by the pigeon hole principle there must be cycles). Such cases are usually discussed within the episodic MDP framework. The other condition of unbiasedness requires $\gamma = 1$, which is never satisfied in discounted MDPs where $0 < \gamma < 1$. Therefore, in MDPs within the scope of this thesis, any algorithm satisfying the Bellman equations suffers from transformation bias.

In addition, Grünewälder and Obermayer [2011] derived a Minimum Variance Unbiased estimator (MVU) based on first-visit Monte-Carlo estimation (MC), which is unbiased, but is not a model-based RL method and does not use the Bellman equations. Since in this thesis we focus on model-based RL based on the Bellman equations, the MVU method is out of our scope and thus will not be further discussed.

2.4.3 Other biases in RL

Although transformation bias of model-based RL is one of the main focuses of this thesis, there are also several other biases in RL. This section shortly reviews relevant researches and discusses their relationship with ours.

Maximisation bias

Maximisation bias is a bias caused by the use of max operator in value estimators, which can happen in both model-free and model-based RL that utilise the Bellman optimality equations (see Equations 2.3,2.4). The use of max operator in value estimates results in overestimation, which can be shown by Jensen's inequality, e.g. $\mathbb{E}[\max_a \hat{Q}(s,a)] \geq \max_a \mathbb{E}[\hat{Q}(s,a)]$. The bias was first systematically studied in [Ormoneit and Sen, 2002]. Mannor et al. [2007] proposed a cross-validation procedure to deal with the maximisation bias. Van Hasselt [2010] proposed a similar method called Double Q-learning and showed that this algorithm results in underestimation in most cases. Double Q-learning was adapted to deep RL by [Van Hasselt et al., 2016] and improved by [Fujimoto et al., 2018]. Fox et al. [2016] proposed a different method called G-learning, which does not reduce the bias directly, but tries to avoid converging to inferior policy due to the bias by directly penalising early convergence (which is essentially an optimistic exploration strategy).

Maximisation bias and transformation bias occur independently in model-based RL. In model-based RL that does not use the Bellman optimality equations directly (e.g. algorithms that conduct policy improvement and value estimation separately), maximisation bias does not occur due to the max operator being absent in value estimation, while transformation bias still occurs. On the other hand, in rare occasions where MDP is acyclic, by [Grünewälder and Obermayer, 2011] there is no transformation bias, but algorithms using the Bellman optimality equations still suffers from maximisation bias. In other cases, the two biases exist simultaneously and independently.

“Model bias”

The term “model bias” [Deisenroth and Rasmussen, 2011, Kurutach et al., 2018, Janner et al., 2019, Grover et al., 2019] refers to the differences between the the leant model MDPs and the true environment MDPs. In recent years, it has been a relatively popular research topic in deep reinforcement learning because deep neural networks can easily overfit the data and make wrong predictions at the state-action pairs where data is scarce, which in turn is exploited by the planning algorithm to yield policies that have high estimated utility but low actual utility. It is in fact a generalisation error which occurs in learning model MDPs using supervised learning techniques, rather than a bias in statistical sense.

The relationship between the “model bias” and the transformation bias can be somewhat confusing, and thus will be clarified here. Let $|M_1 - M_2|$ be the difference under a certain measurement between two arbitrary MDPs M_1 and M_2. Let M be the true unknown MDP and $\hat{M}$ be a model MDP learnt from some data using some supervised learning techniques. The “model bias” issue often refers to having a large $|M - \hat{M}|$ which leads to a large $|V - \hat{V}|$. Since no matter how good the algorithm is, a large $|M - \hat{M}|$ can still happen due to bad luck in environment interaction, it is more reasonable to consider reducing the expectation $\mathbb{E}[|M - \hat{M}|]$. Then the best (in terms of bias) supervised learning algorithm should achieve $\mathbb{E}[|M - \hat{M}|] = 0$.

Remarkably, the existence of transformation bias means that $\mathbb{E}[|V - \hat{V}|] > 0$ happens regardless of whether $\mathbb{E}[|M - \hat{M}|] = 0$, because the nonlinearity of the mapping from $\hat{M}$ to $\hat{V}$ breaks the unbiasedness of $\hat{V}$. In other words, having “optimal” supervised learning algorithms for learning models does not automatically remove the bias of value estimates. What is worse, due to the nonlinear relation, reducing the bias of model does not necessarily reduce the bias of value estimates. Thus, transformation bias correction is a necessary condition to guarantee the usefulness of model accuracy improvement.

Off-policy bias

Off-policy bias is common in model-free off-policy value evaluation where state/action values are estimated using the data generated by some behaviour policy different from the target policy being estimated. Since model-free methods rely on the update frequencies of state/action values to simulate the effect of transition probabilities in value estimation, the mismatch between the two policies effectively changes the transition probabilities of the Markov process and thus introduces significant bias. Off-policy bias is often dealt with by the Importance Sampling (IS) method, first applied to RL by [Precup et al., 2000]. Several IS methods were reviewed and empirically compared together with model-based RL in [Păduraru et al., 2013]. Some recent works on this topic can be found in [Jiang and Li, 2016, Munos et al., 2016, Thomas and Brunskill, 2016, Doroudi et al., 2017, Hanna, 2019].

Although IS methods usually claim themselves to be unbiased, such unbiasedness is often with respect to the off-policy bias but not the transformation bias. [Păduraru et al., 2013] pointed out that IS methods need the behaviour policy to be fixed and known, and that if the behaviour policy is not known but estimated from data then IS becomes equivalent to model-based RL. Thus, when applied to full-scale RL where behaviour policies are always (at least partly) results of estimation, it is likely that IS methods suffer from a similar transformation bias as model-based RL, which cannot be resolved solely by IS. Meanwhile, model-based RL does not suffer from off-policy bias because it does not require the matching of behaviour and target policies.

2.5 Value comparison and policy selection

Although there is rich literature regarding improving the accuracy of state/action value estimates, there is little research on how to compare the estimates to make decisions. Most researches use naive comparison, i.e. if $\hat{V}_1 > \hat{V}_2$ then choose action/policy that corresponds to $\hat{V}_1$. As will be elaborated in Chapter 5, such method is not reliable due to the various biases in the state/action value estimates as well as the asymmetry of the distributions, especially when sample size is small.

This section reviews the work related to value comparison and policy selection.

2.5.1 Fairness of policy selection

To the best of our knowledge, [Doroudi et al., 2017] is the only published work that discussed the problem where an algorithm can select a worse policy more often than a better policy. They defined the fairness of policy selection as follows: an algorithm is said to be fair if the probability of it selecting the optimal policy (with respect to true state/action values) is greater than any other policies. In the case of comparing two policies, being fair means the probability of selecting the optimal policy is at least 0.5. Algorithms are allowed to refuse to output a policy if they have low confidence in their own guesses, and thus the trivial algorithm that never outputs a policy is fair under their definition.

They analysed the fairness of Monte-Carlo and Important Sampling algorithms and showed that neither of them are fair. Specifically, they showed that IS favours myopic policies as well as policies that produce shorter trajectories. They then proposed the MC-fairness which requires the selection method to choose more frequently the policies that have higher Monte-Carlo estimated values (and thus is easier to achieve than the strict fairness). Two methods to improve the fairness of IS estimators were provided.

The true/false nature of the fairness makes it difficult to use when comparing two methods that are both fair or both unfair. Also, the permission to not output policies is somewhat impractical, in the sense that if an algorithm refuses to output policies often then its real-world usefulness can be highly questionable (especially when restarting the whole learning process is required to attempt to change the result).

Although their work focused on IS which is out of scope of this thesis, the definition of fairness is closely related to what we are interested in, and thus will be further discussed in Chapter 5.

There is also a work by Jabbari et al. [2017] which discussed fair action selection in RL. Their work required that a fair algorithm never chooses action a with probability higher than another action a' unless $Q^*(s, a) > Q^*(s, a')$. However, their Q^* in fact refers to the *estimated* action value $\hat{Q}^*$, and thus the fairness requires the algorithm to frequently select actions with higher estimated values. They found that this requirement strongly limits the exploration ability of an algorithm, and thus proposed two relaxations of fairness and an algorithm called Fair-E^3 which was shown to satisfy the relaxed versions of fairness.

From our point of view, the fairness by Jabbari et al. [2017] is just requiring a high degree of greediness during learning, which is improper in learning-then-serving setting where exploration efficiency is of primary concern, while in learning-while-serving setting the usefulness is also questionable because it neither directly leads to a higher learning-process cumulative reward nor has any relevance to real-world fairness. Thus, we will not consider their definition in the remainder of this thesis.

2.5.2 Relationship with maximisation bias

The maximisation bias discussed in Section 2.4.3 is ostensibly related to policy selection, in the sense that both the maximisation operator in the Bellman optimality equation and the policy selection process involves comparing two or more estimated state/action values.

However, the purpose of the value comparison in these two scenarios are different. The output of the maximisation operator in the Bellman optimality equation is the maximum estimated value for that state or state-action pair. Thus, the correction of the maximisation bias aims to improve the accuracy of the maximised estimated value, while whether that value corresponds to a better action/policy is not of primary concern.

On the other hand, the output of the policy selection process we consider in Chapter 5 is a policy, and thus the effectiveness of the process is evaluated by the quality of the output policy. The accuracy of the estimated value is only a factor that might or might not positively affect the effectiveness of policy selection. In fact, we will show in Chapter 5 that an algorithm with an unbiased value estimator can still suffer from poor selection.

Since both the purposes and the outputs of the two process are very different, in this thesis we will not go into detail on the maximisation bias.

2.6 Bootstrap

Bootstrap in statistics is a resampling technique first proposed by [Efron, 1979] as a more widely applicable alternative to the jackknife technique proposed by Quenouille and Tukey [Quenouille, 1956, Miller, 1974]. An abundant literature is available on bootstrap, jackknife, and their application to bias correction; see [Quenouille, 1956, Davison and Hinkley, 1997, Rodgers, 1999, Shao and Tu, 2012, Jiao and Han, 2020] for some pointers.

The word "bootstrapping" in RL usually refers to the idea of estimating $V(s)$ using other value estimates $V(s')$. This occurs naturally when using the Bellman equations in value estimation, and thus is extremely common in both model-based and model-free RL, but not so in Monte-Carlo methods because these methods do not rely on the Bellman equations. See [Sutton and Barto, 2018] for more information about "bootstrapping" in this sense. This "bootstrapping" idea is very different from the statistical technique mentioned above and it *does not* correct biases of value estimates. In this thesis, except for this paragraph, the term "bootstrap" is always used in the first sense (the resampling technique).

Statistical bootstrap has been used in RL mainly for generating data and estimating confidence intervals. Some examples of such application can be found in [White and White, 2010, Thomas et al., 2015, Hanna, 2019].

2.7 Chapter Summary

This chapter provides the background knowledge of model-based RL and bootstrap, discusses the important differences between learning-while-serving and learning-then-serving settings, reviewed related works on exploration, value estimation biases, and value comparison in policy selection, and pointed out the shortcomings of existing researches. The shortcomings are summarised in the following list.

- Exploration of RL has mainly been studied under the learning-while-serving setting, where learning-process cumulative reward needs to be maximised. However, in real-world applications where sample efficiency is crucial, RL is often conducted in a learning-then-serving workflow, where luearning-process cumulative reward is irrelevant. Systematic exploration strategies still work adequately as heuristics in this case, but also miss much chances to do significantly more efficiently. Existing analysis of exploration efficiency also becomes unsuitable because the efficiency is measured by metrics that are based on learning-process cumulative reward.

- Transformation bias has been considered insignificant, and few research has been made for correcting the bias in model-based reinforcement learning. However, our empirical results show that when the sample size is small, not correcting the transformation bias can significantly reduce the usefulness of value estimates, making successful learning very difficult. The second-order approximation of the transformation bias proposed by [Mannor et al., 2007] is not suitable for bias correction when the sample size is small, because their approximation by itself suffers from a similar transformation bias. The MVU estimator proposed by [Grünewälder and Obermayer, 2011] is a Monte-Carlo method and is not directly applicable to model-based RL.

- To the best of our knowledge, there is no systematic study on the policy selection of model-based RL. Existing methods rely on naive policy selection, which has been shown by [Doroudi et al., 2017] for Importance Sampling based RL to have high risk of selecting inferior policies, and suffers from the same problem in model-based RL according to our analysis, especially when sample size is small. In addition, the fairness notion proposed by [Doroudi et al., 2017] is too strict and is practically unusable as a metric, which means that there is neither a good framework for analysing and comparing the effectiveness of policy selection methods, nor a method that can improve it for model-based RL.

CHAPTER 3

EXPLICIT PLANNING FOR EFFICIENT EXPLORATION

3.1 Introduction

As reviewed in Section 2.3, exploration is a key factor to the sample efficiency of reinforcement learning algorithms. Systematic exploration strategies aim to improve sample efficiency by encouraging active exploration when data is not sufficient. However, the execution of such an idea has been more or less problematic in both learning-while-serving workflow and learning-then-serving workflow. In the former case where early exploitation is strongly favoured, conducting early exploration leads to a distinct disadvantage in terms of learning-process cumulative reward for systematic strategies. In the latter case where learning-process cumulative reward is irrelevant, the still-existing exploitation behaviours make systematic strategies less efficient.

In this chapter, we consider the sample efficiency of exploration strategies in the learning-then-serving workflow. Since learning-process cumulative reward becomes unimportant in this case, the cumulative-reward-based metrics reviewed in Section 2.3.2 are no longer suitable. Thus, we need a sample-size-based metric for evaluating strategies, as well as a method for analysing it. We can then use it to find out the weaknesses of existing strategies, and see if there is a way to avoid such weaknesses.

For these purposes, we will formulate the planning for exploration (PFE) problem in Section 3.2, where the efficiency of an exploration scheme is evaluated by its exploration cost, and the optimal exploration scheme of a PFE problem gives the most sample efficient exploration behaviour in the corresponding MDP. We show that the optimal exploration scheme of a PFE problem can be found by solving an augmented MDP generated from the original MDP. An algorithm called Value Iteration for Exploration Cost (VIEC) will be proposed in Section 3.3 to solve such augmented MDPs. With the optimal exploration schemes available, we can compare them with the exploration behaviours of existing strategies to find out their weaknesses. We will use a class of tower MDPs to analyse the exploration behaviour of ε-greedy, R-MAX, MBIE-EB, and optimal exploration scheme, and compare their exploration costs in Section 3.4. Section 3.5 will give a summary to this chapter.

The work presented in this chapter is published as [Zhang et al., 2019].

3.2 Formulation of the planning for exploration problem

3.2.1 The two-phase decomposition of exploration and the data demands

As we discussed above, we need a sample-size-based metric for evaluating the sample efficiency of exploration strategies. An immediate question is: why cannot we simply use the total sample size when the algorithm converges as the metric? The answer is that it is in fact very tricky in RL to confirm whether an algorithm has converged or not. The estimated state/action values and the output policy might keep unchanged for a very long time, but then change abruptly after some new information come in. Such abrupt changes occur less frequently when the sample size is sufficiently large, but obtaining a large sample just for confirming the convergence is counterproductive in a limited budget setting. Thus, to evaluate the exploration efficiency, we need a different stopping criteria for evaluation.

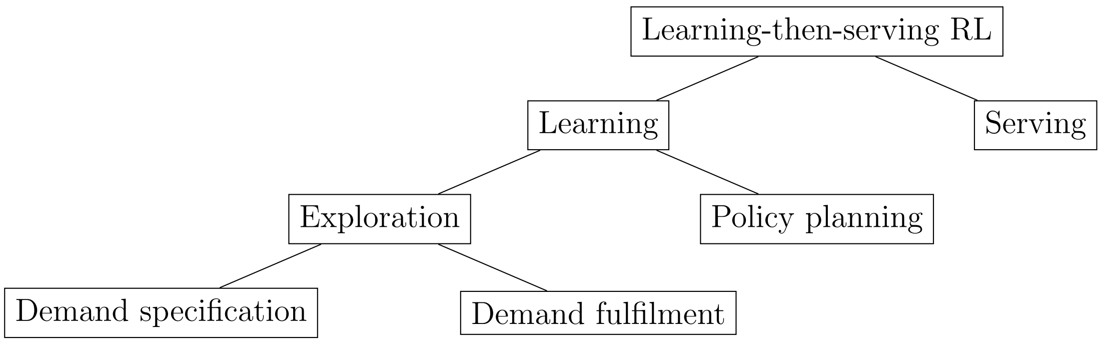

Figure 3.1: Decomposition of RL.

To explain the new criteria, we first introduce our three-level decomposition of RL. RL in the learning-then-serving workflow can be seen as a two-phase process, where in the first phase (learning) the agent learns a policy, then in the second phase (serving) the agent executes its policy to provide service. The learning phase itself can be seen as another two-phase process, where in the first phase (exploration) the agent explores the environment to collect data, and in the second phase (policy planning) the agent creates a model MDP using the collected data and solve the MDP to obtain the policy. The exploration phase can further be decomposed to a two-phase process, where in the first phase (*demand specification*) a requirement to the data collection is specified, and then in the second phase (*demand fulfilment*) the agent satisfies the requirement through the actual environment interaction. The pair of phases at each level of decomposition can either occur only once or be repeated for several times, according to the design of the algorithms. Figure 3.1 gives an illustration to the decomposition.

In the demand specification phase, the agent is informed of how much data should be collected through the subsequent environment interaction. Formally, we use *data demand* (or demand for short) to represent the requirement to the data collection, defined as follows.

Definition 3.1. In an MDP with $|\mathcal{S}|$ states and $|\mathcal{A}|$ actions, a data demand D is an $|\mathcal{S}| \times |\mathcal{A}|$ matrix where entry $D[s, a] = k \geq 0$ indicates that for state s and action a, at least k more data should be collected in the subsequent demand fulfilment process.

After a demand matrix is specified and passed to the agent to fulfil, in the subsequent demand fulfilment process the agent collects data through environment interaction. Whenever an additional data point is collected, the remaining demand reduces accordingly. For clarity, we write D_t to indicate the demand matrix when the agent enters time step t. After some action $A_t = a$ is executed at state $S_t = s$, the agent obtains a data point (s, a, s', r), and thus the corresponding entry in the demand matrix $D_t[s, a]$ is reduced by 1 if it is not already 0, while other entries remain unchanged. That is,

$$D_{t+1}[s,a] = \begin{cases} \max\{0, D_t[s,a] - 1\} & (s,a) = (S_t, A_t) \\ D_t[s,a] & \text{otherwise.} \end{cases}$$

We define the following demand reduction function H for convenience:

$$H(D; s, a) \overset{\text{def}}{=} \begin{cases} D - \boldsymbol{e}_{s,a} & D[s,a] > 0 \\ D & D[s,a] = 0, \end{cases}$$

where $\boldsymbol{e}_{s,a}$ is an $|\mathcal{S}| \times |\mathcal{A}|$ single-entry matrix whose entries are all zero except for $\boldsymbol{e}_{s,a}[s,a] = 1$. Then the change from D_t to D_{t+1} above can be simply denoted as $D_{t+1} = H(D_t; S_t, A_t)$.

Let D_{init} be the initial data demand of a demand fulfilment process. As the process proceeds, the data demand decreases gradually (i.e. $D_1 = D_{\text{init}} \geq D_2 \geq D_3 \geq \ldots \geq D_t$), and should eventually reduce to an all-zero matrix $\mathbf{0}$, which indicates the fulfilment of the initial demand D_{init} and thus the completion of the demand fulfilment process. We can therefore treat the completion of the demand fulfilment process as the stopping criteria, and evaluate the efficiency of exploration by considering the cost of the demand fulfilment process. Since the distribution of the data obtained from exploration decides the distribution of the output policy in the policy planning process, the fulfilment of a proper data demand guarantees the quality of the policy in probability. In this way, we avoid the difficulty in checking the convergence of policy and value estimates mentioned at the beginning of this section, and make it possible to define an unambiguous sample-size based metric for evaluating the exploration efficiency, which will be given in the next section.

The above discussion is based on the premise that the data demand is somehow determined and passed to the learning agent. Readers might ask: where does the demand come from and who determines it? The answer lies in the design of systematic strategies. In the researches on sample complexity reviewed in Section 2.3.2, concentration inequalities such as Hoeffding's and Chernoff's were used to analyse how much data is sufficient for achieving an $(\varepsilon, \delta)-$optimality with respect to sample complexity. Thus, by setting the relevant parameters of these strategies, a data demand is either explicitly or implicitly passed to the agent, while during the learning process the agent is guided to fulfil that data demand by some $\tilde{Q}$ defined by the exploration strategy.

For example, [Strehl et al., 2009] showed that $m = O(\frac{1}{\varepsilon^2(1-\gamma)^4}(|\mathcal{S}| + \ln \frac{|\mathcal{S}||\mathcal{A}|}{\delta}))$ is needed for R-MAX to achieve polynomial sample complexity, where m specifies how many times a state-action should be visited before a non-optimistic value evaluation is made for that state-action pair. Thus, by following R-MAX, the agent visits every state-action pair at least m times before the algorithm converges[1]. This directly translates to an initial demand matrix D_{init} with all entries being m.

For IE methods such as MBIE and UCRL, their parameters do not directly state the data demand, but only specify the confidence level. However, the confidence intervals are still made from concentration inequalities, and thus the demand can be mathematically inferred from the confidence level parameter. For example, Strehl and Littman [2008] showed that with a proper setting to the confidence level, MBIE guides the agent to visit every state-action pair for $O(\frac{1}{\varepsilon^2(1-\gamma)^4}(|\mathcal{S}| + \ln \frac{|\mathcal{S}||\mathcal{A}|}{\varepsilon(1-\gamma)\delta}))$ times, which is slightly more than R-MAX. Thus, the confidence level of IE methods is an indirect specification to the data demand.

As discussed in Section 2.3.4, the sample complexity analysis is made for the learning-while-serving workflow, and thus the demand specifications above are not directly appli-

[1]Due to the design of R-MAX there are some special cases where exploration stops prematurely, but this can be avoided by setting the parameter $R_{\max}$ to a value larger than the actual maximum reward of the MDP.

cable to the learning-then-serving workflow we are discussing in this thesis. It is theoretically possible to adapt these analyses to provide suitable parameter settings for the learning-then-serving workflow. We can then use these parameter settings to infer the data demands specified by these parameter settings, just as has been done in the above-mentioned analysis.

However, we will not conduct such analysis here due to it being out of scope of this chapter. Our main point in this chapter is, *no matter what* data demand is given to the agent, there usually exists a more efficient way to fulfil it than following the existing exploration strategies. Therefore, in the remainder of this chapter, we simply regard the initial demand D_{init} as known and fixed whenever the agent enters the demand fulfilment process, while do not care about how such specification is made or whether it is good or not under any standard (except that the demand should be satisfiable, which is given in the next section).

3.2.2 The undiscounted augmented MDP of the PFE problem

The execution of demand fulfilment is highly important to the overall sample efficiency. The undesirable consequence of thoughtless execution is presented intuitively in the following example.

Example 3.1. Suppose that you live in London and are curious about how it feels like to travel between London and Tokyo, and between Tokyo and Sydney. You decide to try out the outbound and return flights A and A' between London and Tokyo, as well as the outbound and return flights B and B' between Tokyo and Sydney. Now, consider two travel plans:

1. Take flight A to Tokyo, then B to Sydney, then B' back to Tokyo, and then A' back to London.
2. Take flight A to Tokyo, then A' back to London, then A to Tokyo again, then B to Sydney, then B' and A' back to Tokyo and London.

The first plan is more efficient because it only has 4 journeys while the second has 6.

Although the low efficiency of the second plan is obvious, it can actually be taken by those carefully-designed systematic strategies, which will be elaborated in Section 3.4. In the remainder of this section, we formulate the problem of finding good execution of demand fulfilment process as the planning for exploration (PFE) problem.

Let $\mathcal{D}$ denote the demand space, i.e. the set of all possible demand matrices in a learning task. Given an initial demand D_{init}, since all demand matrices satisfy $\mathbf{0} \leq D \leq D_{\text{init}}$, the size of the demand space is at most $(d+1)^{|\mathcal{S}||\mathcal{A}|}$, where $d = \max_{s,a} D_{\text{init}}[s, a]$.

The behaviour of an agent in demand fulfilment can be described as a mapping from demands and states to actions, which we call an exploration scheme, formally defined as follows.

Definition 3.2. An exploration scheme ϕ is a mapping $\mathcal{D} \times \mathcal{S} \mapsto \mathcal{A}$, where $\phi(D; s) = a$ indicates that action a should be taken when the demand is D and state is s.

Given a demand matrix D, we are interested in evaluating the efficiency of exploration schemes fulfilling D. Since each step of environment interaction produces one data point, the total sample size of a learning process equals to its total number of steps. When obtaining data from environment interaction is expensive, an efficient exploration scheme should be able to fulfil the demand with number of steps as few as possible. Thus, we can use the expected number of steps required to fulfil a demand to evaluate the efficiency of an exploration scheme.

To compute the total number of steps required to fulfil a given demand, we first formulate the undiscounted augmented MDP of the planning for exploration (PFE) problem as follows.

Definition 3.3. Given MDP $M = (\mathcal{S}, \mathcal{A}, P, R, \gamma)$ and demand space $\mathcal{D}$, the corresponding undiscounted augmented MDP of the planning for exploration problem is given by $M^{\Delta} \overset{\text{def}}{=} (\mathcal{S}^{\Delta}, \mathcal{A}, P^{\Delta}, R^{\Delta})$, where $\mathcal{S}^{\Delta} \overset{\text{def}}{=} \mathcal{D} \times \mathcal{S}$ is the augmented state space, $P^{\Delta} : \mathcal{S}^{\Delta} \times \mathcal{A} \times \mathcal{S}^{\Delta} \mapsto$

$[0,1]$ is the augmented transition function s.t.

$$P^{\Delta}(D', s'|D, s, a) = \begin{cases} P(s'|s, a) & D' = H(D; s, a) \\ 0 & D' \neq H(D; s, a), \end{cases}$$

and $R^{\Delta} : \mathcal{S}^{\Delta} \times \mathcal{A} \times \mathcal{S}^{\Delta} \mapsto \{0, 1\}$ is the augmented punishment function s.t.

$$R^{\Delta}(D', s'|D, s, a) = \begin{cases} 1 & D \neq \mathbf{0} \\ 0 & D = \mathbf{0}. \end{cases}$$

The definition of the augmented transition P^{Δ} above is simply stating that the probability of a transition in augmented state space $\mathcal{S}^{\Delta}$ equals to its original transition probability if the change of the demand is legitimate, and 0 otherwise. The augmented punishment R^{Δ} is 1 when the demand is not fully fulfilled, and 0 otherwise. Note that by definition if at some time step $t = \tau$ there is $D_{\tau} = \mathbf{0}$, then for all $t \geq \tau$ we have $D_t = \mathbf{0}$ and thus $R^{\Delta}_t = 0$. Therefore, the undiscounted cumulative augmented punishment $R^{\Delta}_1 + R^{\Delta}_2 + ...$ equals to the number of steps in a trajectory before D_t becomes $\mathbf{0}$ for the first time, which is exactly what we want to know.

We can thus use the expectation of the undiscounted cumulative punishment in the PFE problem to measure the efficiency of an exploration scheme fulfilling a given demand in the original MDP.

Definition 3.4. The (state) exploration cost $U^{\phi}(D; s)$ is the expected undiscounted cumulative punishment obtained by starting from demand D, state s, and following ϕ thereafter, i.e. $U^{\phi}(D; s) \overset{\text{def}}{=} \mathbb{E}[\sum_t R^{\Delta}_t | D_1 = D, S_1 = s, A_k = \phi(D_k; S_k)\ (k \geq 1)]$.

Definition 3.5. The (action) exploration cost $C^{\phi}(D; s, a)$ is the expected undiscounted cumulative punishment obtained by starting from demand D and state-action (s, a) then following ϕ thereafter, i.e. $C^{\phi}(D; s, a) \overset{\text{def}}{=} \mathbb{E}[\sum_t R^{\Delta}_t | D_1 = D, S_1 = s, A_1 = a, A_k = \phi(D_k; S_k)\ (k \geq 2)]$.

Lemma 3.1. *Let $\delta_{D,\mathbf{0}}$ be the Kronecker delta which equals to 1 if $D = \mathbf{0}$ and 0 otherwise. The following Bellman equations for the PFE problem holds.*

$$U^{\phi}(D;s) = (1 - \delta_{D,\mathbf{0}}) + \sum_{s' \in \mathcal{S}} P\big(s'|s, \phi(D;s)\big)\, U^{\phi}\Big(H\big(D;s,\phi(D;s)\big);s'\Big), \tag{3.1}$$

$$C^{\phi}(D;s,a) = (1 - \delta_{D,\mathbf{0}}) + \sum_{s' \in \mathcal{S}} P(s'|s,a)\, C^{\phi}\Big(H(D;s,a);s',\phi\big(H(D;s,a);s'\big)\Big). \tag{3.2}$$

Proof. Since the augmented MDP of PFE is an MDP in itself and the exploration costs are its corresponding utility functions, the above equations can be obtained by replacing s with $(D;s)$, P with P^{Δ}, R with R^{Δ}, π with ϕ, V with U, Q with C, and γ with 1 in Equations 2.1 and 2.2, then reducing the equations using Definition 3.3. $\square$

In MDPs with bad connectivity, it is possible to assign demands that are impossible to fulfil. For example, consider a two-state one-action MDP where s_1 leads to s_2 while s_2 leads to itself, both with probability 1. Any demand with $D[s_1, a] \geq 2$ can never be fulfilled because it is not possible to visit s_1 twice in one trajectory. Since it is practically meaningless to assign a demand that is impossible to fulfil, in the remainder of this chapter we assume all demands to be *satisfiable* (unless stated otherwise), defined as follows.

Definition 3.6. A demand D is said to be satisfiable if for any $s \in S$ there exists some exploration scheme ϕ such that $U^{\phi}(D;s) < \infty$.

3.2.3 Optimality in PFE

Let Φ be the set of all possible exploration schemes for a given PFE problem. Since the objective of the PFE problem is to minimise exploration cost, the optimal exploration scheme should have the smallest exploration cost at any state and demand.

Definition 3.7. An exploration scheme is said to be optimal and denoted ϕ^* if it satisfies $C^{\phi^*}(D; s, a) = \min_{\phi \in \Phi} C^{\phi}(D; s, a)$ for any $D \in \mathcal{D}$, $s \in \mathcal{S}$, and $a \in \mathcal{A}$.

The existence of ϕ^* and the uniqueness of C^{ϕ^*} and U^{ϕ^*} given a PFE problem with satisfiable demands can be proved by simply adapting the corresponding lemmas for general MDPs (see any textbooks, e.g. [Sutton and Barto, 2018]) to the augmented MDP of PFE, and thus will not be repeated here. For convenience we write C^{ϕ^*} as C^* and U^{ϕ^*} as U^*.

Lemma 3.2. *Let $\delta_{D,\mathbf{0}}$ be the Kronecker delta which equals to 1 if $D = \mathbf{0}$ and 0 otherwise. The following Bellman optimality equations for the PFE problem holds.*

$$U^*(D; s) = (1 - \delta_{D,\mathbf{0}}) + \min_{a \in \mathcal{A}} \sum_{s' \in \mathcal{S}} P\big(s' | s, a\big)\, U^*\Big(H\big(D; s, a\big); s'\Big), \tag{3.3}$$

$$C^*(D; s, a) = (1 - \delta_{D,\mathbf{0}}) + \sum_{s' \in \mathcal{S}} P(s' | s, a) \min_{a' \in \mathcal{A}} C^*\Big(H(D; s, a); s', a'\Big). \tag{3.4}$$

Proof. By combining Lemma 3.1 with Definition 3.7. □

By solving the Bellman optimality equations for PFE above, we can obtain the optimal exploration scheme of a PFE problem, which is elaborated in the next section.

3.3 Solving the planning problem

Since the undiscounted augmented MDP of the PFE problem is a variant of general MDPs, we can modify the Value Iteration algorithm to solve the Bellman optimality equations for PFE.

Note that the new demands $H(D; s, a)$ at the right-hand side of the Bellman optimality equations for PFE satisfies $H(D; s, a) \leq D$ by definition, and thus we can arrange all $D \leq D_{\text{init}}$ by a topological ordering s.t. for any demands $D_{(i)}$ and $D_{(j)}$, $D_{(i)} < D_{(j)}$

implies $i < j$. We can then iterate D only once, from the all-zero $D = \mathbf{0}$ up to the initial demand $D = D_{\text{init}}$, to compute all exploration costs, because when we compute the exploration cost for some $D_{(j)}$, all exploration costs with $D_{(i)}$ $(i < j)$ that are necessary for computing $D_{(j)}$ have already been computed.

A simple choice of topological ordering is to treat the matrices as $(|\mathcal{S}| \times |\mathcal{A}|)$-digit numbers with radix $d = \max_{s,a} D_{\text{init}}[s,a]$ and use the natural ordering, i.e. $D_{(0)} = (0...0; ...; 0...0)$, $D_{(1)} = (0...0; ...; 0...1)$, $D_{(2)} = (0...0; ...; 0...2)$, ... , $D_{(|\mathcal{D}|-1)} = D_{\text{init}}$. We can also skip the computation for $D = 0$ because their exploration costs are all 0 by definition.

The pseudocode of the Value Iteration for Exploration Cost (VIEC) algorithm is presented in Algorithm 1.

Algorithm 1 Value Iteration for Exploration Cost (VIEC)

Input: Initial demand D_{init}, transition P
Parameter: Stopping threshold ϵ_0
Output: Optimal exploration scheme ϕ^*

```
 1: Initialise all $C(D; s, a) = 0$, $U(D; s) = 0$
 2: Create the list of demand matrices $D_{(0)}, ..., D_{(|\mathcal{D}|-1)}$, arranged in natural ordering
 3: for $i = 1$ to $|\mathcal{D}|-1$ do
 4:   repeat
 5:     $\epsilon = 0$
 6:     for $s \in \mathcal{S}$ do
 7:       for $a \in \mathcal{A}$ do
 8:         $c = 1 + \sum_{s'} P(s'|s,a)\, U\big(H(D_{(i)}; s, a); s'\big)$
 9:         $\epsilon = \max\{\epsilon,\ |C(D_{(i)}; s, a) - c|\}$
10:         $C(D_{(i)}; s, a) = c$
11:       end for
12:       $U(D_{(i)}; s) = \min_a C(D_{(i)}; s, a)$
13:     end for
14:   until $\epsilon < \epsilon_0$
15: end for
16: Output $\phi^*$ such that $\phi^*(D; s) = \text{argmin}_a C(D; s, a)$
```

Similar to the original Value Iteration algorithm, as the stopping threshold $\epsilon_0 \to 0$, the exploration costs $U \to U^*$ and $C \to C^*$, thus $\phi \to \phi^*$. The proof for the convergence to optimal in limit and to near-optimal in finite time can be adapted straightforwardly from the convergence of the original VI algorithm (see [Puterman, 1994, Littman et al., 1995]) and will not be elaborated.

VIEC needs to enumerate all $|\mathcal{D}| = O(d^{|\mathcal{S}||\mathcal{A}|})$ demand matrices to compute the exploration cost at D_{init}, and thus is highly computationally expensive. One might wonder if time complexity can be significantly cut down by improving the computation method. Unfortunately, this is not likely for exact computation, because in the simplest case where all transitions are deterministic and all demands are either 0 or 1, the PFE problem becomes a combinatorial optimisation problem called (directed) rural postman problem[1], which is known to be NP-hard [Eiselt et al., 1995].

The main purpose of proposing VIEC is to show that the optimal exploration schemes not only exist, but also can be obtained from computation in finite time. When analysing the efficiency of demand fulfilment in relative simple MDPs, the optimal exploration scheme can sometime be quite obvious (which is the case in Section 3.4), and in such cases we do not need to actually run the VIEC algorithm in order to analyse exploration strategies.

On the other hand, in real-world applications where obtaining data is much more expensive than computation power, the potential benefit of applying a good exploration scheme can possibly exceed the price for finding it. We might want to use VIEC in a full-scale RL task, rather than treating it as a tool for efficiency analysis. In this case, since the true transition function P is not known to the learning algorithm, we have to use the estimated transition $\hat{P}$ in place of P in VIEC to obtain an estimation of the optimal exploration scheme ϕ^*.

The resulting exploration scheme $\hat{\phi}^*$ is the optimal scheme of the model MDP $\hat{M}$, which does not necessarily equal to the true optimal exploration scheme ϕ^* of M. However, we can refine $\hat{\phi}^*$ iteratively using the data obtained during the demand fulfilment process. With more data been collected, $\hat{M}$ becomes closer to M, and thus $\hat{\phi}^*$ also becomes closer to the true ϕ^* in probability. Although an improving-over-time $\hat{\phi}^*$ is indeed not as good as ϕ^*, it can still be used to avoid many problems in exploration such as the one mentioned

[1]In a rural postman problem, an extension to the well-known Chinese postman problem, the postman is required to traverse a subset of edges in a road network at the minimum cost.

in Example 3.1.

A full-scale model-based RL algorithm with iterative refinement of exploration scheme is given in Algorithm 2

Algorithm 2 Model-based RL with Iterative Refinement of Exploration Scheme

```
Input: Initial demand $D_{\text{init}}$
Output: Policy $\pi$
1: Initialise $\hat{P}, \hat{R}$ randomly or based on prior knowledge, $D_1 = D_{\text{init}}$
2: $\phi = \text{VIEC}(D_1, \hat{P})$
3: repeat
4:     Collect data by following $\phi$
5:     Update $\hat{P}, \hat{R}, D_t$ using collected data
6:     Update $\phi$ using $\text{VIEC}(D_t, \hat{P})$
7:     Update $\pi$ using Value Iteration$(\hat{P}, \hat{R})$
8: until $D_t = \mathbf{0}$
9: Output $\pi$
```

3.4 Analysis of sample efficiency

In this section, we analyse the exploration efficiency, or more precisely the efficiency of demand fulfilment, of some common exploration strategies by comparing their behaviours with the optimal exploration scheme.

As reviewed in Section 2.3.1, systematic strategies such as R-MAX, MBIE, and UCRL guide exploration by choosing actions with the maximum optimism-infused estimated values $\tilde{Q}(s, a)$. These $\tilde{Q}$ can be seen as predefined heuristics that are used in place of the explicitly planning-based exploration schemes. While these heuristics are properly designed to guarantee the completion of demand fulfilment in most cases, their designs do not necessarily lead to highly efficient execution of demand fulfilment. Specifically, they are prone to the following traps.

Distance traps. Although in systematic strategies such as the R-MAX family and the IE methods the uncertainty at the distant states are propagated to $\tilde{Q}(s, a)$ of the current state through the Bellman equations, the discounting mechanism in the Bellman equations makes the agent consider the uncertainty at the distant states less important than the uncertainty at the nearby states. Thus, existing strategies maximising $\tilde{Q}(s,a)$ tend to visit nearby uncertain states first, even if it is more efficient to visit distant uncertain states first.

Reward traps. $\tilde{Q}$ in all systematic strategies are combinations of the estimated utility that drives exploitation and the optimism that drives exploration. By choosing actions with maximum $\tilde{Q}(s,a)$, the agent (at least partly) consults the rewards in the demand fulfilment process. Thus, these strategies tend to visit uncertain states with higher rewards first, even if the rewards actually point to the area that ought to be visited later.

Both traps can appear in any MDP and may significantly increase the exploration cost of the demand fulfilment process. To give a clearer picture of how these traps affect the exploration efficiency, this section introduces a class of MDPs called tower MDPs, and analyse the exploration costs of the optimal exploration scheme, ε-greedy, R-MAX, and MBIE-EB in the tower MDPs.

3.4.1 Tower MDPs

A tower MDP of height h has $2h$ states, which are put into two groups $\{s_1, ..., s_h\}$ and $\{s'_1, ..., s'_h\}$. The first group is called the upward states and the second group is called the downward states. State s_1 is start state where the interaction begins. Figure 3.2 provides an illustration of a tower MDP.

The arrows in Figure 3.2 shows how transition works in tower MDPs. To simplify the analysis, all transitions in tower MDPs are deterministic. At each upward state s_k, there is an action a that transits the agent to the next upward state s_{k+1} if $k < h$ (solid arrows in Figure 3.2), and also an action a' that transits to the corresponding downward state s'_k (dashed arrows). Each downward state s'_k is an m-armed bandit with m actions $a_1, ..., a_m$. By pulling an arm at s'_k, the agent receives some rewards (specified later) and is transited to the downward state s'_{k-1} ($k > 1$) or the start state s_1 ($k = 1$). The actions of bandits are collectively drawn as the double arrows in Figure 3.2.

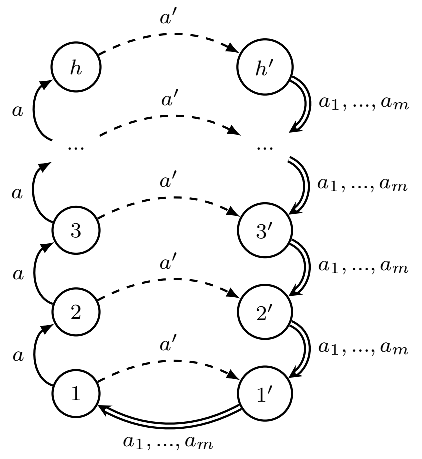

Figure 3.2: A tower MDP of height h.

The optimal policy involves how many levels the agent should climb from s_1, and which arms it should play at each bandit. For example, suppose the reward of the bandit is 1 with probability 0.1 and 0 with probability 0.9 for a_1 at s'_h , and is constantly 0 for all other arms and states. Also suppose that the discount factor γ is sufficiently close to 1. Then the optimal policy is to climb to the top of the tower from s_1, play a_1 at s'_h, then play arbitrary arms at s'_{h-1},...,s'_1 to go back to s_1. For another example, suppose the reward is 1 with probability 0.1 for a_1 at s'_1 and 0 otherwise, then the optimal policy is to take a' at s_1, then take a_1 at s'_1.

To decide which policy is optimal, the agent has to collect information about the rewards of the bandits. Since the rewards of the bandits are stochastic, each bandit arm should be played for several times to obtain the reward distribution. Thus, we can assign an initial demand D_{init} where the entries equal to some $d > 0$ for all bandit arms (s'_i, a_j). For all non-bandit actions (s_i, a) and (s_i, a'), the initial demand is 0 since there is no uncertainty regarding their dynamics and rewards.

3.4.2 Optimal exploration scheme

From the structure of tower MDPs, it is easy to see that the most efficient way to fulfil the demand at the bandits is to climb up to the top s'_h, play each bandit once and back to s_1, repeating for $m \times d$ times. Therefore, the following exploration scheme is optimal in tower MDP of height h:

$$\phi^*(D; s) = \begin{cases} a & s \in \{s_1, ..., s_{h-1}\} \\ a' & s = s_h \\ a_i & s \in \{s'_1, ..., s'_h\}, i = \text{argmin}_j\{D[s, a_j] > 0\}. \end{cases}$$

The optimal exploration scheme is not unique in tower MDPs because the order of choosing arms (i in the above definition) can in fact be arbitrary. However, their exploration costs are the same, which is given as follows.

Lemma 3.3. *The exploration cost of ϕ^* is $2hmd$ in tower MDPs.*

Proof. Executing the exploration scheme ϕ^* results in a trajectory that consists of base sequence $[s_1a, s_2a, ...s_{h-1}a, s_ha', s'_ha_i, s'_{h-1}a_i, ...s'_1a_i]$, repeating d times, with i iterating from 1 to m. The length of this base sequence is $2h$, and thus the total exploration cost is $2hmd$. □

3.4.3 ε-greedy

It is a well-known fact that random strategies such as ε-greedy are not sample efficient due to their randomness in exploration. With probability $(1 - \varepsilon)$, ε-greedy selects the action with maximum $\hat{Q}(s, a)$, and with probability ε, it selects a random action from a uniform distribution. Also, when all actions have the same $\hat{Q}(s, a)$, it selects actions uniformly randomly. In tower MDPs, such randomness leads to an exponential exploration cost, given in the following lemma.

Lemma 3.4. *The worst-case exploration cost of ε-greedy with parameter $0 < \varepsilon < 1$ is $\Omega(hmd\,(\frac{2}{\varepsilon})^h)$ in tower MDPs.*

Proof. At the beginning of exploration, $\hat{Q}$ is 0 for all state-action pairs. Thus, it has 0.5 probability to select a and 0.5 probability to select a' at any $s_1, ..., s_h$. As a result, the probability of it first taking a' and arriving at downward state s'_j is 0.5^j for $j < h$ and is 0.5^{h-1} for $j = h$.

After arriving at some s'_j, the agent is aware of the rewards from the multi-armed bandits. From this time point, ε-greedy starts taking actions with maximum $\hat{Q}$ with high probability, and thus falls into the reward trap.

To obtain the worst-case lower bound, we only need to construct one example where the bound holds. To this end, let the reward of the bandits be 0.01 with probability 1 for all arms at s'_1, be 1 with probability 1 for s'_h, and be 0 otherwise. With a discount factor sufficiently close to 1, this makes ε-greedy try to take the path $[s_1 s_2 ... s_{j-1} s_j s'_j s'_{j-1} ... s'_1 s_1]$ repeatedly, and only with probability 0.5ε at each upward s_i it wanders into some other state that is not in the path.

Notably, there is probability $p = 0.5$ that the first s'_j the agent visits is s'_1. In this case, the probability that ε-greedy later wanders into s'_h through $h-1$ successive random exploration starting from s_1 is only $p' = (0.5\varepsilon)^{h-1}$. The average number of trials it needs to reach s'_h from s_1 is $\frac{1}{p'}$. Thereafter, the number of steps ε-greedy fulfilling the demand at s'_h is at least $n = 2hmd$. Thus, the expected exploration cost of ε-greedy is at least $p \cdot \frac{1}{p'} \cdot n = 0.5 \cdot \frac{1}{(0.5\varepsilon)^{h-1}} \cdot 2hmd = hmd\,(\frac{2}{\varepsilon})^{h-1}$.

The above computation ignores all the addition steps wasted in the paths taking a' at states s_i with $i < h$, as well as by playing the wrong arms that have already been tried d times due to the random exploration. Thus, the final expected exploration cost of ε-greedy is $\Omega(hmd\,(\frac{2}{\varepsilon})^h)$. □

3.4.4 R-MAX

Re-describing in the terms used in this chapter, R-MAX [Brafman and Tennenholtz, 2002, Strehl et al., 2009] works as follows. At time t, every state-action pair (s, a) with demand $D_t[s, a] > 0$ is labelled "unknown" and its $\tilde{Q}(s, a)$ is set to $V_{\max} := \frac{R_{\max}}{1-\gamma}$, where $R_{\max}$ is the maximum reward of the MDP. State-action pairs with $D_t[s, a] = 0$ are labelled "known" and their $\tilde{Q}(s, a)$ is computed from the Bellman optimality equation (see Section 2.1).

R-MAX always chooses actions with maximum $\tilde{Q}(s,a)$.

Lemma 3.5. *The worst-case exploration cost of R-MAX with its exploration parameter $m_{R\text{-}MAX} = d$ is $\Omega(h^2md)$ in tower MDPs.*

Proof. Let the reward of the bandit arm (s'_h, a_i) be 1 with probability 0.1 and 0 with probability 0.9 for all a_i. Let the reward of all other bandits at s'_j with $j < h$ be 0. At the beginning of exploration, all actions in downward states are "unknown" and thus their $\tilde{Q}(s'_i, a_j) = V_{\max} = \frac{1}{1-\gamma}$. All $\tilde{Q}$ for a and a' are computed from the Bellman optimality equation.

At s_h, the only action available is a' which leads to the "unknown" s'_h, thus $\tilde{Q}(s_h, a') = \gamma V_{\max} = \frac{\gamma}{1-\gamma}$. At state s_{h-1}, choosing action a which leads to s_h has action value $\tilde{Q}(s_{h-1}, a) = \gamma\tilde{Q}(s_h, a') = \frac{\gamma^2}{1-\gamma}$. Meanwhile, choosing action a' which leads to s'_{h-1} has action value $\tilde{Q}(s_{h-1}, a') = \gamma\tilde{Q}(s'_{h-1}, a_{(\cdot)}) = \gamma V_{\max} = \frac{\gamma}{1-\gamma}$. Since $0 < \gamma < 1$, we have $\tilde{Q}(s_{h-1}, a) < \tilde{Q}(s_{h-1}, a')$, and thus R-MAX chooses a' at s_{h-1}. The above analysis can be repeated for $s_{h-2}, s_{h-3}, ..., s_1$ to show that at this stage $\tilde{Q}(s_j, a) = \frac{\gamma^2}{1-\gamma}$ and $\tilde{Q}(s_j, a') = \frac{\gamma}{1-\gamma}$ for all $j < h$, and thus R-MAX selects a' at any $s_1, ..., s_h$. Since the agent starts from s_1, this results in R-MAX repeating $[s_1 a' s'_1 a_{(\cdot)}]$ until all $a_1, ..., a_m$ at s'_1 are tried d times.

After the demand at s'_1 are satisfied, it becomes "known" and thus $\tilde{Q}(s'_1, a_{(\cdot)})$ changes from $V_{\max}$ to $\gamma\max\{\tilde{Q}(s_1, a), \tilde{Q}(s_1, a')\}$. Since $\tilde{Q}(s_1, a') = \gamma\tilde{Q}(s'_1, a_{(\cdot)})$ and $\tilde{Q}(s_1, a) = \frac{\gamma^2}{1-\gamma}$, it must hold that $\tilde{Q}(s'_1, a_{(\cdot)}) = \gamma\tilde{Q}(s_1, a) = \frac{\gamma^3}{1-\gamma}$ and $\tilde{Q}(s_1, a') = \frac{\gamma^4}{1-\gamma} < \tilde{Q}(s_1, a)$. Thus, R-MAX chooses a at s_1 and travels to s_2. Then, since as previously proved $\tilde{Q}(s_2, a) < \tilde{Q}(s_2, a')$, R-MAX chooses a' at s_2, resulting in repeating $[s_1 a s_2 a' s'_2 a_{(\cdot)} s'_1 a_{(\cdot)}]$ until all $a_1, ..., a_m$ at s'_2 are tried d times and become "known".

The above analysis can be repeated for $s_3, s_4, ..., s_h$ to show that R-MAX takes the path $[s_1 s_2 ... s_j s'_j ... s'_1]$, each repeated md times, with j iterating from 1 to h. The total number of steps of such a process is $2md + 4md + ... + (2h)md = h(h+1)md$. Since at the last loop the agent fulfils the final demand at s'_h and does not need to return to s_1, the exploration cost of R-MAX is $h(h+1)md - (h-1)$. Thus, the worst-case exploration cost is $\Omega(h^2md)$ in tower MDPs. □

3.4.5 Interval estimation

Interval estimation methods use the upper limit of confidence intervals (CIs) to drive exploration. The lower and upper limits of the CIs in these strategies usually take the form of $X(s,a) \pm \frac{\beta}{\sqrt{N_{s,a}}}$, where X is the quantity being estimated, which can be either P, R, V, or Q, while β is a parameter decided by the confidence level, and $N_{s,a}$ is the amount of data collected at (s,a). For example, MBIE-EB maintains confidence intervals for R, where the upper limit is stored as $\tilde{R}(s,a) \overset{\text{def}}{=} \hat{R}(s,a) + \frac{\beta}{\sqrt{N_{s,a}}}$. Since $N_{s,a} = 0$ results in division by 0, we assume that they are regarded as 1 in division if equals to 0. MBIE-EB chooses actions with maximum $\tilde{Q}(s,a)$ computed from $\tilde{Q}(s,a) = \tilde{R}(s,a) + \gamma \sum_{s'} \hat{P}(s'|s,a) \max_{a'} \tilde{Q}(s',a')$, which is an optimistic estimate due to using $\tilde{R}$ in place of $\hat{R}$.

Lemma 3.6. *The worst-case exploration cost of MBIE-EB is $\Omega(h^2m + hmd)$ in tower MDPs.*

Proof. To simplify the analysis, we first consider the tower MDP with $m = 1$, i.e. all downward states are 1-armed bandits. We also assume for now that the parameter β of MBIE-EB is set to a sufficiently large value s.t. in $\tilde{R}(s,a) \overset{\text{def}}{=} \hat{R}(s,a) + \frac{\beta}{\sqrt{N_{s,a}}}$, the estimated reward $\hat{R}(s,a)$ is negligible[1] and thus $\tilde{R}(s,a) = \frac{\beta}{\sqrt{N_{s,a}}}$.

Due to the structure of tower MDP, the max operator in the Bellman optimality equation is essentially choosing between paths $[s_1 ... s_j s'_j ... s'_1 s1]$ of different j. Let $\tilde{Q}_j$ denotes the optimistic utility $\tilde{Q}$ starting from s_1 and following the j-th path, $\tilde{R}_j$ denotes $\tilde{R}$ provided by the bandit at s'_j, and N_j denotes $N_{s'_j,a_1}$ which is the sample size at the bandit of s'_j. By solving the Bellman equation, we have $\tilde{Q}_j = \frac{\gamma^j}{1-\gamma^{2j}} \sum_{i=1}^{j} \gamma^{j-i} \tilde{R}_i$.

At the beginning of learning, for all i we have $N_i = 0$ which are regarded as 1, and thus $\tilde{R}_i = \beta$. Then, $\tilde{Q}_j = \frac{\beta}{1-\gamma}(1 - \frac{1}{1+\gamma^j})$. Clearly, we have $\tilde{Q}_1 > \tilde{Q}_2 > ... > \tilde{Q}_h$, and therefore MBIE-EB chooses the path $[s_1 s'_1 s_1]$ as the first several steps. Then, although the first visit to s'_1 changes N_1 to 1, $\tilde{R}_1$ is still β, and thus the path $[s_1 s'_1 s_1]$ is repeated once more, changing N_1 to 2, which reduces $\tilde{R}_1$ to $\frac{\beta}{\sqrt{2}}$.

[1]This can be achieved by choosing a sufficiently small δ in $\beta = \frac{1}{1-\gamma}\sqrt{\frac{\ln(2|\mathcal{S}||\mathcal{A}|d/\delta)}{2}}$, see [Strehl and Littman, 2008] for more details.

The expression $\tilde{Q}_j = \frac{\gamma^j}{1-\gamma^{2j}} \sum_{i=1}^{j} \gamma^{j-i} \tilde{R}_i$ can be seen as a weighted sum of $\tilde{R}_i$, where $\tilde{Q}_j$ weights $\tilde{R}_i$ by $\frac{\gamma^{2j-i}}{1-\gamma^{2j}} = \gamma^{-i}(\frac{1}{1-\gamma^{2j}} - 1)$. Clearly, $\tilde{R}_i$ is weighted less with a larger j. Thus, when $\tilde{R}_1$ is reduced, $\tilde{Q}_2$ is reduced less than $\tilde{Q}_1$. As N_1 grows, $\tilde{Q}_2$ eventually exceeds $\tilde{Q}_1$, at which point MBIE-EB stops taking the path $[s_1 s'_1 s_1]$ and starts taking $[s_1 s_2 s'_2 s'_1 s_1]$.

The same analysis can be repeated to show that after N_2 is sufficiently large, MBIE-EB will change to $[s_1 s_2 s_3 s'_3 s'_2 s'_1 s_1]$, then $[s_1 ... s_4 s'_4 ... s'_1 s_1]$, and so on, until for the first time s_h is reached. Thereafter the path $[s_1 ... s_h s'_h ... s'_1 s_1]$ will be repeated until all bandits are tried d times. For the sake of brevity we do not elaborate how many times each $[s_1 ... s_j s'_j ... s'_1 s_1]$, is repeated; it suffices to say that each path is taken at least twice, since as explained above the first visit does not change $\tilde{R}_j$. Thus, the exploration cost of MBIE-EB is at least $2(2 + 4 + ... + 2(h-1)) + 2hd = 2h(h-1) + 2hd$.

Now consider the case of $m \leq 2$ where there is more than one arm in each bandit. In this case, $\tilde{Q}_j = \frac{\gamma^j}{1-\gamma^{2j}} \sum_{i=1}^{j} \gamma^{j-i} \max_{k\in\{1,...,m\}} \tilde{R}_{i,k}$, where $\tilde{R}_{i,k} = \frac{\beta}{N_{s'_i, a_k}}$. Thus, $\tilde{Q}_j$ does not change until all arms at s'_i are tried the same number of times. Therefore, the exploration cost of MBIE-EB at $m \leq 2$ is simply m times the cost at $m = 1$, that is, at least $2h(h-1)m + 2hmd = \Omega(h^2 m + hmd)$. □

Note that in the above analysis for MBIE-EB, β is assumed to be sufficiently large such that the rewards R are negligible. However, in reality this makes MBIE-EB likely to over-explore, i.e. collect more data than needed at some state-action pairs, which is not desirable. On the other hand, with a smaller β, the rewards R takes more effect in $\tilde{R}$, which makes MBIE-EB weaker to the reward trap as ε-greedy does. Specifically, by using the same reward setting as the one used in the analysis of ε-greedy, MBIE-EB can be lured to taking paths $[s_1 ... s_j s'_j ... s'_1 s_1]$ $(j < h)$ more often than in the analysis above, further increasing its exploration cost.

Strategy	Exploration cost	Weakness
Optimal scheme	$\Theta(\lvert\mathcal{S}\rvert\lvert\mathcal{A}\rvert d)$	-
ε-greedy	$\Omega(\lvert\mathcal{S}\rvert\lvert\mathcal{A}\rvert d \cdot 2^{\lvert\mathcal{S}\rvert})$	Distance trap, reward trap
R-MAX	$\Omega(\lvert\mathcal{S}\rvert^2\lvert\mathcal{A}\rvert d)$	Distance trap
MBIE-EB	$\Omega(\lvert\mathcal{S}\rvert^2\lvert\mathcal{A}\rvert + \lvert\mathcal{S}\rvert\lvert\mathcal{A}\rvert d)$	Distance trap, reward trap

Table 3.1: Summary of results in tower MDPs.

3.4.6 Discussion

The exploration costs analysed in the previous subsections can be rewritten in terms of $|\mathcal{S}|$, $|\mathcal{A}|$ and d by replacing $|\mathcal{S}| = 2h$ and $|\mathcal{A}| = m$. The results are presented in Table 3.1.

Clearly, ε-greedy is the worst exploration strategy in tower MDPs because of the exponential term in its exploration cost. Both R-MAX and MBIE-EB have polynomial exploration costs, and thus are far better than ε-greedy, but they are still worse than the optimal exploration scheme. MBIE-EB appears to be better than R-MAX, but the difference in reality is not significant because usually $|\mathcal{S}| \gg d$ and both strategies are quadratic to $|\mathcal{S}|$. Thus, there is still a very large room for improvement for R-MAX and MBIE-EB.

The exploration costs of R-MAX and MBIE-EB share some similarity with their sample complexity. R-MAX and MBIE-EB have sample complexity upper bound quadratic to the size of state space $|\mathcal{S}|$ and linear to the size of action space $|\mathcal{A}|$, which is the same as their exploration costs. Interestingly, there is a variant of R-MAX called MoR-MAX which has a sample complexity upper bound linear to both $|\mathcal{S}|$ and $|\mathcal{A}|$, which is much better than R-MAX with respect to $|\mathcal{S}|$, while its exploration behaviour in tower MDPs is exactly the same as R-MAX[1], and thus has the same worst-case exploration cost $\Omega(|\mathcal{S}|^2|\mathcal{A}|d)$. The difference between the sample complexity and the exploration cost with respect to $|\mathcal{S}|$ is possibly because in the sample complexity analysis early exploitation before completing exploration is allowed as long as such exploitation is sufficiently good,

[1]MoR-MAX achieves a better sample complexity upper bound by using a much smaller threshold than R-MAX when deciding whether an "unknown" state-action pair becomes "known", and forces the agent to discard all the collected data and restart exploration when it finds itself converging prematurely. Thus, given the same threshold (i.e. data demand), the behaviour of MoR-MAX and R-MAX is the same.

while in our exploration cost analysis early exploitation does not bring in any benefit and only increases the cost. Since in the learning-then-serving workflow exploitation during learning is not needed, this difference suggests that sample complexity bound cannot be used to evaluate sample efficiency in learning-then-serving workflow.

The "Weakness" column lists the weaknesses found in the exploration cost analysis. Both ε-greedy and IE methods are prone to the reward trap because their exploration behaviour can be altered by the rewards of the MDP. As long as the reward setting does not coincide with the correct order of exploration, the rewards are just distraction to the demand fulfilment process and should be ignored.

Some readers may argue that reward function can be properly designed such that it both expresses the optimality of desirable policies and points out the areas that should be explored first. However, in reality finding a suitable reward function is already a very difficult task for RL users. It usually involves manual tweaking the function to see whether the learnt policy satisfies the requirement of the application or not, which is expensive in terms of both manpower and data (see the researches on inverse reinforcement learning, e.g. [Abbeel and Ng, 2004]). It is thus unwise to further require the RL users to design reward functions that both fulfil their own purpose and guide exploration efficiently, especially in limited budget setting.

As for the distance trap, both R-MAX and MBIE-EB are weak to it because they tend to explore nearby uncertain state-action pairs first (e.g. s'_1 at the beginning of exploration). The underlying reason behind such a weakness is the use of the discount $\gamma < 1$ in propagating the uncertainty. Specifically, both R-MAX and IE algorithms use optimistic $\tilde{Q}$ to express the uncertainty, and use the discounted versions of the Bellman equation to propagate it to other state-action pairs. A large uncertainty in a farther state is discounted more and thus becomes much smaller when passed to the current state, making them less likely to be explored first. This is an unavoidable problem when expressing the data demand and the estimated utility in one single $\tilde{Q}$, because the estimated utility must be discounted in MDPs with $\gamma < 1$. The solution of our planning for exploration problem does not suffer from distance trap because the exploration costs and the estimated utility are completely separated, and the exploration costs are not discounted when passing through the augmented undiscounted MDP of the PFE problem.

Tower MDPs in the above analysis use deterministic transitions for the sake of simplicity. In non-deterministic cases, the distance trap becomes even more harmful for the existing strategies weak to it. For example, suppose that taking a' from s_1 still transits the agent to s_1' with probability 1, but taking a from s_1 transits to s_2 with probability 0.5 and remains at s_1 with probability 0.5. This further increases the gap in $\tilde{Q}$ between taking a and a', making the IE methods choose a' at s_1 more often than previously analysed.

We choose tower MDPs in the analysis because it is easier to demonstrate the effects of distance and reward traps in these MDPs. Meanwhile, the two traps are not limited to tower MDPs, but can occur in any types of MDPs and make existing strategies inefficient. In some easier cases the difference in exploration costs between the optimal scheme and existing strategies might not be as large as $O(|\mathcal{S}||\mathcal{A}|d)$ vs $O(|\mathcal{S}|^2|\mathcal{A}|d)$, but even a difference in constant factor can still be significant in practice, especially when environment interaction is expensive. Since the exploration schemes computed from explicit planning for exploration are not affected by distance and reward traps, they can potentially be highly useful for improving sample efficiency even if they are only estimates and/or approximations to the true optimal schemes.

3.5 Chapter summary

In this chapter, we have formulated the planning for exploration (PFE) problem, proposed the exploration cost of demand fulfilment process as a metric for evaluating the sample efficiency of exploration, showed that the optimal exploration scheme can be obtained by solving an augmented undiscounted MDP, and proposed the Value Iteration for Exploration Cost (VIEC) algorithm to solve it. These answer the research question Q1.1 ("How to better analyse the sample efficiency of exploration strategies with a sample-size-based performance metric from a planning-based point of view?") proposed in Section 1.3.1 and lead to contribution 1 in Section 1.4.[1]

We have then analysed the exploration behaviours and the corresponding exploration costs of the optimal exploration scheme, ε-greedy, R-MAX, and MBIE-EB, pointed out the two weaknesses of existing strategies (i.e. distance trap and reward trap), and showed that the optimal exploration scheme computed from solving the PFE problem is immune to these traps and thus has much less exploration cost than the other strategies. These answer the research questions Q1.2 ("According to this new analysis, what are the weaknesses of existing exploration strategies?") and Q1.3 ("Can explicit planning for exploration significantly improve the sample efficiency?") in Section 1.3.1, and lead to contribution 2 in Section 1.4.[2]

[1]Contribution 1: "We propose the planning for exploration (PFE) problem and formulate it as an augmented MDP. We show that given the exploration demand and true transitions, the optimal exploration scheme can be found by solving the augmented MDP using a modified version of value iteration algorithm. We show that sample efficiency of exploration strategies can be analysed by comparing their behaviours with the optimal exploration scheme."

[2]Contribution 2: "Exploration behaviours in a class of tower MDPs are analysed to illustrate the benefit of explicit planning for exploration, and to expose two weaknesses of existing methods, namely distance traps and reward traps."

CHAPTER 4

TRANSFORMATION BIAS OF MODEL-BASED RL AND ITS CORRECTION

4.1 Introduction

As reviewed in Section 2.4.2, in model-based RL, there is a particular factor that can possibly reduce significantly the overall accuracy of state/action value estimates, which we call the *transformation bias*. To recapitulate, the Bellman equations form a system of linear equations, where state/action values are the unknowns while the transition probabilities (multiplied by some constants) are the coefficients. Although the equations themselves are linear, the solution of such a system is a nonlinear mapping from the transition probabilities to the state/action values. For example, the Bellman equation for state value in matrix form is $\mathbf{V} = (\mathbf{P} \odot \mathbf{R})\mathbf{1} + \gamma\mathbf{P}\mathbf{V}$, which is a linear equation of $\mathbf{V}$, while its solution $\mathbf{V} = (\mathbf{I} - \gamma\mathbf{P})^{-1}(\mathbf{P} \odot \mathbf{R})\mathbf{1}$ is a nonlinear mapping from $\mathbf{P}$ to $\mathbf{V}$. Due to the nonlinearity, the state/action value estimates are biased, regardless of whether the transition probability estimates are biased or not.[1]

The transformation bias has been studied by [Mannor et al., 2004, 2007], in which second order approximations to the bias and variance of the estimated values are proposed. Their analysis suggests that the magnitude of the variance is much larger than the bias

[1]By the Bellman equations 2.1 and 2.2 the action values Q are just linear transformation of state values V, and thus they have the same mathematical properties with respect to their transformation bias. For brevity this chapter mainly discusses V, but the results are easily applicable to Q.

in general, and therefore correcting the bias is less useful than reducing the variance.

However, our empirical investigation on the transformation bias, which will be presented later in this chapter, shows that the transformation bias can be large enough to make the value estimates completely useless, especially when the sample size per state-action pair is small. As will be explained in Section 4.2, when the bias is sufficiently large to change the relation between two state/action values, having a low variance can even increase the probability of making wrong decisions. Without bias correction, the only way to reduce the negative effect of the transformation bias is to collect a larger sample, which is not desirable when obtaining data is expensive. Thus, correction of the transformation bias is of great importance in a limited budget setting.

In this chapter, we present our detailed study on the transformation bias which reveals its several important characteristics. Firstly, both positive and negative transformation bias can happen in model-based RL, making some value estimates more likely to be overestimated while the others underestimated. Secondly, values that are related to Markov chains with low connectivity tend to have large bias, while the ones related to almost fully connected Markov chains tend to have small bias. Thirdly, under the same sample size per state-action, the magnitude of the bias increases with the number of states, meaning that guaranteeing the model accuracy is not sufficient for guaranteeing the accuracy of value estimates.

To reduce the bias and thus improve the learning efficiency, we propose Bootstrap-based Transformation Bias Correction (BTBC) for model-based RL. BTBC utilises the well-known statistical technique of bootstrap to estimate the unknown transformation bias. At the cost of an increase in the computation time, BTBC can effectively reduce the bias of value estimators without requiring more data, which is helpful when obtaining new data is much more expensive than computational resources.

The remainder of this chapter is organised as follows. In Section 4.2, we explain the reason why bias-variance trade-off is problematic and bias correction is important in model-based RL. In Section 4.3, we present our detailed study on the transformation bias, revealing several of its important properties that have not been mentioned in literature. In Section 4.4, we propose the BTBC method for correcting the transformation bias. In Section 4.5, we present our experiment results to show the empirical efficiency of BTBC. In Section 4.6, we briefly discuss the implicit transformation bias in model-free RL. Finally, in Section 4.7, we give a summary to this chapter.

4.2 The problematic bias-variance trade-off and the importance of bias correction in RL

In supervised learning, there is a classic idea of bias-variance trade-off which claims that low-bias predictions can only be achieved at the cost of high variance, and that it is usually more beneficial to give up unbiasedness for low variance in order to achieve low overall estimation error (e.g. [Geman et al., 1992, Hastie et al., 2009]).

However, simply trading bias for variance can be problematic in reinforcement learning. In RL, having estimates with low errors is not the final objective of learning. Rather, the objective is to make informed decisions, i.e. to choose actions/policies that lead to high learning/serving process cumulative rewards. In this context, perhaps somewhat counter-intuitively, having higher variance in value estimates does not necessarily leads to worse performance when the estimates are biased.

Figure 4.1 gives an illustration of problematic bias-variance trade-off. Suppose that we are comparing two state values $V = 3$ and $V' = 4$, which are represented in Figure 4.1 by the thin and thick vertical lines at $x = 3$ and $x = 4$, respectively. Obviously, the correct answer of the comparison is $V < V'$. For simplicity, suppose that the estimated state value $\hat{V}'$ happens to be $\hat{V}' = V' = 4$. Let us consider three estimators $\hat{V}_1$, $\hat{V}_2$, and $\hat{V}_3$ of V, whose estimates follow normal distributions $\hat{V}_1 \sim \mathcal{N}(3, 1)$, $\hat{V}_2 \sim \mathcal{N}(5, 1)$, and $\hat{V}_3 \sim \mathcal{N}(5, 0.5^2)$, respectively. Their probability density functions are presented by the three curves in Figure 4.1.

As can be seen from the figure, $\hat{V}_1$ has no bias but high variance, $\hat{V}_2$ has high bias and

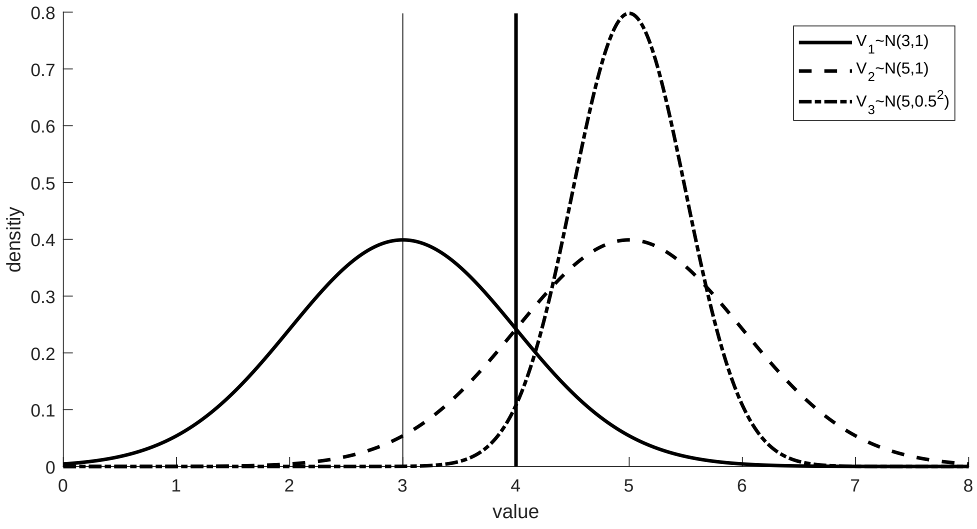

Figure 4.1: Problematic bias-variance trade-off in RL.

high variance, and $\hat{V}_3$ has high bias but low variance. From the pure value estimation point of view, $\hat{V}_2$ seems to be the worst one due to being higher in both bias and variance, and if we blindly follow the trade-off dogma, then the unbiased high-variance $\hat{V}_1$ should be worse than biased low-variance $\hat{V}_3$.

These relations change dramatically when we consider the value comparison problem $\hat{V}$ vs $\hat{V}'$. The regions of the three distributions at the left side of the thick line $x = 4$ indicate the probability density of $\hat{V} < 4 = \hat{V}'$. The areas of these regions are thus the probabilities of yielding the correct answer $\hat{V} < \hat{V}'$ by the three estimators. Interestingly, $\hat{V}_1$ has the most areas in $x < 4$, then follows by $\hat{V}_2$, while $\hat{V}_3$ has the least. The estimator deemed the best in pure value estimation scenario, $\hat{V}_3$, becomes the worst one in terms of probability of successful value comparison. Even estimator $\hat{V}_2$, which has the same bias as $\hat{V}_3$ but a higher variance, is more likely to success in value comparison. In other words, reducing the variance actually increases the chance of failure in this case.

Indeed, it is not difficult to construct opposite cases where reducing the variance does help. The key factor deciding whether variance reduction helps or not is in fact the magnitude of the bias. Intuitively, if the biases of estimators change the relation between two values (i.e. $V > V'$ but $\mathbb{E}[\hat{V}] < \mathbb{E}[\hat{V}']$, or the opposite), then variance reduction makes the estimates more concentrated to the wrong values and thus increases the chance of failure. If the relation is not changed by the bias, then variance reduction helps.

The problem is, both kinds of cases can happen at the same time in one single MDP, and it is difficult to discern which kind of case it is for a specific pair of value estimates without knowing their true values. Therefore, to ensure the usefulness of variance reduction, or more generally the accuracy improvement of estimated values, the most reliable way is thus to always consider the potential negative effect of the biases first, and if necessary reduce such effect by either directly conducting bias reduction/correction, or use more deliberate value comparison methods that are strong to the biases of value estimates (which will be elaborated in Section 5). If none of these are conducted, then the only way to reduce the bias is to collect a larger sample, resulting in low sample efficiency.

4.3 A detailed study of the transformation bias in model-based RL

As discussed in Section 2.4, both the transformation bias and the maximisation bias of RL comes from the Bellman equations. For simplicity, we mainly focus on Bellman equation for $\hat{V}^{\pi}$ in this chapter in which the max operator is not directly involved. However, our results can easily be generalised to other forms as well.

Since bias correction often requires additional computation and makes algorithms more complicated, it is important to know in what kind of MDPs the bias is remarkable and harmful. Although Mannor et al. [2007] provides a second order approximation to the transformation bias, the resulting mathematical expressions are rather difficult to interpret. Therefore, in the following subsections we present our systematic study of the transformation bias based on both analysis and empirical results.

4.3.1 The direction of the transformation bias

Unlike the maximisation bias which always results in $\mathbb{E}[\hat{V}] > V$, the the transformation bias can occur in both directions $\mathbb{E}[\hat{V}] > V$ and $\mathbb{E}[\hat{V}] < V$. To show this, we give two simple examples as follows.

Example 4.1. Consider an MDP with two states s_1 and s_2, one action a, transition matrix $\left[\begin{smallmatrix} 1-p & p \\ 1 & 0 \end{smallmatrix}\right]$, reward $\left[\begin{smallmatrix} 0 & 0 \\ 1 & 0 \end{smallmatrix}\right]$, and discount factor $\gamma \in (0,1)$. Let V_1 and V_2 be the state values by taking a at s_1 and s_2, respectively. By the Bellman equation, we have

$$\begin{aligned} V_1 &= (1-p)\gamma V_1 + p\gamma V_2 \\ &= (1-p)\gamma V_1 + p\gamma(1+\gamma V_1) \\ &= \frac{p\gamma}{1-(1-p)\gamma - p\gamma^2}, \end{aligned}$$

which is a concave function of p for $p \in [0,1]$. Seeing the estimated state value $\hat{V}_1 = V_1(\hat{p})$ as a function of the estimated transition probability $\hat{p}$, by Jensen's inequality we have $\mathbb{E}[V_1(\hat{p})] \leq V_1(\mathbb{E}[\hat{p}])$ and therefore $\mathbb{E}[\hat{V}_1] \leq V_1$.

Example 4.2. Consider an additional action a' in the MDP of Example 4.1 which leads to the same transition matrix as a, but has a different reward matrix $\left[\begin{smallmatrix} 1 & 0 \\ 0 & 0 \end{smallmatrix}\right]$. Let V_1' and V_2' denote the state values by taking a' at s_1 and s_2, respectively. By the Bellman equation, we have

$$\begin{aligned} V_1' &= (1-p)(1+\gamma V_1') + p\gamma V_2' \\ &= (1-p)(1+\gamma V_1') + p\gamma(\gamma V_1') \\ &= \frac{1-p}{1-(1-p)\gamma - p\gamma^2}, \end{aligned}$$

which is a convex function of p at $p \in [0,1]$, and thus by Jensen's inequality $\mathbb{E}[\hat{V}_1'] \geq V_1'$.

As can be seen from the examples, both $\mathbb{E}[\hat{V}] \leq V$ and $\mathbb{E}[\hat{V}] \geq V$ can happen. The inequalities are strict if and only if $p \neq 0$ and $p \neq 1$, which means that in these examples the estimated values are biased as long as the transitions are not deterministic.

More importantly, these two examples together show that positive and negative bias can occur simultaneously for different value estimates in the same MDP. When $p\gamma < 1-p$ and thus $p < \frac{1}{1+\gamma}$, we have $V_1 < V_1'$. Since $\mathbb{E}[\hat{V}_1] \leq V_1$ and $V_1' \leq \mathbb{E}[\hat{V}_1']$, we have

$\mathbb{E}[\hat{V}_1] < \mathbb{E}[\hat{V}_1']$, which means that the transformation bias is harmless in this case. On the other hand, when $p\gamma > 1 - p$ and thus $p > \frac{1}{1+\gamma}$, we have $V_1 > V_1'$. In this case, $\mathbb{E}[\hat{V}_1] \leq V_1$ and $V_1' \leq \mathbb{E}[\hat{V}_1']$ make $\hat{V}_1$ and $\hat{V}_2$ more likely to get mixed or even lead to the wrong conclusion $\hat{V}_1 < \hat{V}_2$, and therefore the transformation bias clearly becomes harmful.

This is an important characteristic of the transformation bias that differs from other biases of RL such as the maximisation bias. The maximisation bias is known to be always $\mathbb{E}[\hat{V}] > V$, which means that when comparing two values V and V', their estimated values $\hat{V}$ and $\hat{V}'$ are shifted to the same direction. If the biases of $\hat{V}$ and $\hat{V}'$ have similar scale, then $V_1 < V_2$ implies $\mathbb{E}[\hat{V}_1] < \mathbb{E}[\hat{V}_1']$ while $V_1 > V_2$ implies $\mathbb{E}[\hat{V}_1] > \mathbb{E}[\hat{V}_1']$, and therefore the maximisation bias is harmless. In other words, it is harmful only when there is a significant difference between the scales of the biases among different estimated values. In contrast, since transformation bias can occur in both directions, even if the biases of two value estimates have the same scale, it can still be harmful for learning.

4.3.2 The effect of transition dynamics

In [Grünewälder and Obermayer, 2011], it is shown that cyclic Markov chains (or cycles for short) in MDPs lead to non-zero transformation bias. However, it is not clear from their results how multiple cycles interact. The transition dynamics in Examples 4.1 and 4.2 are both cycles, but they lead to different directions in the transformation bias. For convenience, we say a cycle is positive if it leads to a positive transformation bias, and negative if it leads to a negative bias. What if a state/action value consists of both types of cycles?

Our extensive experiments showed a general finding that an MDP with better connectivity often has less transformation bias, and vice versa. Our explanation to this finding is, in a fully connected MDP with uniformly random non-zero transition probabilities and random rewards, the positive and negative cycles are equally likely to happen, and thus their corresponding positive and negative transformation biases cancel each other, leading to a small final bias on average. With more connections removed from the MDP, the positive and negative biases become more likely to be unbalanced, resulting in a larger final bias.

To illustrate this, consider a class of $(n-1)$ MDPs with n states and 1 action, denoted $\{M_k^n\}$, in which M_k^n with connectivity parameter $c = k$ $(1 \leq k < n)$ has transition probability function:

$$P_{i,j} = \begin{cases} 1/\min\{c+1, n-i+1\} & i \neq n, i \leq j \leq \min\{i+c, n\} \\ 0.5 & i = n, j = 1 \text{ or } n \\ 0 & \text{otherwise.} \end{cases}$$

Thus the agent transits from s_i to s_i, s_{i+1}, ..., s_{i+c} (or s_n if $i + c > n$) with a uniform distribution when $i \neq n$. At s_n, the agent transits to either s_n or s_1 with probability 0.5. Thus, with $c = k$, the transition by taking an action at a state at most leads to $k + 1$ different states.

At $c = 1$, the transition matrix is the sparsest and thus the MDP has the worst connectivity in $\{M_k^n\}$ with the same n, while at $c = n - 1$ the transition is the densest and thus the MDP has the best connectivity. For example, the transition matrix for $n = 5$ with $c = 1, ..., 4$ are

$$\begin{bmatrix} 1/2 & 1/2 & 0 & 0 & 0 \\ 0 & 1/2 & 1/2 & 0 & 0 \\ 0 & 0 & 1/2 & 1/2 & 0 \\ 0 & 0 & 0 & 1/2 & 1/2 \\ 1/2 & 0 & 0 & 0 & 1/2 \end{bmatrix}, \begin{bmatrix} 1/3 & 1/3 & 1/3 & 0 & 0 \\ 0 & 1/3 & 1/3 & 1/3 & 0 \\ 0 & 0 & 1/3 & 1/3 & 1/3 \\ 0 & 0 & 0 & 1/2 & 1/2 \\ 1/2 & 0 & 0 & 0 & 1/2 \end{bmatrix}, ..., \begin{bmatrix} 1/5 & 1/5 & 1/5 & 1/5 & 1/5 \\ 0 & 1/4 & 1/4 & 1/4 & 1/4 \\ 0 & 0 & 1/3 & 1/3 & 1/3 \\ 0 & 0 & 0 & 1/2 & 1/2 \\ 1/2 & 0 & 0 & 0 & 1/2 \end{bmatrix}.$$

The reward is 1 when the agent moves from s_n to s_1, and 0 otherwise. The discount factor γ is 0.9.

We fixed $n = 20$ and conducted the following experiment to obtain the transformation bias of $\hat{V}(s_1)$. In each run, we collected $m = 20$ transition data (s, s') at each state s (and thus 400 data in total in each run), used them to construct a model MDP $\hat{M}$, then

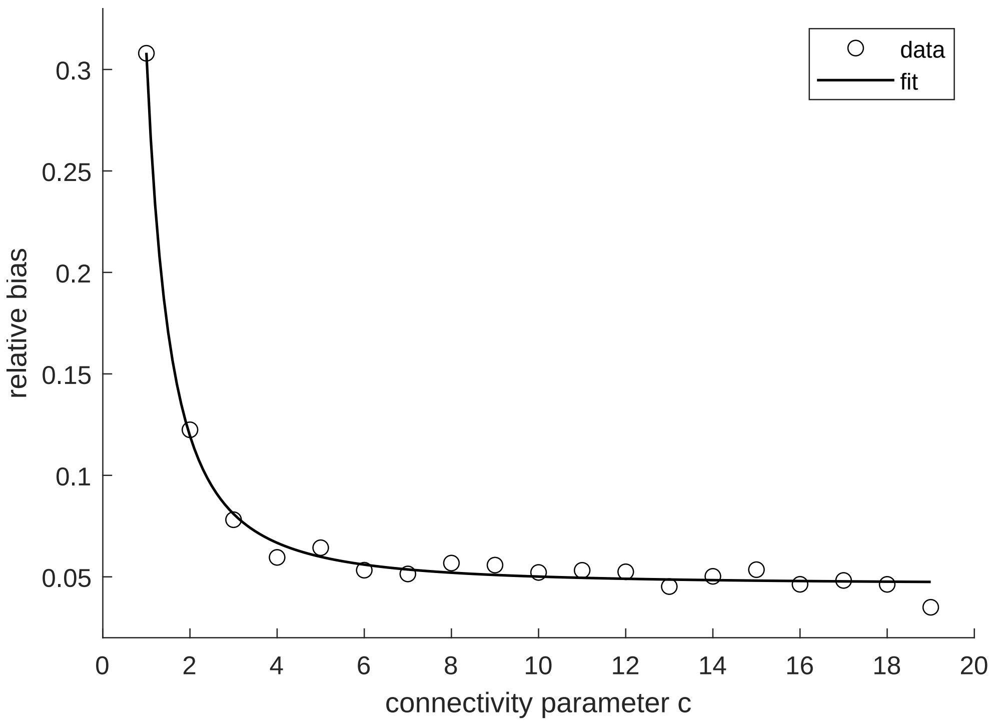

Figure 4.2: Bias vs connectivity.

computed $\hat{V}(s_1)$ by solving Bellman equation.[1] Such process was repeated 1000 times and the relative bias $|\frac{\mathbb{E}[\hat{V}(s_1)]-V(s_1)}{V(s_1)}| \times 100\%$ was computed.[2]

The relative biases under different c are shown by the circle points in Figure 4.2. The relative bias was 30.8% at $c = 1$, but was only around 5% at $c \geq 6$. This clearly shows that lower connectivity leads to larger transformation bias, and vice versa.

If an RL problem is closer to fully connected (i.e. high connectivity, dense transition matrix), then the risk of having a large transformation bias is low. However, in real-world applications, it is very often that each state-action pair only leads to a limited subset of states (i.e. low connectivity, sparse transition matrix). For example, in a Shogi playing task, the next state must comes from playing a legal move at the current state. Although the branching factor of Shogi is relatively large (about 80) among board games, it is almost negligible compared to its state space (10^{71}) [Iida et al., 2002]. In a robot navigation task,

[1]Note that in this chapter we do not consider the problem of data demand fulfilment that has been elaborated Chapter 3. Rather, we simply consider the case where the data demand is fulfilled perfectly, i.e. no excess data is obtained beyond demand $d = m$. This does not affect the generality of our results.

[2]The bias studied here is in fact a negative one. To present in a more intuitive way, the curves in Sections 4.3.2-4.3.4 show the absolute value of the bias.

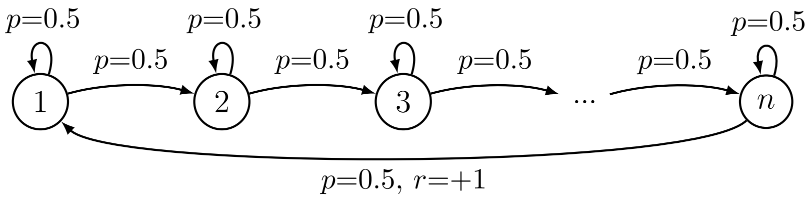

Figure 4.3: Chain MDP with n states.

the robot can only be transited to an adjacent location after each movement action, rather than anywhere in the whole space. Such cases correspond to $\{M_k^n\}$ with low connectivity c, and thus can be expected to have relatively large transformation bias.

4.3.3 The effect of the number of states

In this section, we study the impact of the number of states. Since a cycle is the essential unit for causing transformation bias, in this subsection we focus on chain MDPs, illustrated in Figure 4.3. A chain MDP with n states has a cycle of length n, i.e. $[s_1 s_2 ... s_n s_1]$. Chain MDPs also correspond to the case of $c = 1$ in $\{M_k^n\}$, which is the one with the least connectivity and thus the largest transformation bias.

We fixed the sample size per state-action $m = 20$ and obtained the estimated $\hat{V}(s_1)$ in chain MDPs under different number of states n, ranging from 5 to 800. The process was repeated 1000 times for each n to get the empirical distributions of $\hat{V}(s_1)$.

The relative bias of $\hat{V}(s_1)$ under different n is shown in Figure 4.4. The result was approximately $y = ae^{bx} + c$ with $a = -0.9955$ $(-1.001, -0.9905)$, $b = -0.008263$ $(-0.008342, -0.008185)$, and $c = 1.001$ $(0.9991, 1.002)$, in which the numbers in parentheses give the 95% confidence bounds to the coefficients. The R-squared goodness of fit was 0.9998. The curve shows a fast growth at lower n, while converges to about 1 with higher n.

The relative bias being 1 means that the bias of $\hat{V}(s_1)$ was as large as the true value $V(s_1)$. To better understand what had happened, we obtained the distribution of $\hat{V}(s_1)/V(s_1)$ at $n = 800$, which is presented in Figure 4.5. As can be seen from the distribution, in these 1000 runs, almost all estimated $\hat{V}(s_1)$ were less than $0.01V(s_1)$ and

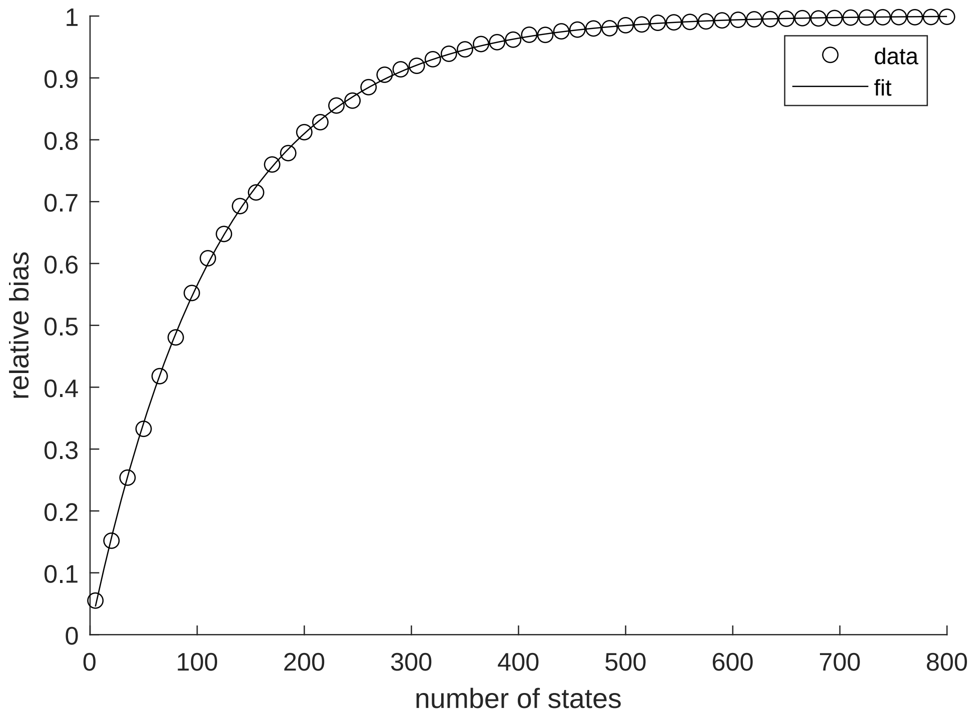

Figure 4.4: Bias vs number of states.

a large portion of them were less than $0.004V(s_1)$. Needless to say, $0.004V(s_1)$ is a terrible (if not completely useless) estimate of $V(s_1)$.

Note that with sample size per state-action $m = 20$, each estimated transition probability $\hat{p}$ has mean 0.5 and variance $0.5 \cdot (1 - 0.5)/20 = 0.0125$ [1], which makes a relatively accurate model of the true MDP. This implies that a slight error in the model MDP can be greatly magnified and bring into the state/action value estimates by the transformation bias. It also suggests that correcting the transformation bias might be more helpful than solely improving the model accuracy.

4.3.4 The effect of sample size per state-action

To have a clearer picture of how increasing the sample size affects transformation bias, we fixed the number of states $n = 200$ and changed the sample size per state-action m from 5 to 800. The process was repeated 1000 times for each m.

[1]The variance of a random variable X following binomial distribution $B(n, p)$ is $np(1 - p)$, thus the variance of $\hat{p} = X/n$ is $np(1 - p)/n^2 = p(1 - p)/n$.

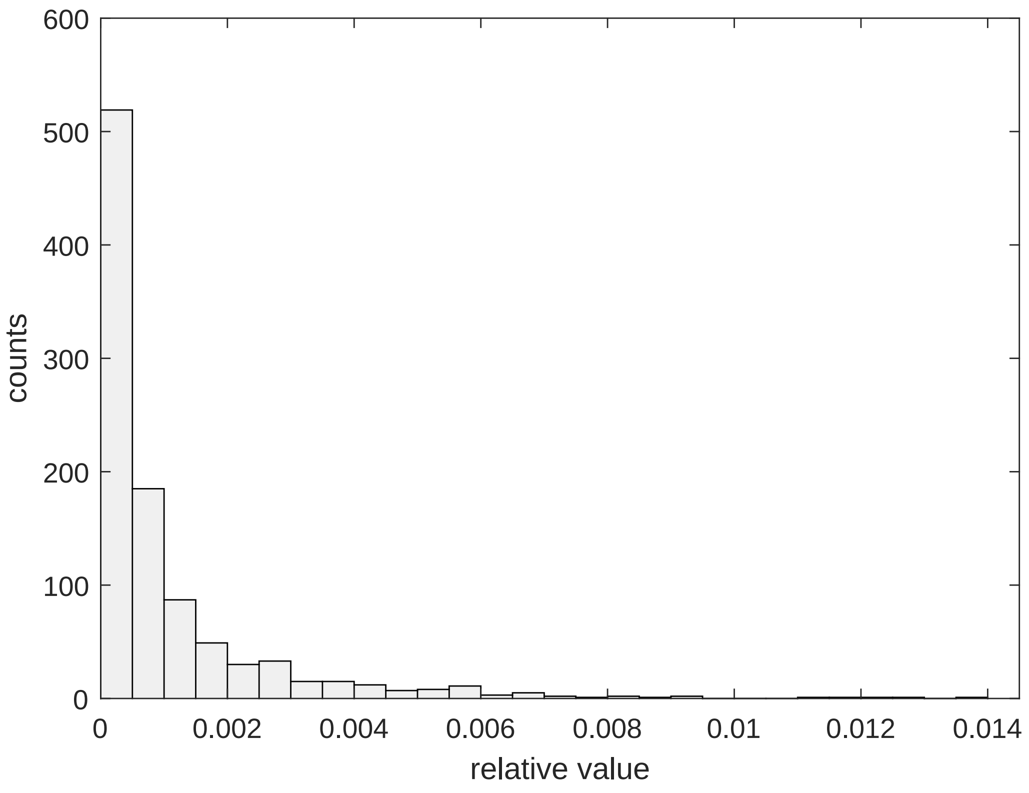

Figure 4.5: Histogram of relative value $\hat{V}(s_1)/V(s_1)$ in 1000 runs at $n = 800$.

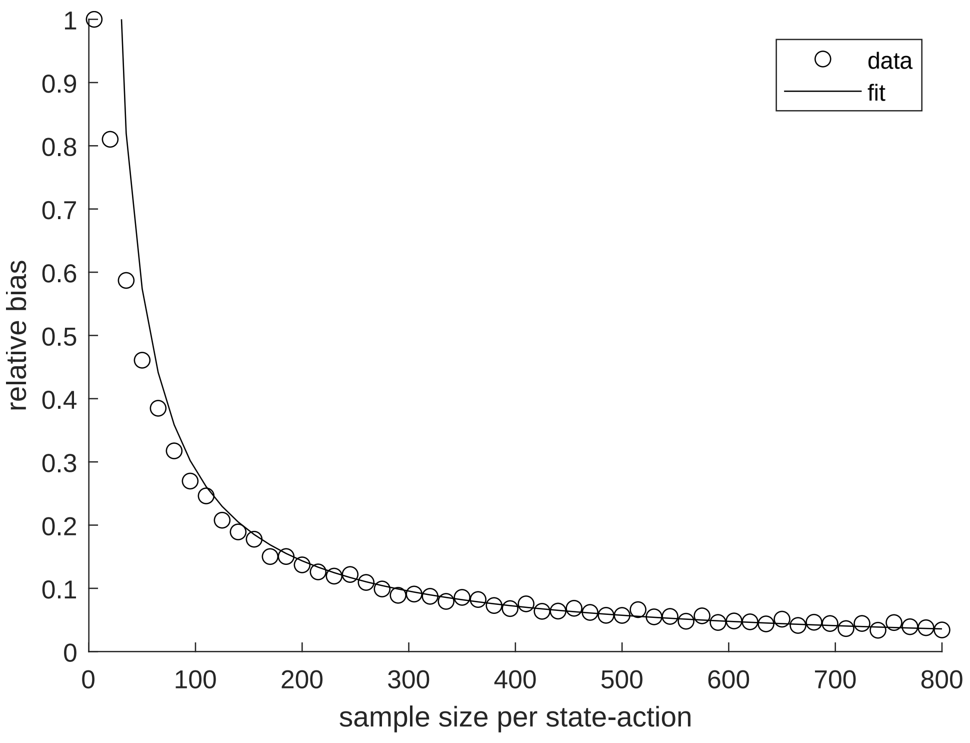

Figure 4.6: Bias vs sample size per state-action.

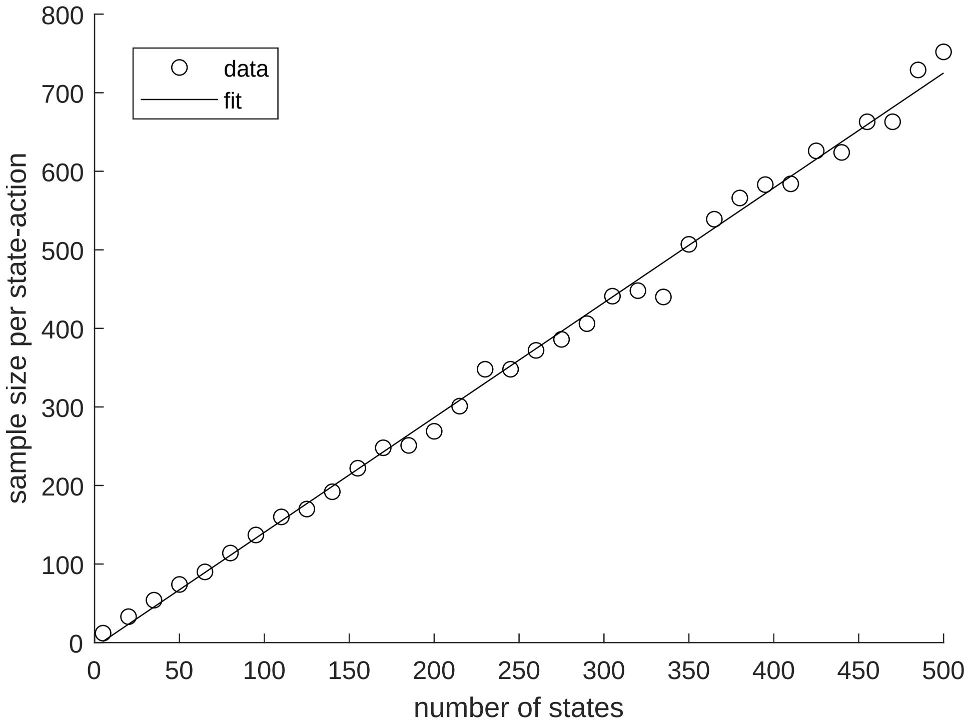

Figure 4.7: Sample size per state-action to achieve $\leq$ 10% relative bias vs number of states.

The relative bias of $\hat{V}(s_1)$ under different m is shown in Figure 4.6. The data was fitted for $x = g(y)$ and resulted in the fitting $x = \frac{\log a}{\log(1-by)}$ with $a = 0.9983$ $(-44.14, 46.14)$, $b = 5.929 \times 10^{-5}$ $(-1.565, 1.565)$, and the R-squared goodness of fit was 0.9712.

As we increased m, the bias decreased relatively fast when m is small, but was still more than 25% of the true value at around $m = 100$. It eventually reduced to near 0 at $m = 800$, but that is an exceptionally high price considering that each transition is just a simple Bernoulli event with $p = 0.5$.

From a practical perspective, it is interesting to know how large sample size is needed under different MDP size to maintain the same level of bias. Therefore, we conducted another experiment as follows. The number of states n changed from 5 to 500, and a binary search was used to find out the minimum sample size per state-action m needed to let the relative bias of 1000 runs no more than 10%.

The result is shown in Figure 4.7. The curve was, somewhat surprisingly, linear $y = ax + b$ with $a = 1.462$ $(1.428, 1.495)$, $b = -5.789$ $(-15.6, 4.018)$, and R-squared goodness of fit 0.9960, indicating that to maintain the same level of transformation bias, the sample size per state-action m should increase linearly with the number of states n. Since there are n states in chain MDPs and each requires $m = an + b$ data, this result suggests that $O(|\mathcal{S}|^2|\mathcal{A}|)$ data in total need to be collected just for preventing the transformation bias from growing with $|\mathcal{S}|$. It can be a substantial price in MDPs with large state spaces.

4.3.5 Putting the results together

The key findings from the previous subsections are highlighted as follows.

1. Both positive and negative transformation bias can occur for different value estimates in one single MDP.
2. MDPs with higher connectivity (the ones that are closer to complete graphs) tend to suffer less from the transformation bias, while the ones with lower connectivity suffer more. (Figure 4.2)
3. Transformation bias increases with the number of states, even if the model accuracy remains the same. (Figure 4.4)
4. With low connectivity and large number of states, transformation bias can become as large as the true value being estimated in some cases, making the value estimates completely useless. (Figure 4.5)
5. To maintain the same bias level, the sample size per state-action needs to be increased proportionally to the number of states. The total amount of data thus increases quadratically with the number of states. (Figure 4.7)

In addition, we have some well-fitted (in terms of R-squared score) relations between the the relative bias η, the number of states n, and the sample size per state-action m in chain MDPs:

- $\eta \approx -0.9955e^{-0.008263n} + 1.001$.
- $m \approx \frac{\log 0.9983}{\log(1-5.929\times10^{-5}\eta)}$.
- $m \approx 1.462n - 5.789$.

There is one surprisingly simple formula that can cover all three relations listed above, which is

$$\eta = C - K^{n/m}, \tag{4.1}$$

where $C > 0$ and $K \in (0, 1)$ are constants decided by the transition dynamics and the reward setting.

Although the empirical formula given in Equation 4.1 is only accurate in chain MDPs, in other general MDPs the relations between η, n and m is likely to be similar. Thus, we can use Equation 4.1 as a simple mathematical model of the transformation bias to estimate the potential scale of the bias before running experiments.

In particular, the formula can be used for tuning the sample size per state-action m in a multi-episode learning process. Suppose that we first run a learning algorithm with $m = m_1$, find the learning result disappointing, then run another episode of learning process with $m = m_2 > m_1$, and find the result somewhat better but still unsatisfying. We then need to run one more episode with $m = m_3 > m_2$ to further improve the learning result. To find a suitable $m = m_3$, we can represent the quality of the previous results at $m = m_1$ and $m = m_2$ as η_1 and η_2, put them into the formula above to obtain two equations for C and K^n, i.e.

$$\begin{cases} \eta_1 = C - (K^n)^{1/m_1} \\ \eta_2 = C - (K^n)^{1/m_2}. \end{cases}$$

We can obtain the values of C and K^n by solving these equations.[1] Then, by expressing the wanted result as η_3, we can make an informed guess of possibly suitable sample size per state-action setting $m = m_3$ by

$$m_3 = \frac{\log K^n}{\log(C - \eta_3)}.$$

The resulting m_3 might not be an accurate estimate to the actual minimum sample size

[1]Note that we only need to know K^n as a whole, because we usually do not need to change the number of states n. Also, the solution might not be unique and we might need to choose one based on some domain knowledge.

needed to achieve η_3 if the underlying MDP is very different from chain MDPs. However, we can still use this value as a reference when deciding whether we should just collect more data or whether we should improve the learning algorithm (by e.g. bias correction) in order to achieve η_3 in a limited-budget setting.

4.4 Bootstrap-based Transformation Bias Correction for model-based RL (BTBC)

Since transformation bias can be disastrous in some cases and relying on large samples can be costly, we propose a method called Bootstrap-based Transformation Bias Correction for model-based RL (BTBC for short) to help reduce the transformation bias without needing more data. BTBC is based on the resampling technique of bootstrap in statistics.

4.4.1 Single-depth BTBC

Before we propose the more complicated full-scale BTBC, we first introduce a single-depth version of it. Single-depth BTBC works as follows. Let $\mathbf{N}$ collectively represent all transition counts $\{N_{s,a,s'}\}$ and $\mathbf{N}_{*,*}$ represents sample sizes at each state-action pair $\{N_{s,a}\}$. By definition, transition counts $N_{s,a,s_1}, ..., N_{s,a,s_n}$ at (s, a) follow multinomial distribution with $N_{s,a}$ trials and probabilities $P(s_1|s, a), ..., P(s_n|s, a)$. For convenience, we write this as $\mathbf{N} \sim \text{Multinomial}(\mathbf{N}_{*,*}; P)$. Expressing the state value estimator as function $\hat{V}(\mathbf{N})$ and the unknown true value as $V(P)$, the bias of $\hat{V}$ is

$$\text{Bias}(\hat{V}) \overset{\text{def}}{=} \mathbb{E}[\hat{V}(\mathbf{N})] - V(P).$$

Since $\mathbb{E}[\hat{V}(\mathbf{N}) - \text{Bias}(\hat{V})] = \mathbb{E}\big[\hat{V}(\mathbf{N}) - \big(\mathbb{E}[\hat{V}(\mathbf{N})] - V(P)\big)\big] = V(P)$, we have that $\hat{V}(\mathbf{N}) - \text{Bias}(\hat{V})$ is an unbiased estimate of $V(P)$.

With only one instance of $\mathbf{N}$ we cannot directly compute $\mathbb{E}[\hat{V}(\mathbf{N})]$, nor do we know the true value $V(P)$. However, we can estimate $\text{Bias}(\hat{V})$ using bootstrap. Specifically, the following steps are taken:

- Estimate transition probabilities $\hat{P}$ from data $\mathbf{N}$ by $\hat{P}(s'|s,a) \overset{\text{def}}{=} N_{s,a,s'}/N_{s,a}$.
- Generate k bootstrap data $\mathbf{N}_1^+, \mathbf{N}_2^+, ..., \mathbf{N}_k^+$ from $\hat{P}$. Each $\mathbf{N}_i^+$ is a collection of transition counts such that $\mathbf{N}_i^+ \sim \text{Multinomial}(\mathbf{N}_{*,*}; \hat{P})$.
- Compute the estimated bias of estimator $\hat{V}$ by

$$\widehat{\text{Bias}}(\hat{V}) \overset{\text{def}}{=} \frac{1}{k}\left(\hat{V}(\mathbf{N}_1^+) + ... + \hat{V}(\mathbf{N}_k^+)\right) - \hat{V}(\mathbf{N}).$$

Lemma 4.1. $\lim_{k\to\infty} \widehat{Bias}(\hat{V})$ *is a consistent estimator of the unknown true Bias*$(\hat{V})$.

Proof. Since by definition $\hat{V}(\mathbf{N}) = V(\hat{P})$, we have $\lim_{k\to\infty} \widehat{\text{Bias}}(\hat{V}) = \mathbb{E}[\hat{V}(\mathbf{N}^+)] - V(\hat{P})$, which equals to the true $\text{Bias}(\hat{V}) = \mathbb{E}[\hat{V}(\mathbf{N})] - V(P)$ if $\hat{P} = P$. As $N_{s,a} \to \infty$ for all (s,a) we have $\hat{P} \xrightarrow{p} P$, thus $\lim_{k\to\infty} \widehat{\text{Bias}}(\hat{V})$ is a consistent estimator of the unknown true bias $\text{Bias}(\hat{V})$. $\square$

With the estimated bias of $\hat{V}$ computed from bootstrap, we have the bias-corrected estimator of V which is

$$\begin{aligned}\tilde{V} &\overset{\text{def}}{=} \hat{V}(\mathbf{N}) - \widehat{\text{Bias}}(\hat{V}) \\ &= \hat{V}(\mathbf{N}) - \left(\frac{1}{k}(\hat{V}(\mathbf{N}_1^+) + ... + \hat{V}(\mathbf{N}_k^+)) - \hat{V}(\mathbf{N})\right) \\ &= 2\hat{V}(\mathbf{N}) - \frac{1}{k}\left(\hat{V}(\mathbf{N}_1^+) + ... + \hat{V}(\mathbf{N}_k^+)\right).\end{aligned} \tag{4.2}$$

4.4.2 Remarks on single-depth BTBC

Readers might ask why not simply use the average of bootstrapped value estimates

$$\text{avg}[\hat{V}(\mathbf{N}_i^+)] = \frac{1}{k}\left(\hat{V}(\mathbf{N}_1^+) + ... + \hat{V}(\mathbf{N}_k^+)\right)$$

as the corrected value estimate, as what has been done in many other application of the bootstrap method. The answer is that in the context of bias correction, using $\text{avg}[\hat{V}(\mathbf{N}_i^+)]$

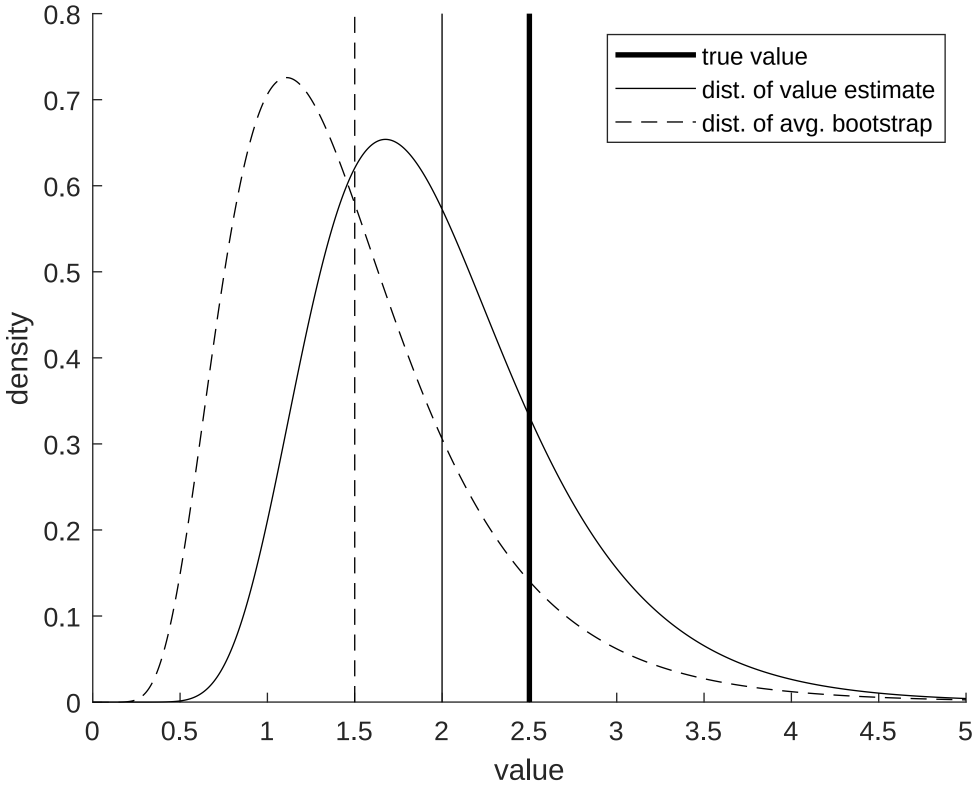

Figure 4.8: Illustration of wrong and correct use of bootstrap for bias corrections. A correct use of bootstrap leads to a $\tilde{V}$ close to the true V drawn by the thick vertical line, while a wrong use leads to one close to the dashed vertical line, which actually increases the bias.

is simply wrong. It is of great importance that Equation 4.2 not only differs from the simple avg$[\hat{V}(\mathbf{N}_i^+)]$, but also changes the sign to it when computing the bias-corrected estimate $\tilde{V}$.

An illustration of the wrong and the correct bias correction is given in Figure 4.8. In this figure, the true value $V = 2.5$ is shown as the thick vertical line. The estimated value $\hat{V}$ is biased with $\mathbb{E}[\hat{V}] = 2 < V$, which is given by the thin vertical line. The distribution of the value estimates is given by the thin curve. When using $\hat{V}(\mathbf{N})$ to generate bootstrap values $\hat{V}(\mathbf{N}_1^+)$,...,$\hat{V}(\mathbf{N}_k^+)$, there is a transformation bias between $\hat{V}(\mathbf{N}_i^+)$ and $\hat{V}(\mathbf{N})$ similar to the one between $\hat{V}(\mathbf{N})$ and V, i.e. $\mathbb{E}[\hat{V}(\mathbf{N}_i^+)] - \mathbb{E}[\hat{V}(\mathbf{N})] \approx \mathbb{E}[\hat{V}(\mathbf{N})] - V$. The distribution of the average bootstrap values thus is placed farther from V, as shown in the dashed curve in the figure, with its expectation $\mathbb{E}[\hat{V}(\mathbf{N}_i^+)] \approx \mathbb{E}[\hat{V}(\mathbf{N})] - (V - \mathbb{E}[\hat{V}(\mathbf{N})]) = 1.5$, drawn with the dashed vertical line.

Clearly, if we simply use $\text{avg}[\hat{V}(\mathbf{N}_i^+)] = \big(\hat{V}(\mathbf{N}_1^+) + ... + \hat{V}(\mathbf{N}_k^+)\big)/k$ as the corrected value estimate of V, then it not only fails to correct the bias in $\hat{V}$, it even *increases* the bias and thus is completely counterproductive. On the other hand, by using $\tilde{V} = \hat{V}(\mathbf{N}) + \big(\hat{V}(\mathbf{N}) - \text{avg}[\hat{V}(\mathbf{N}_i^+)]\big)$ in single-depth BTBC, we shift $\hat{V}(\mathbf{N})$ *back* towards V by $\hat{V}(\mathbf{N}) - \text{avg}[\hat{V}(\mathbf{N}_i^+)]$, which makes the resulting $\tilde{V}$ closer to the true value V.

Readers might also ask whether the estimated $\widehat{\text{Bias}}(\hat{V})$ of single-depth BTBC suffers from transformation bias or not, since its computation involves the Bellman equation. The answer is no. The process of BTBC estimating the transformation bias can be considered from another perspective: it pretends the estimated $\hat{V}(\mathbf{N})$ to be the ground truth, and use the bootstrap process to *deliberately* generate a transformation bias in $\hat{V}(\mathbf{N}^+)$. Since it already knows the "ground truth" $\hat{V}(\mathbf{N})$ and can generate arbitrarily many data sets $\mathbf{N}^+$, it can accurately compute the transformation bias of $\hat{V}(\mathbf{N}^+)$ seen as an estimator of $\hat{V}(\mathbf{N})$. This bias is then used as the estimate of the transformation bias $\widehat{\text{Bias}}(\hat{V})$ for $\hat{V}(\mathbf{N})$, in this case treated as the estimated value of interest rather than the ground truth. In other words, BTBC estimates the transformation bias of interest from another real transformation bias of similar property, and thus does not suffer from it, but in fact *benefits* from it.

4.4.3 Full BTBC

Since in practice we cannot generate infinite bootstrap samples, we have to use a finite k in BTBC. In this case, the computation of the expectation is no longer accurate and the resulting bias-corrected $\tilde{V}$ is still slightly biased. To reduce this remaining part of the bias, we can re-apply the bias correction process given above recursively to estimate $\text{Bias}(\widehat{\text{Bias}}(\hat{V})) \overset{\text{def}}{=} \mathbb{E}[\widehat{\text{Bias}}(\hat{V})] - \text{Bias}(\hat{V})$, then further to $\text{Bias}(\widehat{\text{Bias}}(\widehat{\text{Bias}}(\hat{V})))$, and so on, until the remaining bias is negligible. This recursive process leads to what we call full BTBC, in contrast to the single-depth BTBC. (In the remainder of this thesis, when we say BTBC we refers to the full BTBC, unless stated otherwise.)

The pseudo-code of full BTBC is given in Algorithm 3. BTBC can be put into any

Algorithm 3 Full BTBC for model-based RL

Input: number of bootstrap samples b, depth d, transition counts $\mathbf{N}$
Output: bias-corrected estimated state value function $\hat{\mathbf{V}}$

```
1: Compute transition probabilities $\hat{P}_{\mathbf{N}}$ using $\mathbf{N}$
2: $\hat{\mathbf{V}}$ = BellmanSolution($\hat{P}_{\mathbf{N}}$)
3: for $i = 1$ to $d$ do
4:     $\hat{\mathbf{V}} = \hat{\mathbf{V}}+$ RecursiveBootstrap($b$, $i$, $\mathbf{N}$)
5: end for
```

Algorithm 4 RecursiveBootstrap(b, d, $\mathbf{N}$)

Input: number of bootstrap samples b, depth d, transition counts $\mathbf{N}$ (not necessarily the original input to Algorithm 1)
Output: ($-$bias) of the current estimator

```
1: Compute transition probabilities $\hat{P}_{\mathbf{N}}$ using $\mathbf{N}$
2: if $d == 1$ then
3:     $\mathbf{\Theta}$ = BellmanSolution($\hat{P}_{\mathbf{N}}$)
4: else
5:     $\mathbf{\Theta}$ = RecursiveBootstrap($b$, $d-1$, $\mathbf{N}$)
6: end if
7: $\mathbf{T} = \mathbf{0}$
8: for $j = 1$ to $b$ do
9:     Generate bootstrap data $\mathbf{N}^+$ using $\hat{P}_{\mathbf{N}}$, i.e. $N^+_{s,a,*} \sim$ Multinomial($N_{s,a}; \hat{P}_{\mathbf{N}}(*|s,a)$)
10:    if $d == 1$ then
11:        Compute $\hat{P}_{\mathbf{N}^+}$ using $\mathbf{N}^+$
12:        $\mathbf{T} = \mathbf{T} +$ BellmanSolution($\hat{P}_{\mathbf{N}^+}$)$/b$
13:    else
14:        $\mathbf{T} = \mathbf{T} +$ RecursiveBootstrap($b, d-1, \mathbf{N}^+$)$/b$
15:    end if
16: end for
17: Return ($\mathbf{\Theta} - \mathbf{T}$)
```

model-based RL algorithm by replacing its original planning method (e.g. Value Iteration) with Algorithm 3. The original planning method is used in Algorithms 3 and 4 in the form of $\hat{V} = \text{BellmanSolution}(\hat{P})$.

Algorithm 4 is a helper method for Algorithm 3 which recursively computes the bias at different depths. At depth $d = 1$, Algorithm 4 estimates the biases of every state values; at $d = 2$, it estimates the biases of the estimated biases at $d = 1$, and so on. All the biases are eventually passed to Algorithm 3 and are used to correct the biases of the state value estimates. Note that each call of Algorithm 4, including the recursive ones, creates its own b sets of bootstrap data $\mathbf{N}^+$, so that no dependency in bootstrap data occurs between different calls.

The drawback of such design is that the total number of BellmanSolution calls in Algorithm 4 is $O(b^d)$, which can be very expensive with large d. However, our empirical results on BTBC (presented in Section 4.5) suggest that setting of $1 \leq b \leq 2$ and $1 \leq d \leq 4$ might be sufficient for bias correction in practical use. In this case, $O(b^d)$ becomes just a constant factor, and thus as long as the BellmanSolution method used in BTBC is computationally efficient, BTBC is also computationally efficient.

4.5 Empirical Evaluation of BTBC

4.5.1 Parameters of BTBC

There are two parameters in BTBC, the number of bootstrap samples b and the maximum depth of recursion d. To see how these parameters affect the effectiveness of bias correction, we conducted sensitivity analysis for BTBC executed on chain MDPs, which are shown to have relatively large bias in Section 4.3.

We first tested the effectiveness of BTBC under different number of bootstrap samples b, ranging from 1 to 15. The depth parameter d was set to 2. The sample size per state-action pair m was set to 20, while the number of states $n = 100$. The (signed) relative error

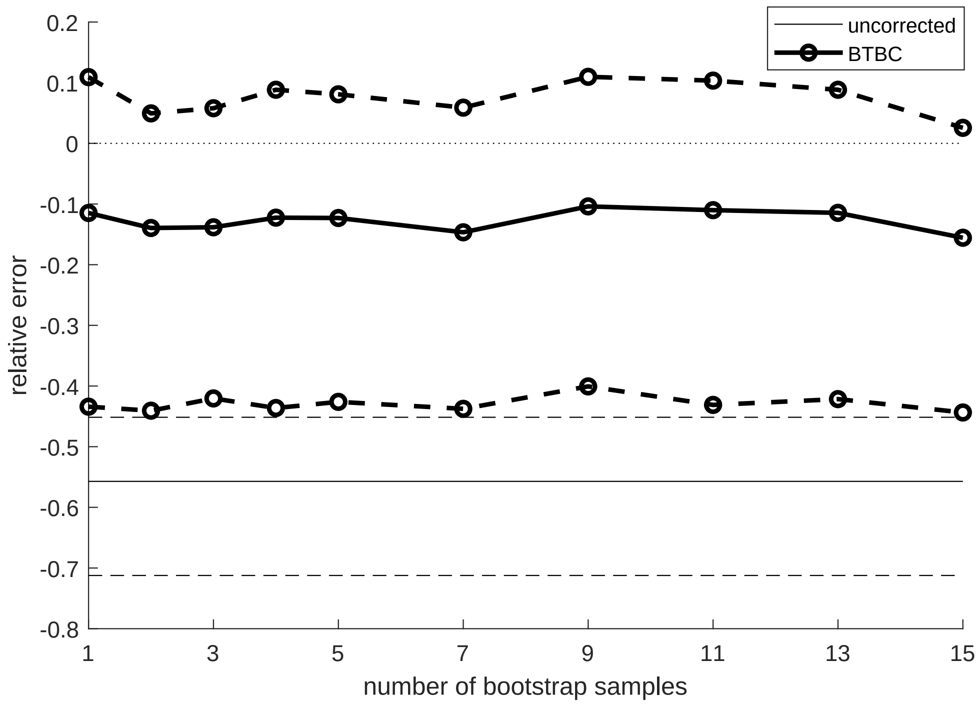

Figure 4.9: Signed relative error vs bootstrap size.

$\frac{\hat{V}(s_1)-V(s_1)}{V(s_1)}$ of the state value estimate $\hat{V}(s_1)$ was collected in each run. The experiment was repeated 1000 times under each b to get the distribution of the signed relative errors.

The result is shown in Figure 4.9. The thick and thin lines represent the results for BTBC and uncorrected estimator, respectively. The solid lines are the average relative error (i.e. empirical relative bias) while the dashed lines are the 0.25 and 0.75 quantiles. The uncorrected value estimate $\hat{V}(s_1)$ had bias about $-0.56V(s_1)$ and thus $\mathbb{E}[\hat{V}(s_1)] \approx 0.44V(s_1)$, indicating a remarkable bias. BTBC reduced the bias to about $-0.13V(s_1)$, and thus the bias-corrected estimate has $\mathbb{E}[\tilde{V}(s_1)] \approx 0.87V(s_1)$ which is much closer to the real value $V(s_1)$ than the uncorrected one.

Somewhat surprisingly, changing the number of bootstrap samples b did not lead to any significant difference in relative bias. Even the very aggressive $b = 1$ could reduce as much bias as $b = 15$. After a closer look at the result we noticed that the variance of the relative error became slightly smaller (from 0.1996 to 0.1927) when b increased from 1 to 2. We then repeated the whole experiment under $b = 1$ and 2 for 15 times, and observed this small improvement consistently, with a 8.89% reduction of variance on

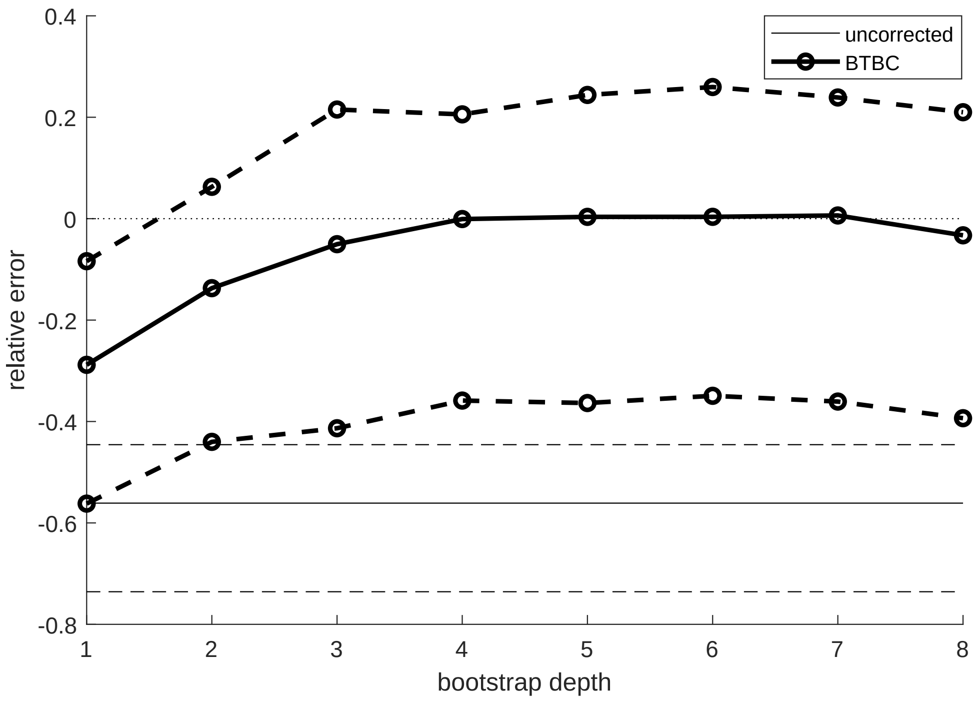

Figure 4.10: Signed relative error vs bootstrap depth.

average when b changed from 1 to 2. This suggests that increasing b reduces the variance of bias-corrected value estimates, but does not have strong impact on the bias itself.

The effectiveness of BTBC under small bootstrap size b is an interesting phenomenon because in Section 4.4 we have shown that theoretically BTBC should work better with a larger $b = k$. Our hypothesis is that the transformation bias is "distributed" over the whole transition dynamics and thus having a few bootstrap models in model-based RL is effectively having a large number of bootstrap samples in the case of univariate estimation. This however still lacks firm evidence and is left to future work.

The second parameter investigated was the bootstrap depth d. We tested BTBC under d ranging from 1 to 8, each repeated 1000 times, with number of bootstrap samples $b = 2$, sample size $m = 20$, and number of states $n = 100$.

The result is shown in Figure 4.10. Clearly, the bootstrap depth d had more impact on the effectiveness of bias reduction than the bootstrap size b. At $d = 1$, the relative bias was 28.8%, which is still very large, while at $d = 4$ it was reduced to less than 0.1%, which means that $d = 4$ is sufficient in this setting to completely remove the bias of $\hat{V}(s_1)$.

The effectiveness of BTBC under small depth d can be explained as follows. The recursive bootstrap described in Algorithm 4 can be seen as a process of expanding the bias as an infinite series and correcting the terms one by one, and thus correcting only the first several terms (i.e. using a small d) can often provide sufficiently good results. The idea of treating bootstrap as expansion can be found in [Hall, 1992].

Considering that using a larger b and d increases the computational cost, we suggest setting $1 \leq b \leq 2$ and $1 \leq d \leq 4$ in practical use.

4.5.2 Effectiveness and scalability of BTBC

To show the effectiveness of BTBC in correcting transformation bias, we compared it with the second-order approximation method proposed by [Mannor et al., 2007]. Since their bias estimation method relies on full knowledge about the true transition P, it is not directly applicable in general RL setting where P is unknown. Thus, we used the estimated transition $\hat{P}$ in place of P when estimating the bias using their approximation. The resulting bias-corrected estimator is denoted Mannor07 in this section.

The experiment was conducted in chain MDPs. We set the sample size per state-action $m = 20$ and changed the number of states n from 5 to 120. The parameters of BTBC were set to number of bootstrap samples $b = 1$ and depth $d = 3$ based on the recommendation of the last section. The relative error $\frac{\hat{V}(s_1)-V(s_1)}{V(s_1)}$ of the state value estimate $\hat{V}(s_1)$ in 1000 runs for each n was collected. The results of 1000 runs were then put into 20 groups (50 runs each) to compute the standard deviation of the average relative error.

Figure 4.11 shows the average relative errors of uncorrected estimator, BTBC, and Mannor07. The shaded areas show the values within one standard deviation of the average relative errors. It can be seen from the curves that both BTBC and Mannor07 significantly reduced the transformation bias, but BTBC did much better. At $n = 120$, the uncorrected estimator had bias $-0.645V$, Mannor07 reduced it to $-0.268V$, while BTBC estimator only had $-0.115V$. The curves also show that the bias of the uncorrected estimator grew quickly with the number of states, while for BTBC it grew notably slower, which indicates

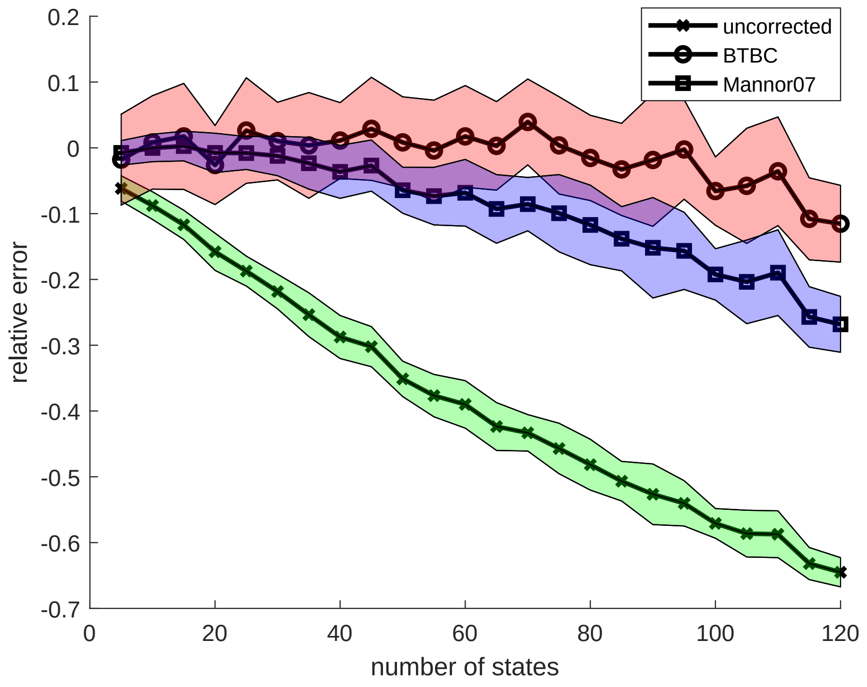

Figure 4.11: Average relative error vs number of states.

that BTBC has good scalability with respect to the MDP size.

To see how well BTBC performs when sample size per state-action is particularly small, we conducted another experiment where the number of states n was fixed to 60, and the relative errors under different sample size per state-action m ranging from 8 to 20 was tested. BTBC still used $b = 1$ and $d = 3$.

Note that with a small m such as 8 and 9, there is a remarkable probability[1] that some $N_{i,i+1} = 0$, which indicates that the agent fails to reach the goal s_n of the chain MDP. In this case, the estimated value becomes constantly 0 and no bias correction method can change it. Since it is more of a data collection problem than bias correction, whenever such cases happened in this experiment, we considered it a invalid run and reran the process until all $N_{i,i+1} \neq 0$. The results of 1000 valid runs for each m was collected and put into 20 groups to compute the standard deviation.

Figure 4.12 shows the average relative error of each estimators under different settings

[1]At $m = 8$ and $n = 60$ the probability of $N_{i,i+1} = 0$ for at least one i is $1 - (1 - (1 - 0.5)^m)^n = 1 - (1 - 0.5^8)^6 0 = 0.2093$. Thus, in every 1000 runs there are about 209 runs that have at least one $N_{i,i+1}$ being 0.

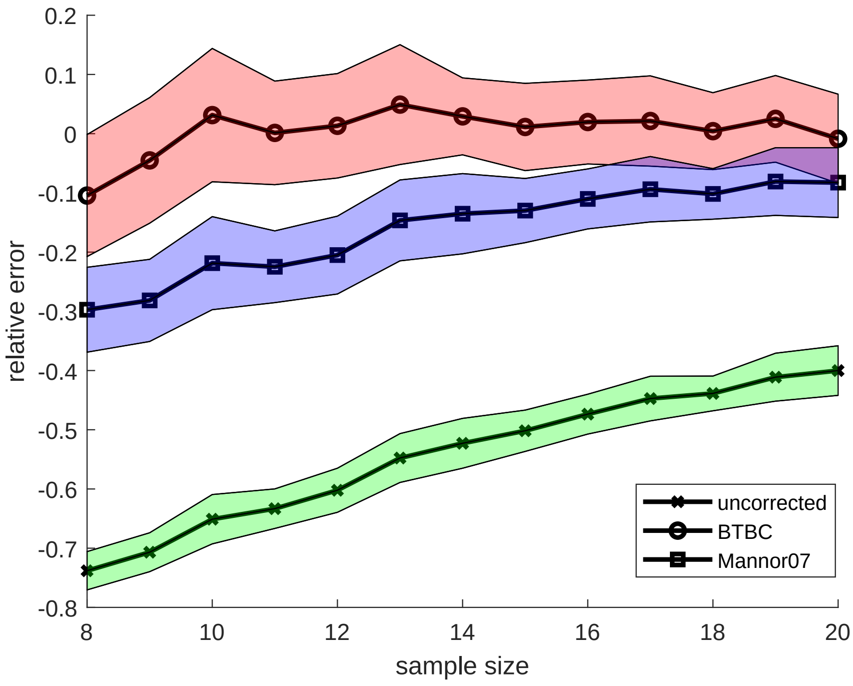

Figure 4.12: Average relative error vs sample size per state-action pair.

of m. Clearly, BTBC worked well even under a small sample size per state-action. The bias of BTBC was $-0.105V$ at $m = 8$, but quickly reduced to near-0 as m increased. Mannor07 performed much worse, where the bias was $-0.297V$ at $m = 8$, and even with $m = 20$ there was still a notable bias $-0.083V$, which was worse than BTBC at $m = 9$.

The ineffectiveness of Mannor07 at small sample size is due to it using an estimator of the transformation bias that in itself suffers from transformation bias (i.e. nonlinear with respect to $\hat{P}$). BTBC does not suffer from this problem because it estimates the unknown bias by deliberately generating another transformation bias in a known environment, as explained in Section 4.4.2.

The effect of increasing the sample size for the uncorrected estimator can also be observed from Figure 4.12 as well. The bias for the uncorrected estimator was $-0.651V$ at $m = 10$ and was $-0.400V$ at $m = 20$, which means that even doubling the sample size did not help reduce the bias by half. Even Mannor07 at $m = 8$ was better than the uncorrected at $m = 20$, and thus we recommend conducting bias correction in model-based RL whenever obtaining data is expensive.

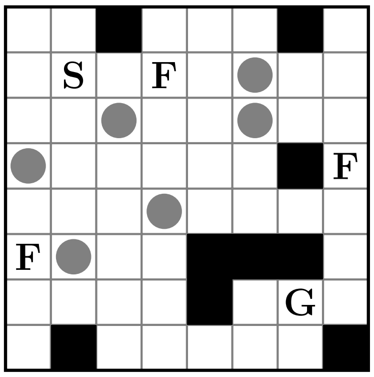

Figure 4.13: The maze problem.

4.5.3 Performance in maze domain

To show the effectiveness of BTBC in more complicated MDPs, we tested BTBC in a maze problem shown in Figure 4.13. In this problem, the task is to find a policy that can travel safely from the start point (represented by "S" in the figure) to the goal ("G"), collecting all three flags ("F") on the way, and avoiding the traps (circles) and walls (black blocks). State is represented by a vector (x, y, f_1, f_2, f_3) containing both the position (x, y) and the boolean values f_1, f_2, f_3 representing whether the corresponding flag has been obtained or not. The agent moves to the intended direction with probability 0.7 and to the other three directions with probability 0.1 randomly. It receives reward 100^k at the goal, where k is the number of flags collected, then loses all flags and returns to the start. If it walks into a trap, it loses all flags and returns to the start without reward. Discount factor γ is 0.9.

The signed relative bias of the uncorrected and BTBC estimates of the optimal state value $\hat{V}^*(s_{\text{start}})$ under different sample sizes per state-action m, ranging from 5 to 15, were collected. We did not test Mannor07 in this experiment due to its substantial computational cost. Each case was repeated 1000 times and put into 10 groups to draw the box plot.

The result is shown in Figure 4.14. At $m = 5$, the median of the relative bias was greatly reduced from 28.0% to 1.5% by BTBC. Even the sample size was tripled to 15, the median of the relative bias for uncorrected estimator was still 9.7%, far worse than

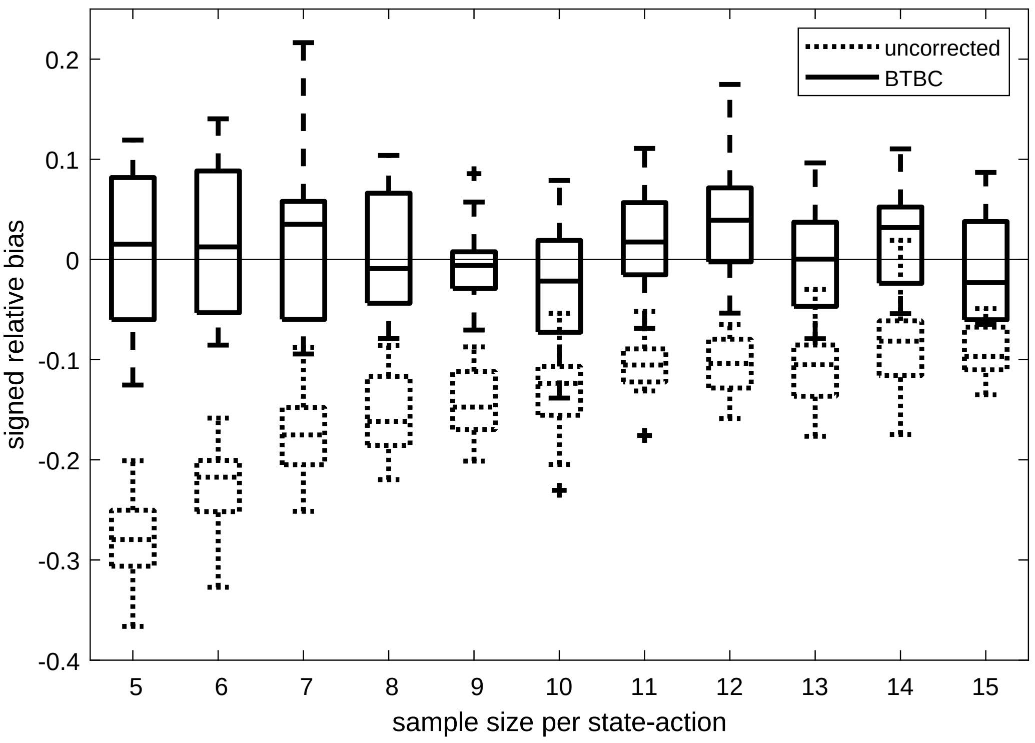

Figure 4.14: Performance of BTBC in the maze problem.

BTBC under $m = 5$. The p-value of two-sample t-test on uncorrected vs BTBC under $m = 15$ was 5.35×10^{-4}, while for uncorrected under $m = 15$ vs BTBC under $m = 5$ it was 1.79×10^{-3}, thus in both cases the BTBC estimator had significantly less bias than the uncorrected one at the 0.5% significance level.

4.6 Regarding transformation bias in model-free RL

In previous sections of this chapter, we have studied the transformation bias in model-based reinforcement learning. Before we summarise our results presented in this chapter, we digress a little and discuss the transformation bias in model-free RL.

We start from the mathematical connection between model-free and model-based RL, which is given by [Sutton and Barto, 2018]. Concretely, consider using Equation 2.1 to estimate state values from trajectory $\psi = S_1, A_1, R_1, \ldots$. Since

$$\begin{aligned}
\hat{V}^\pi(s) &= \sum_{s' \in \mathcal{S}} \hat{P}(s'|s, \pi(s))\big(\hat{R}(s, \pi(s), s') + \gamma \hat{V}^\pi(s')\big) \\
&= \sum_{s' \in \mathcal{S}} \frac{N_{s,\pi(s),s'}}{N_{s,\pi(s)}} \big(\hat{R}(s, \pi(s), s') + \gamma \hat{V}^\pi(s')\big) \\
&= \frac{1}{N_{s,\pi(s)}} \cdot \sum_{(S_k, A_k) = (s, \pi(s))} \big(R_k + \gamma \hat{V}^\pi(S_{k+1})\big),
\end{aligned}$$

it is clear that $\hat{V}^\pi$ of model-based estimator is essentially a cumulative moving average of $T_k \stackrel{\text{def}}{=} \big(R_k + \gamma \hat{V}^\pi(S_{k+1})\big)$ over the trajectory. Re-writing the above summation in an incremental style, we get the mathematically equivalent update rule

$$\hat{V}^\pi_k(S_k) = \frac{n_k - 1}{n_k} \hat{V}^\pi_{k-1}(S_k) + \frac{1}{n_k} T_k,$$

where n_k stands for the sample size $N_{S_k, \pi(S_k)}$ at step k in trajectory ψ. Then, by further changing the weight $\frac{n_k - 1}{n_k}$ and $\frac{1}{n_k}$ to $(1 - \alpha)$ and α respectively, where $\alpha \in (0, 1)$ is a parameter called learning rate, we get the *exponentially weighted moving average* [Hunter, 1986] as an approximation to the cumulative moving average of T_k, i.e.

$$\tilde{V}^\pi_k(S_k) = (1 - \alpha) \tilde{V}^\pi_{k-1}(S_k) + \alpha T_k. \tag{4.3}$$

Equation 4.3 is the well-known model-free Temporal Difference (TD) estimator proposed by [Sutton, 1984]. It does not need to keep track of $N_{s,a,s'}$ and $N_{s,a}$, has less per-update computation time, and converges with probability 1 to the true value under certain conditions [Dayan, 1992]. There exist several other model-free estimators such as Q-learning [Watkins, 1989] and Sarsa [Rummery, 1995], all of them have similar characteristics.

Now, let us consider the transformation bias in model-free RL. As elaborated in previous sections, the transformation bias of model-based RL is due to the matrix inversion involved in solving the linear system given by the Bellman equation. In model-free RL, we do not compute matrix inversion, and thus the transformation bias seems to be absent.

However, as discussed above, the state/action value estimators in model-free RL are just approximations to the corresponding model-based estimators (by e.g. changing cumulative moving average to exponentially weighted moving average). Let $\check{V}$ denotes the model-free estimator and $\hat{V}$ be the corresponding model-based estimator. $\check{V}$ being approximation to $\hat{V}$ means that given the same data set, the stationary mean of $\check{V}$ is $\hat{V}$ [Perry, 2011]. Since the model-based $\hat{V}$ has transformation bias, this means the model-free $\check{V}$ at best converges to a biased estimate of the true V. In this sense, it is reasonable to say that model-free estimators suffer from an *implicit* transformation bias.

Since model-free RL does not store the estimated transition function, we cannot directly use bootstrap to estimate this implicit transformation bias as in BTBC for model-based RL. However, if the interaction history is stored, which is done in many RL algorithms using batch updates, then it might be possible to conduct bootstrap by resampling these data. The effectiveness of such method is left to future work.

4.7 Chapter summary

In this chapter, we have pointed out in Section 4.2 why transformation bias is harmful and should be corrected in model-based RL. We have presented in Section 4.3 our study on the relation between the scale of the transformation bias η, the transition dynamics, the sample size per state-action m, and the number of states n, which eventually leads to an empirical formula $\eta = C - K^{n/m}$ for chain MDPs, where C and K are constants decided by the transition dynamics. We have shown that the transformation can be remarkably large when the sample size does not grow as fast as the number of states, which makes learning particularly difficult when sample size is small. These results answer the research questions Q2.1 ("How does the transformation bias relate to sample efficiency?") and Q2.2 ("In what cases is the transformation bias remarkable and needs correction?") in Section

1.3.2 and lead to contribution 3 in Section 1.4.[1]

We have then proposed Bootstrap-based Transformation Bias Correction (BTBC) in Section 4.4 for correcting the transformation bias of model-based RL. We have presented our empirical results in Section 4.5 to show that BTBC can significantly reduce the transformation bias even with a small sample size, at the cost of an increased computational cost. This answers the research questions Q2.3 ("How to correct the transformation bias when sample size is small?") in Section 1.3.2 and leads to contribution 4 in Section 1.4.[2]

We have also briefly discussed the implicit transformation bias in model-free RL in Section 4.6, which can be an interesting topic for future work.

[1]Contribution 3: "We present an extensive study on the transformation bias of state/action value estimates, revealing the relationship between the scale of the bias and relevant factors including sample size, scale of MDP, and transition dynamics."

[2]Contribution 4: "We propose the Bootstrap-based Transformation Bias Correction (BTBC) method which can significantly reduce transformation bias of state/action value estimates even if sample size is small."

CHAPTER 5

EFFECTIVENESS OF POLICY SELECTION IN MODEL-BASED RL AND ITS IMPROVEMENT

5.1 Introduction

In the last chapter we have discussed how the transformation bias affects the result of value comparison, and also have elaborated how to reduce the bias without collecting a larger sample. However, we have not gone into detail about how to measure the effectiveness of value comparison, or how the proposed bias correction method improves the effectiveness. This chapter elaborates our study regarding value comparison and policy selection.

Reinforcement learning can be seen as an iterative process of value estimation and value comparison. When value comparison happens at policy level, i.e. comparing $V^{\pi}(s)$ vs $V^{\pi'}(s)$ or $Q^{\pi}(s,a)$ vs $Q^{\pi'}(s,a)$, it is a *policy selection problem* where the algorithm chooses one policy between π and π'. Value comparison can also happens at action level, which is a very similar problem and will not be elaborated in this chapter.

The question then arises whether we can use the quality of value estimation (accuracy, unbiasedness, etc.) to represent the effectiveness of policy selection. The answer is negative. Doroudi et al. [2017] provided a good example for multi-armed bandit problem, which we slightly modified to fit the setting of model-based RL, presented as follows.

Example 5.1. [Doroudi et al., 2017] Consider an one-state two-action MDP with self-transition and stochastic reward, where action a_1 yields reward r with probability 1, while action a_2 yields reward 1 with probability p and reward 0 with probability $1-p$, where $r < p < 0.5$. Discount factor is $\gamma > 0$. Policy π_1 selects a_1, while policy π_2 selects a_2. By the Bellman equation $V^{\pi_1} = r/(1-\gamma)$ and $V^{\pi_2} = p/(1-\gamma)$, and thus $V^{\pi_1} < V^{\pi_2}$. However, with one observation for each action, the model-based estimated value for π_1 is $\hat{V}^{\pi_1} = r/(1-\gamma)$, while for π_2 it is $\hat{V}^{\pi_2} = 1/(1-\gamma) > \hat{V}^{\pi_1}$ with probability p and $\hat{V}^{\pi_2} = 0 < \hat{V}^{\pi_1}$ with probability $(1-p)$. Therefore, a naive policy selector selects the inferior π_1 with probability $(1-p) > 0.5$, which is worse than random guessing.

In the above example, both value estimates are unbiased, but a naive comparison between these two unbiased value estimates still leads to a result worse than random guessing. Thus, although as discussed in Section 4.2 unbiasedness of value estimates can be very useful, it is still not sufficient for guaranteeing good policy selection outcomes.

The reason behind the poor policy selection in the example above is the asymmetry of the distribution for $\hat{V}^{\pi_2}$. It has probability $p < 0.5$ to be larger than its true value, and has probability $(1-p) > 0.5$ to be less than both its true value and $\hat{V}^{\pi_1}$. Thus, its distribution is skewed towards the smaller value, making $\hat{V}^{\pi_2}$ more likely to be underestimated.

Note that $(1-\gamma)\hat{V}^{\pi_2}$ is binomial distributed $B(m, p)$, where m is the sample size. Since a binomial distribution is more asymmetric with a smaller m, this suggests that policy selection can be more problematic when sample size is small.

As reviewed in Section 2.5.1, Doroudi et al. [2017] proposed the notion of fairness for reinforcement learning, in which an algorithm is said to be fair if it outputs the optimal policy with probability larger than any other policies in any possible MDPs. However, they also showed that it is very difficult to achieve strict fairness. For two unfair selectors, their definition can only tell that both are unfair, and thus is not informative enough to be used in comparing the effectiveness of policy selectors.

Therefore, in this chapter, we propose the notion of *policy selection risk*, which is a quantitative metric and is more suitable for comparing different methods than the original fairness of [Doroudi et al., 2017] In particular, the family-wise policy selection risk measures the likelihood of a policy selector making wrong decisions in a family of

MDPs, and thus is more comprehensive than its instance-wise version. We also propose the family-wise unfairness metric as an extension of the strict fairness, which can be used in a manner similar to the family-wise policy selection risk to represent the overall (soft) fairness of a selector in a family of MDPs.

We propose a policy selector called Bootstrap-based Policy Voting (BPV) that works better than the naive model-based policy selector in terms of both family-wise policy selection risk and family-wise unfairness. We also propose a tournament-based policy refinement mechanism that can utilise the improved policy selectors, and apply both BPV and BTBC (proposed in the previous chapter) to the policy refinement mechanism to improve overall performance of model-based RL algorithms without needing to acquire a larger sample.

The remainder of this chapter is organised as follows. In Section 5.2, we provide our definition of the instance-wise and family-wise policy selection risk as well as the family-wise unfairness, and present the analysis for some basic selectors. In Section 5.3, we propose our Bootstrap-based Policy Voting method for policy selection. In Section 5.4, we propose two tournament-based policy refinement methods, Policy Voting Tournament (PVT) and Bias-corrected Tournament (BCT), for general model-based RL. In Section 5.5, we present our empirical results for these proposed methods. Finally, we summary this chapter in section 5.6.

5.2 Family-wise policy selection risk and unfairness

In this section, we give our definitions of the instance-wise and family-wise policy selection risk as well as the family-wise unfairness, analyse their properties and give some examples, and discuss their relation with the original fairness proposed by [Doroudi et al., 2017].

5.2.1 Pairwise policy selector

A policy selector W is a mapping $W : \Psi \times 2^{\Pi} \mapsto \Pi$, where Ψ is the set of all possible interaction histories (i.e. the data), Π is the set of all policies, and 2^{Π} is the power set of Π (i.e. the set of all subsets of Π). Thus, given a data set obtained from environment interaction and a set of candidate policies, a policy selector outputs one of the candidate policies that it considers the best. For simplicity, in this chapter we mainly discuss pairwise policy selectors where the number of the candidate policies is 2, and thus the selector only needs to choose between some $\{\pi_1, \pi_2\}$.

The random guesser W_{Ran} randomly chooses policies uniformly regardless of data:

$$W_{\text{Ran}}(\psi, \{\pi_1, \pi_2\}) \overset{\text{def}}{=} \text{Uniform}\{\pi_1, \pi_2\}. \tag{5.1}$$

The naive model-based policy selector W_{MB} is defined as

$$W_{\text{MB}}(\psi, \{\pi_1, \pi_2\}) \overset{\text{def}}{=} \begin{cases} \pi_1 & \hat{V}_{\psi}^{\pi_1} > \hat{V}_{\psi}^{\pi_2} \\ \pi_2 & \hat{V}_{\psi}^{\pi_1} < \hat{V}_{\psi}^{\pi_2} \\ \text{Uniform}\{\pi_1, \pi_2\} & \hat{V}_{\psi}^{\pi_1} = \hat{V}_{\psi}^{\pi_2}, \end{cases} \tag{5.2}$$

where $\hat{V}_{\psi}$ is the model-based value estimate $\hat{V}$ computed from data ψ for some unknown true value V of interest. By definition, the naive selector simply compares two estimated values and chooses the policy with the higher estimated value. Most model-based RL algorithms either explicitly use W_{MB} to decide their output policies, or implicitly use it through action selection.

5.2.2 Instance-wise policy selection risk

In Example 5.1, the naive policy selector chooses the better one of the two candidate policies with probability less than 0.5. Since random guessing between two policies chooses the better one with probability 0.5, the naive selector is worse than random guessing in this case.

The problem of choosing inferior policy with high probability is formulated as the fairness problem by [Doroudi et al., 2017]. However, as pointed out in Sections 2.5.1 and 5.1, their definition is too strict and is unsuitable for comparing different selectors. For example, their fairness cannot tell the difference between two fair selectors that choose π^* with probabilities 0.501 and 0.999, although the latter is obviously better than the former. Also, from a practical perspective, there is no real difference between a fair selector that chooses π^* with probability 0.501 and an unfair selector that chooses π^* with probability 0.499 in a pairwise policy selection problem. Therefore, although fairness can partly reflect the effectiveness of a selector, it is neither informative nor decisive. To better compare the selectors, we need a quantitative metric for measuring their effectiveness.

To this end, a straightforward idea is to directly use the probability of choosing the inferior policy in a pairwise policy selection problem instance, given as follows.

Definition 5.1. A pairwise selection problem instance is a tuple $\xi = (M, V^\pi, \{\pi^+, \pi^-\})$, where M is an MDP, V^π_M is some state/action value function of interest, π^+ and π^- are two policies s.t. π^+ is better than π^- with respect to V^π_M, i.e. $V^{\pi^+}_M > V^{\pi^-}_M$.

Definition 5.2. Let $\xi = (M, V^\pi, \{\pi^+, \pi^-\})$ be a selection problem instance. Let $\pi^W_M = W(\psi, \{\pi^+, \pi^-\})$ be the output of selector W given data ψ generated from M. Then the policy selection risk of W in ξ is $\text{Risk}(W; \xi) \overset{\text{def}}{=} \mathbb{P}(\pi^W_M = \pi^-)$.

By this definition, a policy selector W has a positive risk if it does not always choose the better policy π^+ among $\{\pi^+, \pi^-\}$. Although zero-risk under this definition is generally not achievable for non-oracle selectors, the size of the risk can nevertheless be used to indicate the (in)effectiveness of the policy selector in a specific problem instance.

5.2.3 Family-wise policy selection risk

The problem with $\text{Risk}(W; \xi)$ given in Definition 5.2 is, because it is an instance-wise measurement, a selector can have lower risks in some selection problem instances while have higher risks in some other instances than another selector. For example, consider the following variant of Example 5.1:

Example 5.2. Change the reward r in Example 5.1 to $p < r < 1$. In this case, since $V^{\pi_1} = r/(1-\gamma)$ and $V^{\pi_2} = p/(1-\gamma)$, policy π_1 that is inferior in the original example becomes superior than the alternative π_2. Since W_{MB} still chooses π_1 with probability $(1-p) > 0.5$, it becomes better than random guessing W_{Ran} in this case.

The strong connection between $\text{Risk}(W;\xi)$ and selection problem instance ξ makes it still difficult to use in comparing two selectors. Doroudi et al. [2017] required that a fair selector is no worse than W_{Ran} in *all* possible selection problems, but this requirement practically rules out every non-oracle selector because Example 5.1 can be made arbitrarily hard by using a sufficiently small r and p.

To address this, we propose *family-wise policy selection risk*, or family-wise risk for short, which considers the overall risk of a selector across a family of selection problem instances. To define the family-wise risk, we start with the definition of the MDP family as follows.

Definition 5.3. Let π_1 and π_2 be two policies for state space $\mathcal{S}$ and action space $\mathcal{A}$. A set of MDPs $\{M_{(x)}\}$ with state space $\mathcal{S}$ and action space $\mathcal{A}$ is said to be a (π_1, π_2)-family of MDPs with parameter $x \in [-x_{\max}, x_{\max}]$ if for each $x = k \in [-x_{\max}, x_{\max}]$, there is exactly one MDP $M_{(k)} \in \{M_{(x)}\}$ s.t. $V^{\pi_1}_{M_{(k)}} - V^{\pi_2}_{M_{(k)}} = k$, where V^{π}_{M} is some value function of interest for policy π in MDP M, and $x_{\max} > 0$.

By definition, $M_{(x)}$ with $x > 0$ are the MDPs in $\{M_{(x)}\}$ where $V^{\pi_1}_{M_{(x)}} - V^{\pi_2}_{M_{(x)}} > 0$ and thus π_1 is better, while $M_{(x)}$ with $x < 0$ are the ones where π_2 is better. $M_{(0)}$ is the MDP where π_1 and π_2 are equally good.

The family-wise policy selection risk is formally given as follows.

Definition 5.4. The family-wise policy selection risk of policy selector W with respect to $\{M_{(x)}\}$ over policies $\{\pi_1, \pi_2\}$ is defined as

$$\text{Risk}_{\{M_{(x)}\}}(W) \overset{\text{def}}{=} \int_{x=-x_{\max}}^{0} \mathbb{P}(\pi^W_{M_{(x)}} = \pi_1)\, dx + \int_{x=0}^{x_{\max}} \mathbb{P}(\pi^W_{M_{(x)}} = \pi_2)\, dx, \qquad (5.3)$$

given that $\mathbb{P}(\pi^W_{M_{(x)}} = \pi)$ as a function of x is integrable on $[-x_{\max}, x_{\max}]$.

Since by definition when $x \in [-x_{\max}, 0)$ we have $V^{\pi_1}_{M_{(x)}} - V^{\pi_2}_{M_{(x)}} < 0$, the first integral gives the overall risk of W in policy selection problem instances within $\{M_{(x)}\}$ where π_2 is the ground truth answer. Similarly, the second integral gives the overall risk in instances where π_1 is the ground truth answer.

Note that in Definition 5.3 we do not explicitly require the similarity between MDPs within the same family. Rather, we only require that there is some parameter x that can differentiate the MDPs and express the gap between the values of the two candidate policies. That being said, when we conduct analysis in practice, we should always choose meaningful families of MDPs in order to make the corresponding $\mathbb{P}(\pi^W_{M_{(x)}} = \pi)$ integrable. This can in general be achieved by fixing the transition core and change the reward function smoothly, or fixing the reward and change the transition smoothly. We will give an example of constructing a MDP family in the next section.

5.2.4 Analysis of family-wise policy selection risk

This section analyses the family-wise policy selection risk for some basic selectors, and points out several important properties of the family-wise risk. We start with the corollaries that give the family-wise risk for some trivial selectors.

Corollary 5.1. *Risk*$_{\{M_{(x)}\}}(W^*) = 0$ *for oracle selector* $W^* \overset{def}{=} argmax_\pi V^\pi$.

Proof. By definition $\mathbb{P}(\pi^{W^*}_M = \pi_1) = 0$ if $V^{\pi_1}_M - V^{\pi_2}_M = x < 0$, and $\mathbb{P}(\pi^{W^*} = \pi_2) = 0$ if $V^{\pi_1}_M - V^{\pi_2}_M = x > 0$, thus $\text{Risk}_{\{M_{(x)}\}}(W^*) = 0 + 0 = 0$. □

Corollary 5.2. *Risk*$_{\{M_{(x)}\}}(W_{Ran}) = x_{max}$ *for random guesser* W_{Ran}.

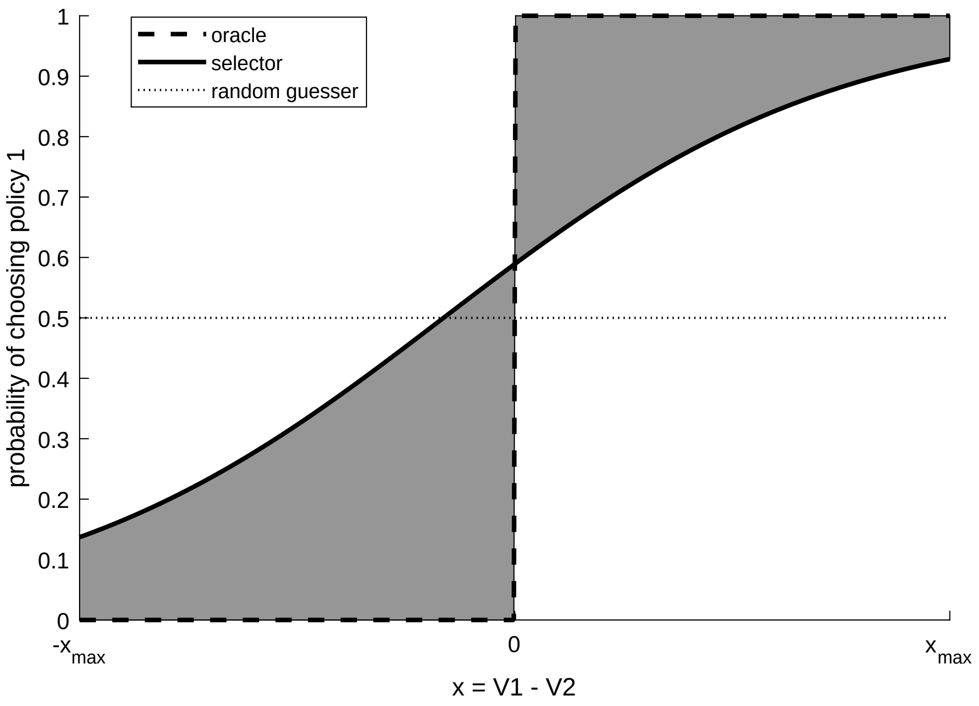

Figure 5.1: An illustration of the family-wise policy selection risk. The area of the grey region equals to the family-wise risk of the selector drawn in solid curve.

Proof. By definition $\mathbb{P}(\pi^{W_{\text{Ran}}} = \pi_1) = \mathbb{P}(\pi^{W_{\text{Ran}}} = \pi_2) = 0.5$, thus $\text{Risk}_{\{M_{(x)}\}}(W_{\text{Ran}}) = 0.5(0 - (-x_{\max})) + 0.5(x_{\max} - 0) = x_{\max}$. □

Corollary 5.3. *$Risk_{\{M_{(x)}\}}(W_{=1}) = Risk_{\{M_{(x)}\}}(W_{=2}) = x_{max}$ for trivial selectors $W_{=1}$ and $W_{=2}$ which always selects π_1 and π_2, respectively.*

Corollary 5.4. *$Risk_{\{M_{(x)}\}}(W_{worst}) = 2x_{max}$ for the worst selector which always gives the wrong answer, i.e. $W_{worst} \overset{def}{=} argmin_{\pi} V^{\pi}$.*

Corollary 5.5. *Family-wise risk is at least 0 and at most $2x_{max}$.*

Thus, we can divide family-wise risk by $x_{\max}$ to obtain the *normalised family-wise risk*, which is between 0 and 2. Selectors $W^*, W_{\text{Ran}}, W_{=1}$, and W_{worst} have normalised family-wise risk 0, 1, 1, and 2, respectively.

Figure 5.1 gives an illustration of the family-wise risk. The x-axis of this figure is the parameter $x = V^{\pi_1} - V^{\pi_2}$ while y-axis the probability of selecting π_1. The oracle selector W^*, the random guesser W_{Ran}, and some general selector W are given by the dashed line, the dotted line, and the s-shape curve, respectively.

The grey region in the figure indicates the difference between the oracle and selector W. In fact, the area of the grey region equals to the family-wise risk of W. Thus, a larger area indicates a larger likelihood of making wrong decisions.

It can be seen clearly from the figure that (1) selector W makes less mistakes than the random guesser in general; (2) W has stronger overall tendency to choose π_1, and thus makes less mistakes when $x > 0$ than when $x < 0$. These two points suggest that a selector does not need to be fair in order to be better in terms of family-wise risk than the random guesser. In fact, moving up/down the curve (and thus changing the overall tendency of favouring a particular policy) does not change the effectiveness as long as it does not go through the inverse-Z-shaped curve of the oracle. This property is formally given by the following lemma.

Lemma 5.6. *Let W, W' be two selectors s.t. $\mathbb{P}(\pi^{W}_{M_{(x)}} = \pi_1) = \mathbb{P}(\pi^{W'}_{M_{(x)}} = \pi_1) + c$ for all $M_{(x)} \in \{M_{(x)}\}$, where c is a constant. Then $\mathit{Risk}_{\{M_{(x)}\}}(W) = \mathit{Risk}_{\{M_{(x)}\}}(W')$.*

Proof. By definition, we have

$$\begin{aligned}
\mathrm{Risk}_{\{M_{(x)}\}}(W) &= \int_{x=-x_{\max}}^{0} \mathbb{P}(\pi^{W}_{M_{(x)}} = \pi_1)\, dx + \int_{x=0}^{x_{\max}} \big(1 - \mathbb{P}(\pi^{W}_{M_{(x)}} = \pi_1)\big)\, dx \\
&= \int_{x=-x_{\max}}^{0} \Big(\mathbb{P}(\pi^{W'}_{M_{(x)}}{=}\pi_1) + c\Big)\, dx + \int_{x=0}^{x_{\max}} \Big(1 - \big(\mathbb{P}(\pi^{W'}_{M_{(x)}}{=}\pi_1) + c\big)\Big)\, dx \\
&= \mathrm{Risk}_{\{M_{(x)}\}}(W') + \int_{x=-x_{\max}}^{0} c\, dx - \int_{x=0}^{x_{\max}} c\, dx \\
&= \mathrm{Risk}_{\{M_{(x)}\}}(W').
\end{aligned}$$

□

Thus, making a selector choose a policy more often by a fixed probability at any x does not change the family-wise risk of that selector.

This is illustrated in Figure 5.2, where two selectors W and W' have the same family-wise risk due to Lemma 5.6. The grey region in this figure indicates the difference between the two selectors. Compared to W', the family-wise risk of W is increased by part of the grey region at $x < 0$, and is decreased by another part of the grey region at $x > 0$. Since the difference of the probability selecting π_1 is the same at any $x \in [-x_{\max}, x_{\max}]$, these

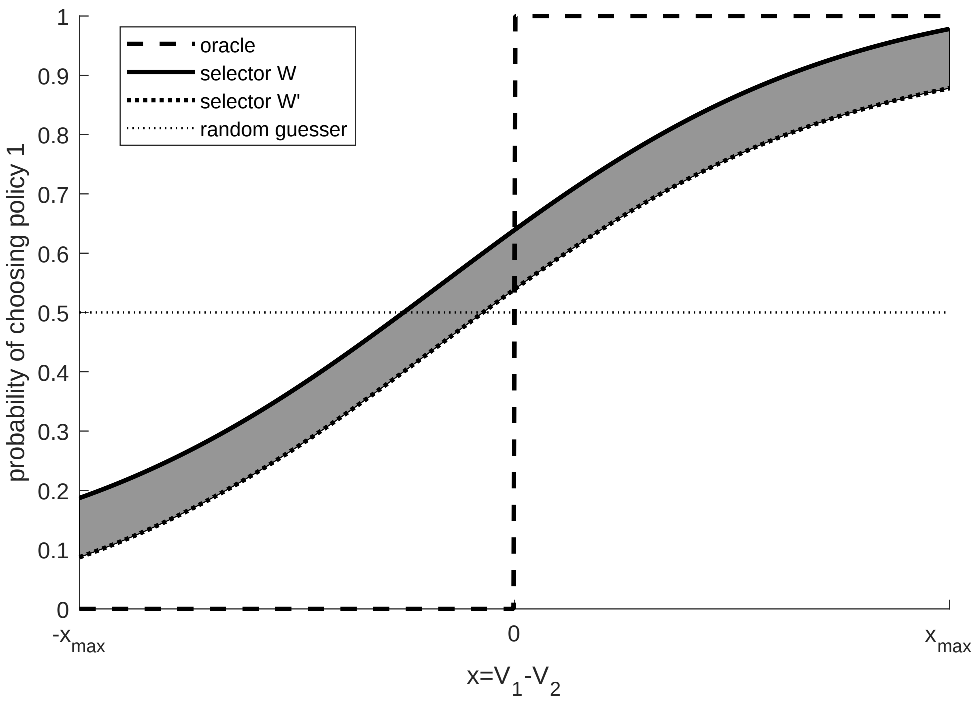

Figure 5.2: An illustration of Lemma 5.6, where W and W' have the same family-wise risk.

two regions have the same area, and thus the two selectors have the same family-wise risk, despite that selector W has stronger overall tendency to select policy π_1 than W'.

For the same reason, the trivial selectors $W_{=1}$ and $W_{=2}$ in Corollary 5.3, which always choose π_1 and π_2 respectively, have the same family-wise risk as the random guesser.

The rationale behind this is that if we assume all MDPs in $\{M_{(x)}\}$ occurs in real-world applications with equal probability, then a selector with less family-wise risk has higher overall probability to make the correct decision. Since under this setting both the trivial selectors $W_{=1}$, $W_{=2}$, and the random guesser W_{Ran} make right and wrong decisions with a 50/50 chance, they should be regarded equally good (or equally bad), which is correctly reflected by the family-wise risk. As for the selectors W and W' in Figure 5.2, although W is worse than W' when $x < 0$, it is also better than W' when $x > 0$, and thus the gain and the loss neutralise each other, making the two selectors equally good in general.

Let us now go back to the example given by [Doroudi et al., 2017], where naive model-based selector W_{MB} is worse than random guesser W_{Ran} at $r < p$, but better at $r > p$. Let $x = V^{\pi_1} - V^{\pi_2} = \frac{r}{1-\gamma} - \frac{p}{1-\gamma} = \frac{r-p}{1-\gamma}$, then we have $r = (1-\gamma)x + p$. Thus, we can create a family of MDPs by varying r, given as follows.

Definition 5.5. Let $x \in [-\theta, \theta]$ where $\theta > 0$. Let $M_{(x)}$ be the MDP given in Example 5.1 but with $r = (1-\gamma)x + p$. In this way we construct a (π_1, π_2)-family $\{M_{(x)}\}$ with parameter $x \in [-\theta, \theta]$.

Lemma 5.7. *The naive model-based selector W_{MB} is as good as the random guesser W_{Ran} with respect to the family-wise policy selection risk on $\{M_{(x)}\}$ given in Definition 5.5 when $\theta \le \frac{p}{1-\gamma}$, and is strictly better than W_{Ran} when $\theta > \frac{p}{1-\gamma}$.*

Proof. By Corollary 5.2.4, the family-wise policy selection risk of random guesser W_{Ran} is $\text{Risk}_{\{M_{(x)}\}}(W_{\text{Ran}}) = \theta$. Now consider the family-wise risk of W_{MB}.

(1) When $\theta \le \frac{p}{1-\gamma}$, since $0 < p < 0.5$, we have $0 \le (1-\gamma)(-\theta)+p \le r \le (1-\gamma)\theta + p \le 2p < 1$. Thus $0 \le \hat{V}^{\pi_1} = V^{\pi_1} = \frac{r}{1-\gamma} \le \frac{2p}{1-\gamma} < \frac{1}{1-\gamma}$. Since $\hat{V}^{\pi_2}$ is 0 with probability $(1-p)$ and is $\frac{1}{1-\gamma}$ with probability p, W_{MB} selects π_1 with probability $(1-p)$ and selects π_2 with probability p, as in Example 5.1. Its family-wise risk is

$$\begin{aligned}\text{Risk}_{\{M_{(x)}\}}(W_{\text{MB}}) &= \int_{x=-\theta}^{0} (1-p)\,dx + \int_{x=0}^{\theta} p\,dx \\ &= (1-p)\theta + p\theta = \theta = \text{Risk}_{\{M_{(x)}\}}(W_{\text{Ran}}).\end{aligned}$$

(2) When $\frac{p}{1-\gamma} < \theta \le \frac{1-p}{1-\gamma}$, we have $(1-\gamma)(-\theta)+p < 0$, and thus when $x \in [-\theta, -\frac{p}{1-\gamma})$, we have $r < 0$. In this case, $\hat{V}^{\pi_2} \ge 0 > \hat{V}^{\pi_1}$ holds whatever the observation of the reward of a_2 is, which means W_{MB} always choose π_2 correctly when $x \in [-\theta, -\frac{p}{1-\gamma})$. Therefore, its family-wise risk becomes

$$\begin{aligned}\text{Risk}_{\{M_{(x)}\}}(W_{\text{MB}}) &= \int_{x=-\theta}^{-\frac{p}{1-\gamma}} 0\,dx + \int_{x=-\frac{p}{1-\gamma}}^{0} (1-p)\,dx + \int_{x=0}^{\theta} p\,dx \\ &= \frac{p(1-p)}{1-\gamma} + p\theta < (1-p)\theta + p\theta = \theta = \text{Risk}_{\{M_{(x)}\}}(W_{\text{Ran}}).\end{aligned}$$

(3) When $\theta > \frac{1-p}{1-\gamma}$, we have $(1-\gamma)\theta + p > 1$, and thus when $x \in (\frac{1-p}{1-\gamma}, \theta]$, we have $r > 1$. In this case, $\hat{V}^{\pi_2} \le 1 < \hat{V}^{\pi_1}$ holds whatever the observation of the reward of

a_2 is, which means W_{MB} always choose π_1 correctly when $x \in (\frac{1-p}{1-\gamma}, \theta]$. Therefore, its family-wise risk becomes

$$\begin{aligned}\mathrm{Risk}_{\{M_{(x)}\}}(W_{\mathrm{MB}}) &= \int_{x=-\theta}^{-\frac{p}{1-\gamma}} 0\,dx + \int_{x=-\frac{p}{1-\gamma}}^{0} (1-p)\,dx + \int_{x=0}^{\frac{1-p}{1-\gamma}} p\,dx + \int_{x=\frac{1-p}{1-\gamma}}^{\theta} 0\,dx \\ &= \frac{p(1-p)}{1-\gamma} + \frac{p(1-p)}{1-\gamma} = \frac{2p(1-p)}{1-\gamma} < 2p\theta < \theta = \mathrm{Risk}_{\{M_{(x)}\}}(W_{\mathrm{Ran}}).\end{aligned}$$

Putting the above together, the lemma is proved. $\square$

By Lemma 5.7, the naive model-based selector is at least as good as the random guesser in terms of family-wise risk, and is strictly better than the random guesser as long as a sufficiently large family of MDPs are covered.

Note that Lemma 5.7 only discusses the case of sample size per state-action $m = 1$. With $m > 1$, it can be derived that the family-wise risk of W_{MB} is

$$\mathrm{Risk}_{\{M_{(x)}\}}(W_{\mathrm{MB}}) = \theta - \frac{1}{1-\gamma}\Big(G\big(p-(1-\gamma)\theta\big) + G\big(p+(1-\gamma)\theta\big) - 2G(p)\Big), \tag{5.4}$$

where $G(y) \overset{\mathrm{def}}{=} \int I_{1-p}(m-y, y+1)dy$, and $I_z(a,b) = \frac{\mathrm{Beta}(z;a,b)}{\mathrm{Beta}(a,b)}$ is the regularised incomplete beta function. It can be shown that under proper setting of θ the function $G(y)$ is convex, and thus $\mathrm{Risk}_{\{M_{(x)}\}}(W_{\mathrm{MB}}) < \theta = \mathrm{Risk}_{\{M_{(x)}\}}(W_{\mathrm{Ran}})$. Also, as m gets larger, $G\big(p-(1-\gamma)\theta\big) + G\big(p+(1-\gamma)\theta\big) - 2G(p)$ becomes larger and thus $\mathrm{Risk}_{\{M_{(x)}\}}(W_{\mathrm{MB}})$ becomes smaller, and eventually converges to 0 as $m \to \infty$. Therefore, the family-wise risk correctly reflects the fact that although W_{MB} is not very good with small sample size m, it is still far better than random guessing with a larger sample size.

That being said, being only marginally better than the random guesser when sample size is small is only borderline acceptable in applications where obtaining data is expensive. Therefore, we still need to develop selectors that can work better than naive W_{MB} under small sample size, which will be elaborated in later sections.

5.2.5 Family-wise unfairness

As elaborated in the last section, a selector does not need to be fair in order to be better in general than the random guesser. Lemma 5.6 shows that a selector strongly favours one policy can be as good as a more balanced selector in terms of family-wise bias. The rationale behind this is that although a selector can be disadvantaged in some MDPs for being "biased" (in the general sense) toward some particular policies, such a bias can be beneficial in some other MDPs, and thus the additional gain and loss caused by the bias is neutralised as long as the bias is universal (i.e. has the same amount of bias in all MDPs).

However, there are also cases in practice where being relatively "fair" (in general sense) can be helpful. An example to such cases is given as follows.

Example 5.3. Consider two MDPs M_1, M_2 and two policies π_1 and π_2, where π_1 is better in M_1 and is worse in M_2 than π_2. Selector W_1 selects π_1/π_2 with probability 0.9/0.1 in M_1 and 0.7/0.3 in M_2, while selector W_2 selects π_1/π_2 with probability 0.6/0.4 in M_1 and 0.4/0.6 in M_2. The overall risk of W_1 is $0.1 + 0.7 = 0.8$, and for W_2 it is $0.4 + 0.4 = 0.8$, thus the two selectors have the same overall risk.

Now, suppose that we need to run the learning algorithms until they output the correct policy in each MDP once. Then the average number of runs W_1 needs is $1/0.9 + 1/0.3 \approx 4.44$, while for W_2 it is $1/0.6 + 1/0.6 \approx 3.33$, and therefore selector W_2 needs less runs on average than W_1.

Note that in this example W_1 is strongly biased towards π_1, while W_2 does not. This implies that the property of not being biased toward a particular policy can still have practical meaning in some scenarios.

Since the fairness proposed by [Doroudi et al., 2017] is not suitable for comparing selectors, we extend their idea and combine it with our family-wise formulation as follows.

Definition 5.6. The family-wise unfairness of policy selector W with respect to $\{M_{(x)}\}$

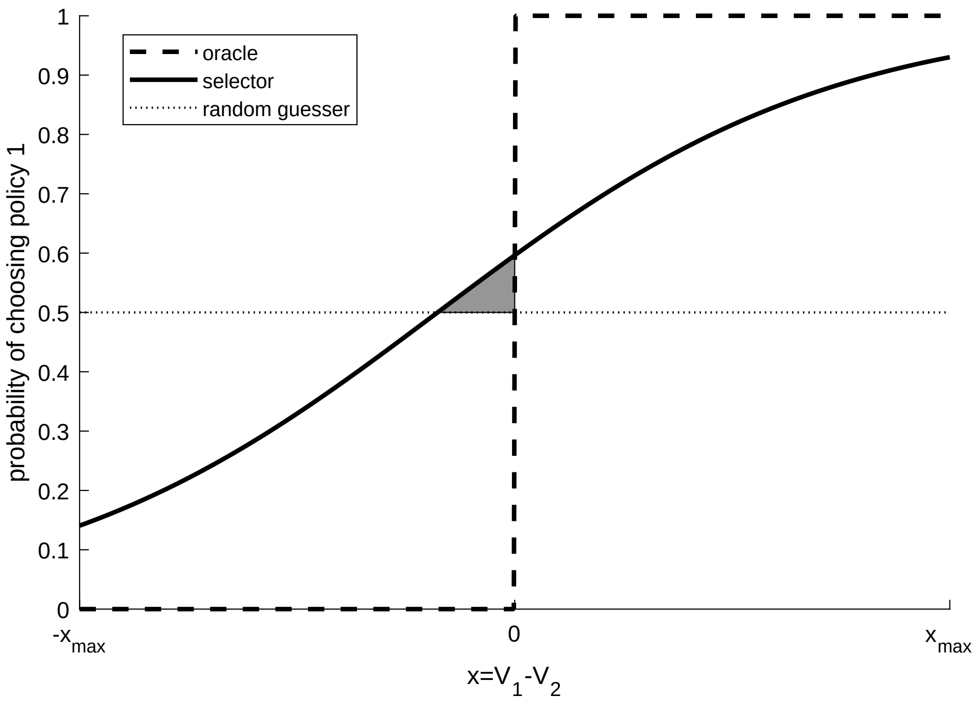

Figure 5.3: An illustration of family-wise unfairness. The area of the grey region equals to the family-wise unfairness of the selector drawn in solid curve.

over policies $\{\pi_1, \pi_2\}$ is defined as

$$\begin{aligned}\text{Unfairness}_{\{M_{(x)}\}}(W) \overset{\text{def}}{=} &\int_{x=-x_{\max}}^{0} \max\left\{0,\, \mathbb{P}(\pi^W_{M_{(x)}} = \pi_1) - 0.5\right\} dx \\ &+ \int_{x=0}^{x_{\max}} \max\left\{0,\, \mathbb{P}(\pi^W_{M_{(x)}} = \pi_2) - 0.5\right\} dx.\end{aligned}$$

In the integrals above, only the the extra parts of the regions where W chooses the wrong policies with more than 0.5 probability is considered. An illustration to this interpretation is given in Figure 5.3, where the area of the grey region corresponds to the family-wise unfairness of the selector.

Corollary 5.8. *The family-wise unfairness of a strictly fair selector is 0.*

Corollary 5.9. *The family-wise unfairness of random guesser W_{Ran} is 0.*

Corollary 5.10. *The family-wise unfairness of trivial selectors $W_{=1}$ and $W_{=2}$ which always select π_1 and π_2 respectively is $0.5x_{max}$.*

Corollary 5.11. *The family-wise unfairness of the worst selector W_{worst} which always selects the inferior policy is x_{max}.*

Corollary 5.12. *In the MDP family given by Definition 5.5 with $x_{max} = \theta \leq \frac{p}{1-\gamma}$, the naive model-based selector W_{MB} has $Unfairness_{\{M_{(x)}\}}(W_{MB}) = (0.5 - p)x_{max}$.*

Thus, the family-wise unfairness of naive model-based selector lies between a strictly fair selector and a trivial selector that always choose one specific policy. These corollaries give examples to how our family-wise unfairness describes the overall soft fairness property of selectors quantitatively, which is more informative than the strict fairness given by [Doroudi et al., 2017].

5.3 Bootstrap-based Policy Voting

5.3.1 The method

As discussed in Section 5.2.4 (see Equation 5.4), the naive model-based policy selector W_{MB} is better than the random selector W_{Ran} in terms of family-wise policy selection risk when sample size is sufficiently large, but with a small sample it becomes only marginally better than W_{Ran} in the family of MDPs given in Definition 5.5. Section 5.2.5 also shows that W_{MB} has non-zero family-wise unfairness.

Note that the value functions in the MDPs of Definition 5.5 do not suffer from the transformation bias, because both $V^{\pi_1} = \frac{r}{1-\gamma}$ and $V^{\pi_1} = \frac{p}{1-\gamma}$ are linear functions with respect to the unknown random variables. This implies that correcting the transformation bias by methods such as our BTBC proposed in Chapter 4 is not sufficient for achieving good performance in policy selection.

These observations inspire us to design policy selection methods that can work better than the naive model-based selector, especially when sample size is small. Although there is nothing much can be done with sample size per state-action m as small as 1, when it becomes around 10 there should be a much larger room for improvement.

To this end, we propose Bootstrap-based Policy Voting (BPV) method for policy selection. The algorithm can be seen as an adaption of our single-depth BTBC, originally designed for value estimation bias correction, to policy selection. Specifically, given the collection of the transition counts $\mathbf{N} \overset{\text{def}}{=} \{N_{s,a,s'}\}$ and the sample size at each state-action pair $\mathbf{N}_{*,*} \overset{\text{def}}{=} \{N_{s,a}\}$, the following steps are taken:

1. Estimate transition probabilities $\hat{P}$ from data $\mathbf{N}$ by $\hat{P}(s'|s,a) \overset{\text{def}}{=} N_{s,a,s'}/N_{s,a}$.
2. Generate k bootstrap sample $\mathbf{N}_1^+, \mathbf{N}_2^+, ..., \mathbf{N}_k^+$ from $\hat{P}$, where each $\mathbf{N}_i^+$ is a collection of transition counts such that $\mathbf{N}_i^+ \sim \text{Multinomial}(\mathbf{N}_{*,*}; \hat{P})$.[1]
3. For each bootstrap sample $\mathbf{N}_i^+$, compute $\hat{V}^{\pi_1}(\mathbf{N}_i^+)$ and $\hat{V}^{\pi_2}(\mathbf{N}_i^+)$ for candidate policies π_1 and π_2.
4. For each i, let $y_i \overset{\text{def}}{=} \big(2\hat{V}^{\pi_1}(\mathbf{N}) - \hat{V}^{\pi_1}(\mathbf{N}_i^+)\big) - \big(2\hat{V}^{\pi_2}(\mathbf{N}) - \hat{V}^{\pi_2}(\mathbf{N}_i^+)\big)$. The i-th bootstrap votes for π_1 if $y_i > 0$, votes for π_2 if $y_i < 0$, and gives both π_1 and π_2 0.5 vote if $y_i = 0$.
5. Count the votes and select the policy in $\{\pi_1, \pi_2\}$ that has the most votes.

The first three steps of BPV is the same as single-depth BTBC, where k bootstrap samples of transition counts are generated from the estimated transition probabilities and then are used to compute bootstrap state/action values of interest. The last two steps are very different from BTBC. In the original single-depth BTBC, the bootstrap values $\hat{V}(\mathbf{N}_i^+)$ are averaged and used in the form of $\big(2\hat{V}(\mathbf{N}) - \text{avg}[\hat{V}(\mathbf{N}_i^+)]\big)$ to compute the bias-corrected value estimates. In BPV, on the other hand, the bootstrap values $\hat{V}(\mathbf{N}_i^+)$ are used to compute $\big(2\hat{V}^{\pi}(\mathbf{N}) - \hat{V}^{\pi}(\mathbf{N}_i^+)\big)$ separately for each bootstrap without taking average, and the computed values are then used to decide the vote from that bootstrap. The results from these bootstraps are put together in the last step by choosing the policy that has received the most votes.

Note that BPV does not directly compare the values $\hat{V}(\mathbf{N}_i^+)$ computed from the bootstrap samples, but use them to shift the original $\hat{V}(\mathbf{N})$ by $\big(\hat{V}(\mathbf{N}) - \hat{V}(\mathbf{N}_i^+)\big)$ and compares these shifted value estimates, similar to what is done in BTBC. The rationale behind this

[1]See Section 4.4.1 for more detailed description regarding the generation of bootstrap samples.

design is the same as BTBC, which has already been elaborated in Section 4.4.2, and thus will not be repeated here.

The pseudo-code of BPV is given in Algorithm 5.

Algorithm 5 Bootstrap-based Policy Voting ($\mathbf{N}$, $\{\pi_1, \pi_2\}$)

```
Input: transition counts $\mathbf{N}$, candidate policies $\{\pi_1, \pi_2\}$
Parameter: number of bootstrap samples $b$
Output: one of the candidate policies
 1: Compute transition probabilities $\hat{P}_{\mathbf{N}}$ using $\mathbf{N}$
 2: $\hat{V}_1 = \text{BellmanSolution}(\hat{P}_{\mathbf{N}}, \pi_1)$
 3: $\hat{V}_2 = \text{BellmanSolution}(\hat{P}_{\mathbf{N}}, \pi_2)$
 4: $\text{vote}_1 = \text{vote}_2 = 0$
 5: for $j = 1$ to $b$ do
 6:     Generate bootstrap data $\mathbf{N}^+$ using $\hat{P}_{\mathbf{N}}$, i.e. $N^+_{s,a,*} \sim \text{Multinomial}(N_{s,a}; \hat{P}_{\mathbf{N}}(*|s,a))$
 7:     Compute $\hat{P}_{\mathbf{N}^+}$ using $\mathbf{N}^+$
 8:     $\hat{V}_1^+ = \text{BellmanSolution}(\hat{P}_{\mathbf{N}^+}, \pi_1)$
 9:     $\hat{V}_2^+ = \text{BellmanSolution}(\hat{P}_{\mathbf{N}^+}, \pi_2)$
10:     if $2\hat{V}_1 - \hat{V}_1^+ > 2\hat{V}_2 - \hat{V}_2^+$ then
11:         $\text{vote}_1 = \text{vote}_1 + 1$
12:     else if $2\hat{V}_1 - \hat{V}_1^+ < 2\hat{V}_2 - \hat{V}_2^+$ then
13:         $\text{vote}_2 = \text{vote}_2 + 1$
14:     else
15:         $\text{vote}_1 = \text{vote}_1 + 0.5$
16:         $\text{vote}_2 = \text{vote}_2 + 0.5$
17:     end if
18: end for
19: Return $\pi_i \in \{\pi_1, \pi_2\}$ s.t. $i = \text{argmax}_{j \in \{1,2\}} \text{vote}_j$
```

5.3.2 Connection with BTBC selector

We can also apply BTBC to policy selection by replacing the uncorrected value estimates in naive model-based selector with the bias-corrected ones:

$$W_{\text{BC}}(\{\pi_1, \pi_2\}) \overset{\text{def}}{=} \begin{cases} \pi_1 & \tilde{V}^{\pi_1} > \tilde{V}^{\pi_2} \\ \pi_2 & \tilde{V}^{\pi_1} < \tilde{V}^{\pi_2} \\ \text{Uniform}\{\pi_1, \pi_2\} & \tilde{V}^{\pi_1} = \tilde{V}^{\pi_2}, \end{cases} \tag{5.5}$$

where $\tilde{V}^\pi = 2\hat{V}^\pi(\mathbf{N}) - \text{avg}[\hat{V}^\pi(\mathbf{N}_i^+)]$ is the bias-corrected value estimate of single-depth BTBC. As pointed out in the last section, the main difference between the BTBC selector W_{BC} and the BPV selector W_{PV} is that W_{BC} takes average over $\big(2\hat{V}^\pi(\mathbf{N}) - \hat{V}^\pi(\mathbf{N}_i^+)\big)$ first, then compare the results to decide a single vote, while BPV compare $\big(2\hat{V}^\pi(\mathbf{N}) - \hat{V}^\pi(\mathbf{N}_i^+)\big)$ first to get b votes, then adds up the votes to get the voting result.

Let $\text{sign}(x)$ be the sign function which equals to 1, 0, and -1 for $x > 0$, $x = 0$, and $x < 0$, respectively. The operation of comparing two values x_1 and x_2 can be represented by $\text{sign}(x_1 - x_2)$. Let $y_i \overset{\text{def}}{=} \big(2\hat{V}^{\pi_1}(\mathbf{N}) - \hat{V}^{\pi_1}(\mathbf{N}_i^+)\big) - \big(2\hat{V}^{\pi_2}(\mathbf{N}) - \hat{V}^{\pi_2}(\mathbf{N}_i^+)\big)$, then the BTBC selector is mathematically equivalent to

$$f_{\text{BC}}(\{y_i\}) = \text{sign}(\text{avg}(y_i)),$$

while the BPV selector is equivalent to

$$f_{\text{PV}}(\{y_i\}) = \text{sign}(\text{avg}(\text{sign}(y_i))).$$

The selectors output π_1 if the above f value is 1, and output π_2 if f is -1.

It is clear that the two selectors are different, but which one is better? Our observation is that they can be similarly good in terms of family-wise policy selection risk in general, but BPV selector tends to be better in terms of family-wise fairness, thanks to the additional sign operation before taking average. To explain this, consider the case where $\{y_i\} = \{3, 3, 2, -1, -8\}$. Since there are three positives and two negatives, $f_{\text{PV}}(\{y_i\}) = \text{sign}(\text{avg}(\text{sign}(y_i))) = \text{sign}((3-2)/5) = 1$, while $f_{\text{BC}}(\{y_i\}) = \text{sign}(\text{avg}(y_i)) = \text{sign}((3+3+2-1-8)/5) = -1$. As can be seen from the process of computation, the final vote by the BTBC selector is heavily weighted by the "outlier" -8, while for BPV the -1 and the -8 have the same weight in the final result.

Due to such property, BTBC selector is more likely to select policies whose estimated value distributions have heavy right tails (i.e. the ones that have small but non-negligible probability to yield very large returns), while BPV is less affected by heavy tails and thus is more likely to have better family-wise fairness than BTBC.

5.4 Utilising the improved policy selection

The discussion on policy selection in previous sections is mainly for pairwise policy selection where the number of candidate policies is two. In this section, we first propose a simple tournament method to conduct policy selection with more than two candidate policies, then propose a bootstrap policy generation method for generating candidate policies from a learnt model, and finally combine them with the improved policy selection methods proposed in Section 5.3 to further improve overall model-based RL performance.

5.4.1 The tournament method for multi-policy policy selection

The policy selection methods proposed in the last section are for pairwise policy selection. When there are more than two candidate policies, we can conduct a single-elimination tournament of pairwise policy selection to choose the best policy.

Specifically, let $\pi_1, \pi_2, ..., \pi_n$ be n candidate policies, arranged in a random order. In the first round, we conduct pairwise policy selection for $\{\pi_1, \pi_2\}$, $\{\pi_3, \pi_4\}$, and so on, and if the last policy π_n does not have a competitor then it automatically wins. The winners (the policies been selected) of the first round then goes to the next round, and the process is repeated until there is only one policy left. This champion policy is then chosen as the output of the tournament process. In this way, we can choose one policy from n candidates by conducting $(n-1)$ pairwise policy selection.

Due to the randomness occurred in generating bootstrap samples in our bootstrap-based selectors, the policy selection result does not have transitivity, i.e. $W\{\pi_1, \pi_2\} = \pi_1$ and $W\{\pi_2, \pi_3\} = \pi_2$ in one tournament does not imply $W\{\pi_1, \pi_3\} = \pi_1$, and even the same $W\{\pi_1, \pi_2\}$ does not necessarily give the same result (unless there is a sufficiently large gap in their values).

However, since we are only interested in the average utility of the final winner and do not care about the complete ranking, the single-elimination tournament is sufficient for our purpose. The random allocation of the initial positions in the tournament for the policies makes no policy be favoured over others in probability, and therefore the best policy with respect to the order decided by the selector has the most probability to win the tournament.

That being said, single-elimination tournament is not necessarily the tournament method that has the highest probability of selecting the best policy. Appleton [1995] conducted simulations on different tournament methods under different settings, and found that both seeded draw-and-process tournament and round-robin-played-twice tournament are better than single-elimination in terms of the probability of selecting the best competitor. The application of these more sophisticated tournament methods to multi-policy policy selection is left for future work.

5.4.2 Bootstrap policy generation

Most RL algorithms only output one policy, and many model-based RL algorithms (e.g. the ones that use Value Iteration algorithm for policy planning) do not explicitly conduct policy selection. However, we can still use the bootstrap method to generate multiple candidate policies, and conduct tournaments to find policies that can possibly better than the original output policy.

Let $\mathbf{N} \overset{\text{def}}{=} \{N_{s,a,s'}\}$ be the transition counts of the learning process and $\mathbf{N}_{*,*} \overset{\text{def}}{=} \{N_{s,a}\}$ be the corresponding sample size at each state-action pair. The *bootstrap policy generation process* consists of the following steps:

1. Estimate transition probabilities $\hat{P}$ from data $\mathbf{N}$ by $\hat{P}(s'|s,a) \overset{\text{def}}{=} N_{s,a,s'}/N_{s,a}$.
2. Generate k bootstrap sample $\mathbf{N}_1^+, \mathbf{N}_2^+, ..., \mathbf{N}_k^+$ from $\hat{P}$, where each $\mathbf{N}_i^+$ is a collection of transition counts such that $\mathbf{N}_i^+ \sim$ multinomial$(\mathbf{N}_{*,*}; \hat{P})$.
3. Use a policy planning algorithm (e.g. Value Iteration) to compute the optimal policy $\hat{\pi}_i^+$ with respect to each bootstrap model $\hat{M}_i^+ \overset{\text{def}}{=} (\mathcal{S}, \mathcal{A}, \hat{P}_i^+, \hat{R}, \gamma)$, where $\hat{P}_i^+$ is computed from $\mathbf{N}_i^+$ in the same way as $\hat{P}$.

In this way, we can easily generate k candidate policies for policy selection tournament. Due to the inherent similarity between the bootstrap samples $\mathbf{N}_i^+$ and the original data $\mathbf{N}$, the quality of the generated policies $\hat{\pi}_i^+$ is close to the original output policy π in most cases, and thus they are good competitors to π in the tournament.

5.4.3 Tournament-based policy refinement

By combining the single-elimination tournament method for multi-policy policy selection with the bootstrap policy generation method, we obtain a general policy refinement process for model-based RL, which can be used regardless of whether the original learning algorithm explicitly conducts policy selection or not.

Algorithm 6 Tournament-based Policy Refinement

```
Input: transition counts $\mathbf{N}$
Parameter: tournament size $k$
Output: a refined policy
 1: Compute transition probabilities $\hat{P}_{\mathbf{N}}$ using $\mathbf{N}$
 2: (Original output policy) $\pi_1 = \text{PolicyPlanning}(\hat{P}_{\mathbf{N}})$
 3: for $i = 2$ to $k$ do
 4:     Generate bootstrap data $\mathbf{N}^+$ using $\hat{P}_{\mathbf{N}}$, i.e. $N^+_{s,a,*} \sim \text{Multinomial}(N_{s,a}; \hat{P}_{\mathbf{N}}(*|s,a))$
 5:     Compute $\hat{P}_{\mathbf{N}^+}$ using $\mathbf{N}^+$
 6:     $\pi_i = \text{PolicyPlanning}(\hat{P}_{\mathbf{N}^+})$
 7: end for
 8: Number of remaining competitors $n = k$
 9: while $n > 1$ do
10:     for $i = 1$ to $\lfloor n/2 \rfloor$ do
11:         $\pi_i = \text{PolicySelection}(\pi_{2i-1}, \pi_{2i})$
12:     end for
13:     if $n \bmod 2 == 1$ then
14:         $\pi_{\lceil n/2 \rceil} = \pi_n$
15:     end if
16:     $n = \lceil n/2 \rceil$
17: end while
18: Return $\pi_1$
```

The pseudocode of this combined process is given in Algorithm 6. With tournament size k, Algorithm 6 generates $(k-1)$ policies by bootstrap at Lines 3-7, then let them and the original policy compete in a single-elimination tournament at Lines 8-17. The winners of the policy selection at each round override the policies at first $\lceil n/2 \rceil$ positions of the array of policies $\{\pi_i\}$, and thus the output policy at Line 18 does not necessarily equals to the original output policy computed at Line 2.

The policy selector used in the algorithm is represented with PolicySelection(π, π'). By using the BTBC selector given in Section 5.3.2 we get the *Bias-corrected Tournament* (BCT) method, while by using the BPV selector given in Section 5.3.1 we get the *Policy Voting Tournament* (PVT) method.

Note that the bootstrap samples generated at Line 4 are *not* used in bootstrap-based policy selection. The selectors have to create their own bootstrap samples in order to make the tournament meaningful. Also, although in Section 5.4.1 we have mentioned that the random allocation of the initial positions in the tournament makes it "fair" (in general sense) to all candidate policies, the random allocation is not necessary in Algorithm 6 because $(k-1)$ candidates are generated by independent random processes.

Although we can also apply the naive model-based selector to Algorithm 6, it is practically meaningless because it simply chooses policies with maximum estimated utility, which is exactly the original output policy, unless there are some generated policies that happen to have the same estimated utility as the original output policy.

Algorithm 6 does not guarantee to output a policy different from the original output, nor does it guarantee to output a policy that has higher true state/action value than the original output. There can be cases where the policy generation process fails to generate a better policy, and cases where the policy selector fails to pick the better one from the candidates. However, as can be seen from our experiment results presented in the next section, with a proper policy selector such as BTBC-based W_{BC} and BPV-based W_{PV}, Algorithm 6 can utilise the information from the collected data more efficiently and achieve a better overall performance than naive model-based learning, even if the sample size is relatively small.

5.5 Experiments

5.5.1 Policy selection

This section presents our empirical results for the policy selectors including the naive model-based W_{MB}, the transformation bias corrected W_{BC}, and Bootstrap-based Policy Voting W_{PV}. The transformation-bias-corrected W_{BC} used the single-depth BTBC method, so that W_{BC} and W_{PV} create the same number of bootstrap samples under the same setting of bootstrap size parameter b.

The double chain MDP family

To evaluate the family-wise metrics proposed in Section 5.2, we construct an MDP family based on the chain MDPs as follows.

Definition 5.7. Let $n \geq 3$ be the chain length and $\gamma \in (0, 1)$ be the discount factor. A double chain with length n, forward probability p, and parameter x is defined as follows.

1. The states are $s_1, s_2, ..., s_n$. The agent starts interaction from the start state s_1. State s_n is called the goal, and state $s_{\lfloor n/2 \rfloor}$ is called the mid-point.
2. At states s_i with $i \neq n$ and $i \neq \lfloor n/2 \rfloor$, the agent has one action a_1. Taking this action transits the agent to s_{i+1} with probability p, and to s_i with probability $(1-p)$.
3. At the goal s_n, there is one action a_1 which transits back to s_1 with probability p and yields reward $r_1 = 1$, and transits to s_n with probability $(1-p)$ with no reward.
4. At the midpoint $s_{\lfloor n/2 \rfloor}$, there are two actions a_1 and a_2. Action a_1 transits the agent to $s_{\lfloor n/2 \rfloor + 1}$ with probability p, and to $s_{\lfloor n/2 \rfloor}$ with probability $(1-p)$. Action a_2 transits the agent back to s_1 with probability p and yields reward

$$r_2 \overset{\text{def}}{=} \frac{F(n) - x}{F(\lfloor n/2 \rfloor)},$$

where

$$F(k) \overset{\text{def}}{=} \frac{1}{\gamma} \cdot \frac{(\gamma p)^k}{(1 - \gamma(1-p))^k - (\gamma p)^k}$$

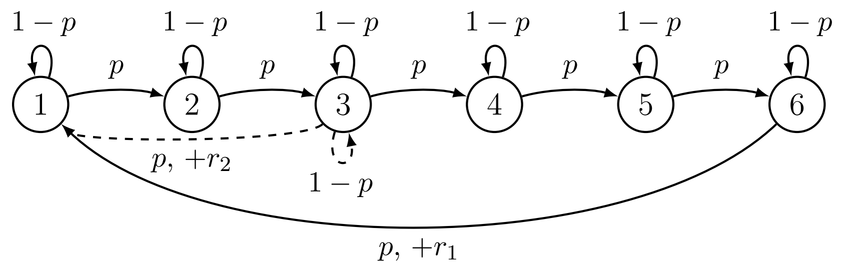

Figure 5.4: Double chain MDP with $n = 6$ states. Solid arrows are transitions by taking action a_1, while dashed arrows are the ones by taking a_2.

and x is the parameter of the double chain, and transits the agent to $s_{\lfloor n/2 \rfloor}$ with probability $(1 - p)$ without reward.

Lemma 5.13. *Let π_1 be a policy which selects a_1 at all states. Let π_2 be a policy which selects a_2 at state $s_{\lfloor n/2 \rfloor}$, and selects a_1 at other states. Let state value $V^{\pi}(s_1)$ be the utility of interest. Then the double chain MDPs given in Definition 5.7 forms a (π_1, π_2)-family of MDPs with parameter x.*

Proof. By the Bellman equation it can be shown that $V^{\pi_1}(s_1) = F(n)$ and $V^{\pi_2}(s_1) = r_2 F(\lfloor n/2 \rfloor)$. Since $r_2 \overset{\text{def}}{=} \frac{F(n)-x}{F(\lfloor n/2 \rfloor)}$, we have $x = V^{\pi_1}(s_1) - V^{\pi_2}(s_1)$ and thus the lemma is proved. □

Figure 5.4 gives an illustration of a double chain MDP with length $n = 6$.

The computation of family-wise risk and family-wise unfairness involves integrals with respect to the parameter x. In the experiments, we used middle Riemann sum[1] for approximation.

We conducted experiments under the following setting. Unless stated otherwise, the number of states n, the discount factor γ, and the forward probability p of the double chain MDP were set to $n = 50$, $\gamma = 0.97$, and $p = 0.7$. To test the effectiveness under a small sample size, we set the sample size per state-action $m = 10$. As for the range of the parameter x of MDP families, we set $x \in [-0.3V^{\pi_1}, 0.3V^{\pi_1}]$, where V^{π_1} is the true state value of π_1 at s_1.

[1]The sum of rectangles with height being the function value of the middle point of each small intervals.

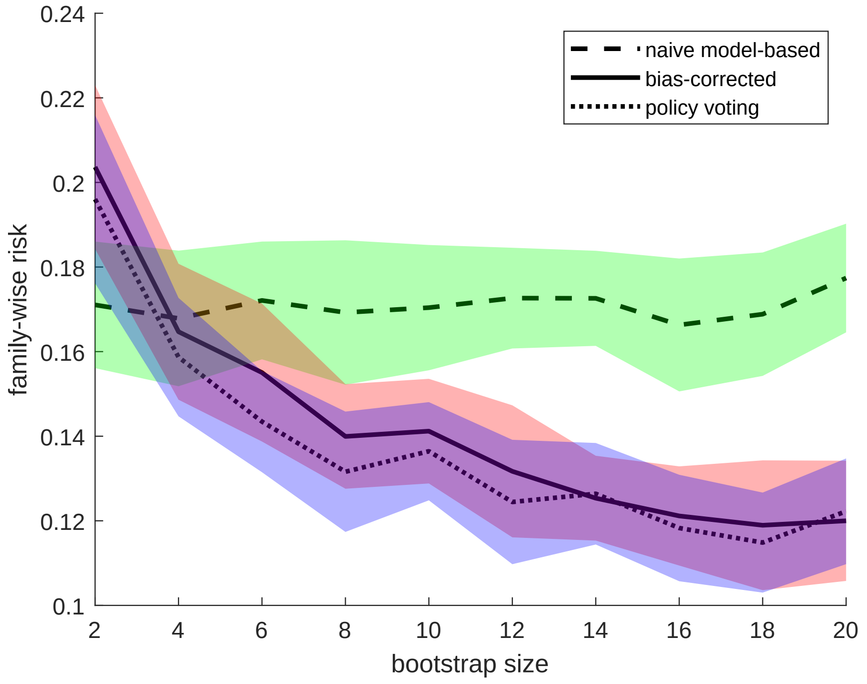

Figure 5.5: Normalised family-wise policy selection risk vs bootstrap size.

Effect of bootstrap size

In the first experiment, we changed the bootstrap size b of single-depth BTBC selector W_{BC} and BPV selector W_{PV} from 2 to 20 in order to see their family-wise risk and family-wise unfairness metrics under different bootstrap size. In each run, a dataset with 10 observations of transitions for each state-action pair was obtained, and sent to selectors W_{MB}, W_{BC}, and W_{PV} for them to select one from $\{\pi_1, \pi_2\}$. Each setting of b was tested for 1000 runs to get the empirical distributions of the selection results.

The results of the experiment are presented in Figures 5.5 and 5.6. Both of the family-wise metrics were normalised by the half-length of the domain of x (i.e. $0.3\hat{V}^{\pi_1}$) so that their values can be roughly interpreted as "average probability" (though it is more appropriate to compare their ratio rather than the absolute values). The shaded areas represent the intervals within one standard deviation (green - W_{MB}; red - W_{BC}; blue - W_{PV}).

As can be seen from Figure 5.5, in terms of family-wise risk, our selectors W_{BC} and W_{PV} were worse than the naive model-based selector W_{MB} at bootstrap size $b = 2$, but

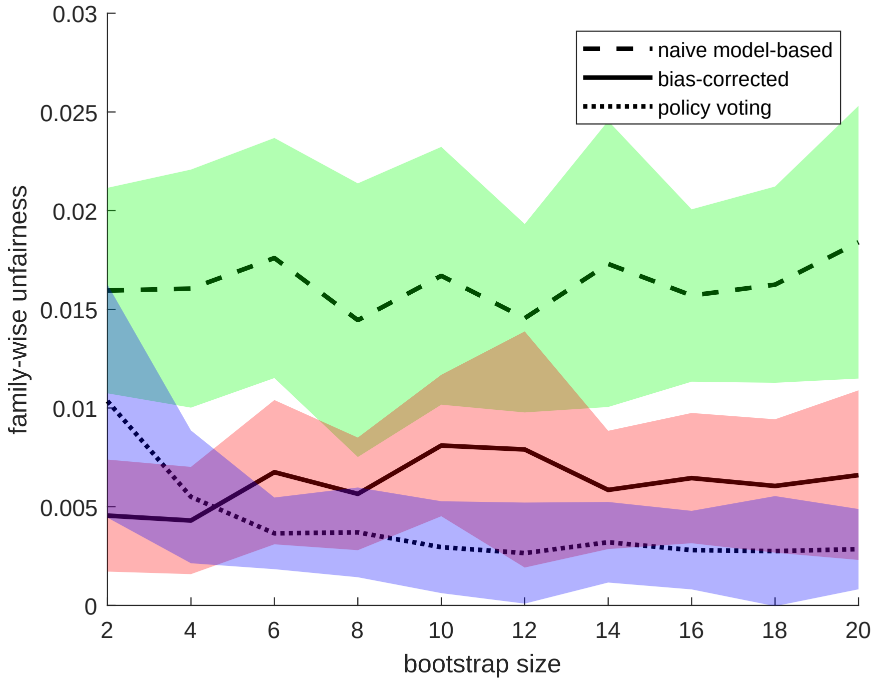

Figure 5.6: Normalised family-wise unfairness vs bootstrap size.

became much better with a larger b. Since family-wise risk indicates the overall possibility of choosing wrong policies in different MDPs, these results indicate that our selectors were remarkably better than the naive model-based selector in correctly selecting the better candidate policy.

Specifically, at $b = 20$, the normalised family-wise risk was 0.120 for W_{BC}, 0.122 for W_{PV}, while for naive W_{MB} it was about 0.171 for all b. Thus, W_{BC} and W_{PV} made about $(0.171 - 0.120)/0.171 \approx 29.8\%$ and $(0.171 - 0.122)/0.171 \approx 28.7\%$ less mistakes in policy selection than W_{MB}, respectively. Considering that the sample size per state-action m was only 10, the above results clearly shows that our selectors could make better use of the information hidden inside a small sample.

It is interesting to see that the bootstrap size parameter b of our BTBC selector had remarkable effect in policy selection, in contrast to our previous results in transformation bias correction presented in Chapter 4, where changing b did not lead to a significant change to the result (see Figure 4.9). One explanation to this is, by increasing the bootstrap size, the variance of the bias-corrected value estimates were reduced, which was

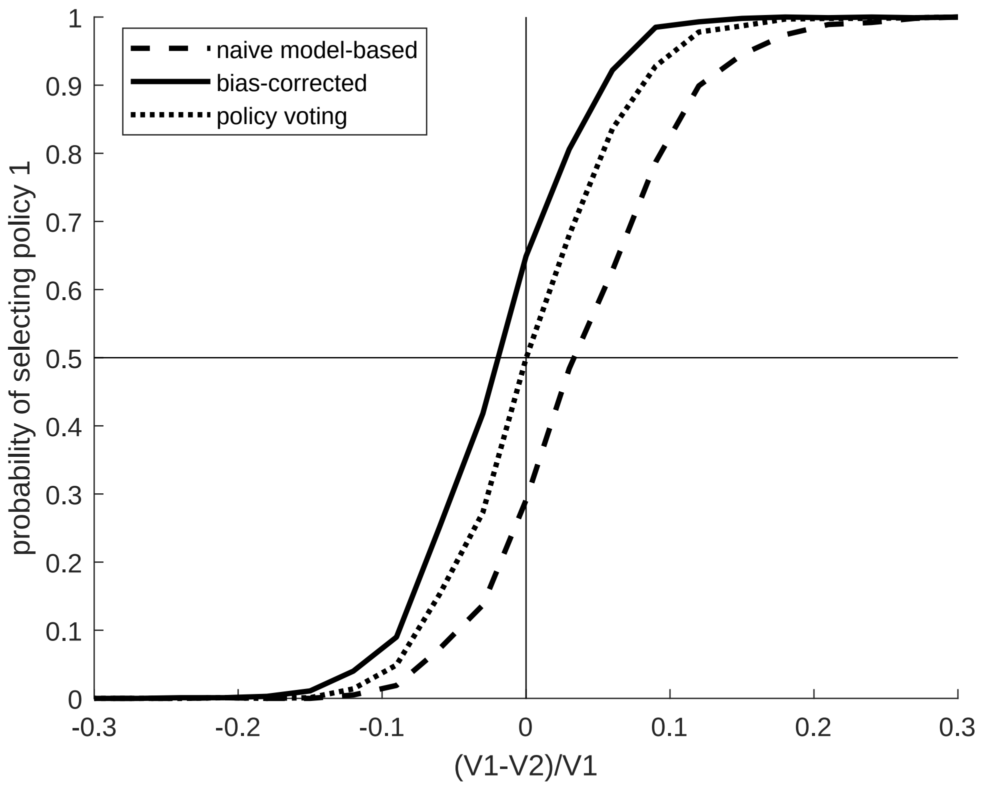

Figure 5.7: Probability of selecting π_1 under different $\frac{x}{V^{\pi_1}} = \frac{V^{\pi_1} - V^{\pi_2}}{V^{\pi_1}}$.

helpful for policy selection but did not have direct impact on the accuracy of value estimation.

The result of the family-wise unfairness shown in Figure 5.6 is also very interesting. As can be seen from the figure, both of our selectors had less family-wise unfairness than the naive selector, and with $b \geq 4$, the BPV selector had less family-wise unfairness than the BTBC selector, which is in accordance with our analysis in Section 5.3.2. More importantly, the BTBC selector showed an increasing trend of being unfair under a larger b. A possible reason for this is, as b increased and the variance of the corrected value decreased, the heavy tail of the distribution became more apparent and thus were more likely to drive the BTBC selector to select the corresponding policy.

To have a clearer picture of the family-wise metrics, we draw the curves of the probability of choosing π_1 under different MDP family parameter $x = V^{\pi_1} - V^{\pi_2}$, in the same manner as in Figure 5.1. The curves are shown in Figure 5.7.

Clearly, the BTBC selector showed a strong tendency of selecting policy π_1, which is

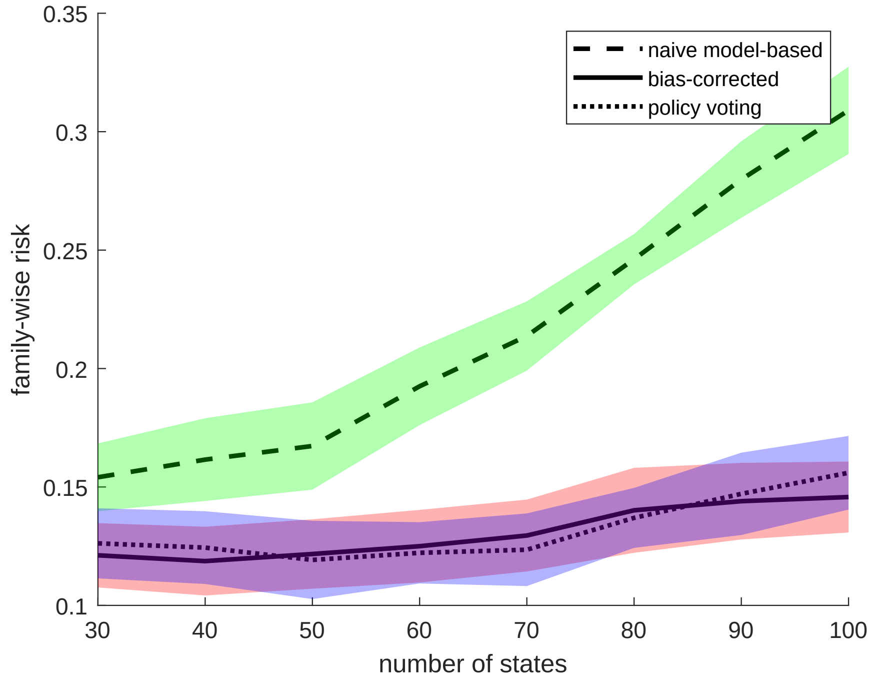

Figure 5.8: Normalised family-wise policy selection risk vs number of states.

the one that collects rewards from the goal of the chain, while naive model-based selector showed a strong tendency of selecting π_2, which is the one that collect rewards from the mid-point. Since π_1 involves more transitions than π_2 and thus has a heavier right tail in its value estimate distribution, these results are in accordance with our analysis that the BTBC selector favours policies with heavy right tails. The results also indicate that the naive model-based selector had the opposite tendency and disliked such policies.

Effect of number of states

As discussed in Section 4.3.3, the transformation bias of model-based RL increases with the number of states under the same sample size per state-action pair. We are interested in whether this also happens in terms of family-wise metrics of policy selection, and thus conducted experiment under different number of states.

Specifically, we fixed the bootstrap size b of BTBC and BPV selectors to 20, and changed the number of states n from 30 to 100. The sample size per state-action m was still set to 10. Each selector under each n were tested 1000 runs.

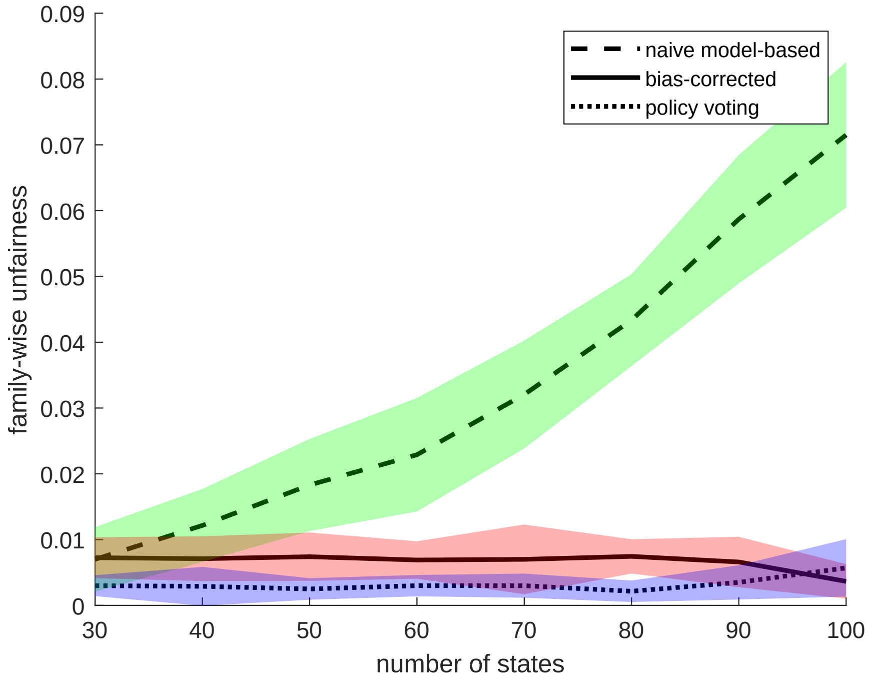

Figure 5.9: Normalised family-wise unfairness vs number of states.

The empirical normalised family-wise risk and family-wise unfairness under different number of states n are shown in Figures 5.8 and 5.9. Figure 5.8 shows that the family-wise risk of the naive model-based selector increased quickly with the number of states (from 0.154 at $n = 30$ to 0.309 at $n = 100$), while for our bias-corrected and policy voting selectors it increased much slower (from 0.121 to 0.146 and from 0.126 to 0.156). In Figure 5.9, the family-wise unfairness of the naive model-based selector also increased remarkably from 0.0070 to 0.0715, while for our W_{BC} and W_{PV} it was almost unchanged.

These results indicate that for the naive model-based selector, the family-wise risk and the family-wise unfairness do increase with the number of states, while our bootstrap-based selectors can scale far better with respect to the number of states.

Effect of sample size per state-action

To see how the sample size per state-action m affects the family-wise metrics, we also conducted experiments under different m ranging from 8 to 20. The number of states was fixed to 50 and the bootstrap size was set to 20. Figures 5.10 and 5.11 show the empirical

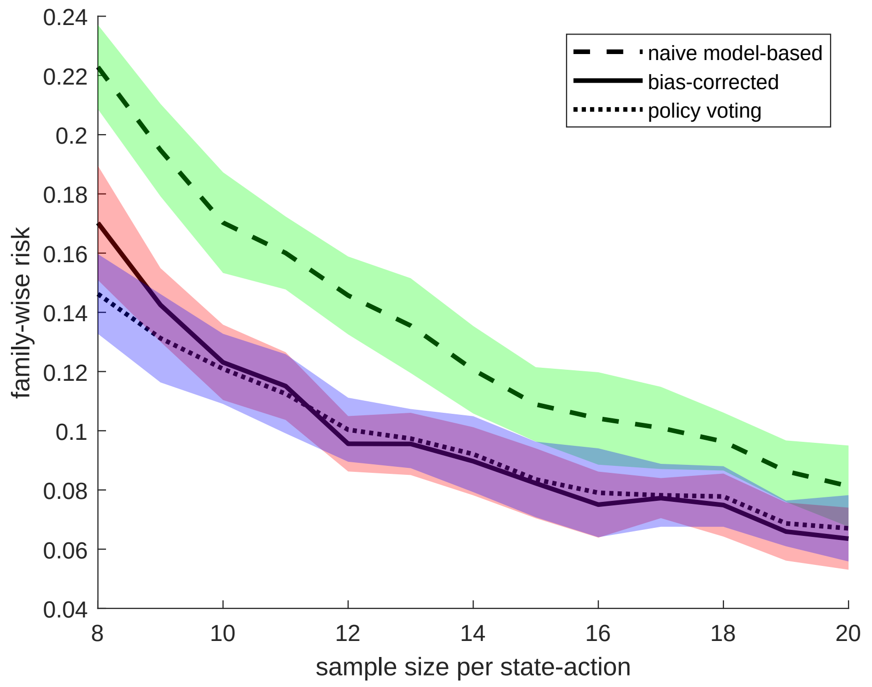

Figure 5.10: Normalised family-wise policy selection risk vs sample size per state-action.

normalised family-wise risk and family-wise unfairness, respectively.

From Figure 5.10 it can be seen that the family-wise risk reduced as the sample size per state-action m grew, but the relations between the selectors did not change much. Both of our bootstrap-based selectors W_{BC} and W_{PV} were significantly better than the naive model-based selector W_{MB} . At very small sample size $m \leq 10$, the BTBC selector was slightly worse than the BPV selector, but the difference was not significant except for $m = 8$.

Figure 5.11 showed that the family-wise unfairness also reduced with the sample size per state-action for W_{MB} and W_{BC}. For W_{PV}, it was close to 0 even for very small m, far better than both W_{MB} and W_{BC}. Thus, when fairness is needed and sample size is small, BPV can be a better choice than BTBC.

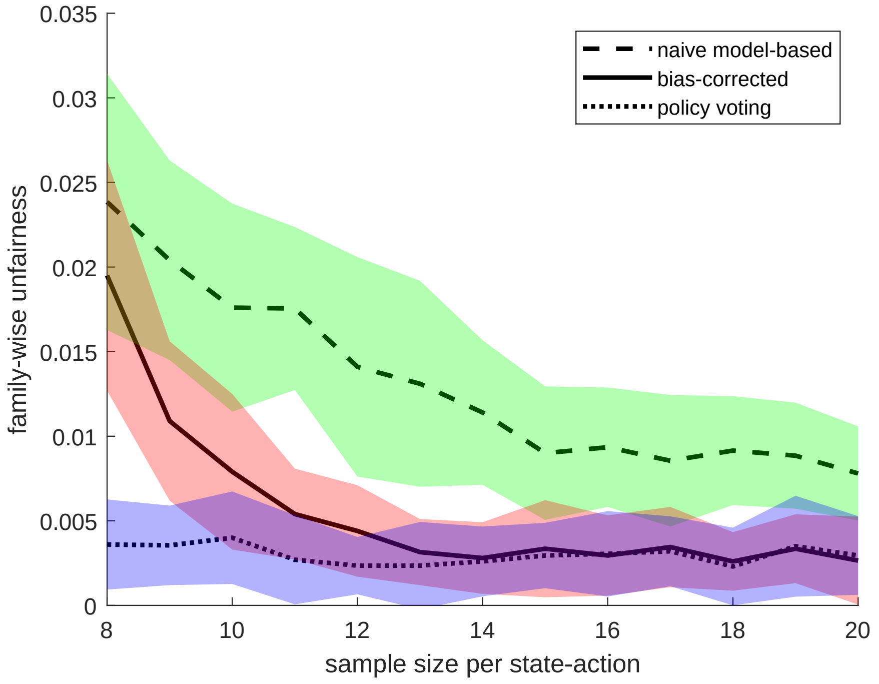

Figure 5.11: Normalised family-wise unfairness vs sample size per state-action.

5.5.2 Policy refinement through tournaments

This section presents our empirical results for our tournament-based policy refinement methods including the Bias-corrected Tournament (BCT) and the Policy Voting Tournament (PVT).

Performance metrics

The tournament-based policy refinement methods have two separate parts, where the first part generates the competitors for the tournament, and the second part conducts the tournament to decide the champion. Thus, we define several performance metrics to indicate the effectiveness of each part as well as the overall performance, given as follows.

Definition 5.8. Let k be the total number of runs. In these k runs, let k^* be the number of the runs where the original output policy is optimal, and k^+ be the number of the runs where at least one policy generated by bootstrap is better than the original output policy. Let k^W_{win} and k^W_{lose} be the number of runs where the champion policy of the tournament with selector W is better than and worse than the original output policy, respectively. Then the metrics are defined as follows.

- Rate of Generating at least one Better policy: $\mathrm{RGB} = k^+/(k - k^*)$.
- Rate of Successful/Unsuccessful Refinement when at least one better policy is generated: $\mathrm{RSR}^W = k^W_{\mathrm{win}}/k^+$, $\mathrm{RUR}^W = k^W_{\mathrm{lose}}/k^+$.
- Overall Improvement: $\mathrm{OI}^W = \mathrm{RGB}(\mathrm{RSR}^W - \mathrm{RUR}^W) = (k^W_{\mathrm{win}} - k^W_{\mathrm{lose}})/(k - k^*)$.

RGB measures the effectiveness of bootstrap policy generation. RSR and RUR measure the effectiveness of policy selection tournament when the policy generation is successful. OI measures the effectiveness of the policy refinement method as a whole.

The modified maze domain

Since our double chain MDPs used in the previous section has only two distinct policies, it is not suitable for testing policy generation where more than one policy can be generated. Thus, we modified the maze domain previously used in Section 4.5.3 for the experiments.

The map of the maze been used was still the one presented in Figure 4.13 of Chapter 4. However, the transitions and the rewards were modified to increase the number of suboptimal but still sufficiently competitive policies. Specifically, in the modified maze, the goal reward is set such that the true state value at the start state for policies obtaining less than three flags and entering the goal with minimum expected number of steps is 90% of the true state value for the optimal policy which obtains all three flags before entering the goal. In addition, entering the traps no longer sends the agent back to the start state. Instead, the agent is forever trapped at that trap position, and at each step thereafter receives a reward such that the state value at the start state for a policy which intentionally enters that trap with minimum expected number of steps is 95% of the true optimal state value. To make these reward settings possible, the transition of the movement actions at general states is changed to having 0.7 probability to move to the intended direction while having 0.3 probability to stay at the current position. Without this change, it is not possible to find the appropriate reward settings using Value Iteration.

Under this setting, the optimal policy is still the one that collects all three flags and enters the goal with minimum expected number of steps, but the policies that enters the goal with less flags and the ones that intentionally enters the traps also have 90% and 95% of the optimal utility, which makes figuring out the true optimal policy more difficult in the modified maze than the original maze.

Results

We conducted an experiment to see how the performance metrics proposed in Definition 5.8 change with the size of the tournament (i.e. the number of candidate policies). In this experiment, the following process was repeated 200 times for each setting of tournament size to get the average results. In each run, we first obtained a dataset containing $m = 8$ transition observations for each state-action pair. Then the dataset was passed to naive model-based RL algorithm with Value Iteration as policy planning method to obtain the original output policy. Then, the dataset and the original output policy were passed to the tournament-based policy refinement methods (Bias-corrected Tournament and the Policy Voting Tournament) to yield the final result of the tournament process. Finally, the results of the 200 runs were put together to compute the metrics.

We set the bootstrap size parameter $b = 20$ for BTBC and BPV. The tournament size was set to 2, 4, 8, 16, ..., 128. The results are shown in Figures 5.12, 5.13, and 5.14. Clopper-Pearson method [Clopper and Pearson, 1934] was used to compute the 95% confidence intervals. Note that the x-axes of these figures are drawn in logarithmic scale.

From Figure 5.12 it can be seen that the rate of generating at least one policy better than the original output increased with the tournament size. This is not surprising because when the original output is not optimal, each bootstrap-generated policy has an independent chance to be a policy better than the original in terms of its true utility, and thus a larger tournament size means a bigger chance of generating at least one better policy through bootstrap policy generation.

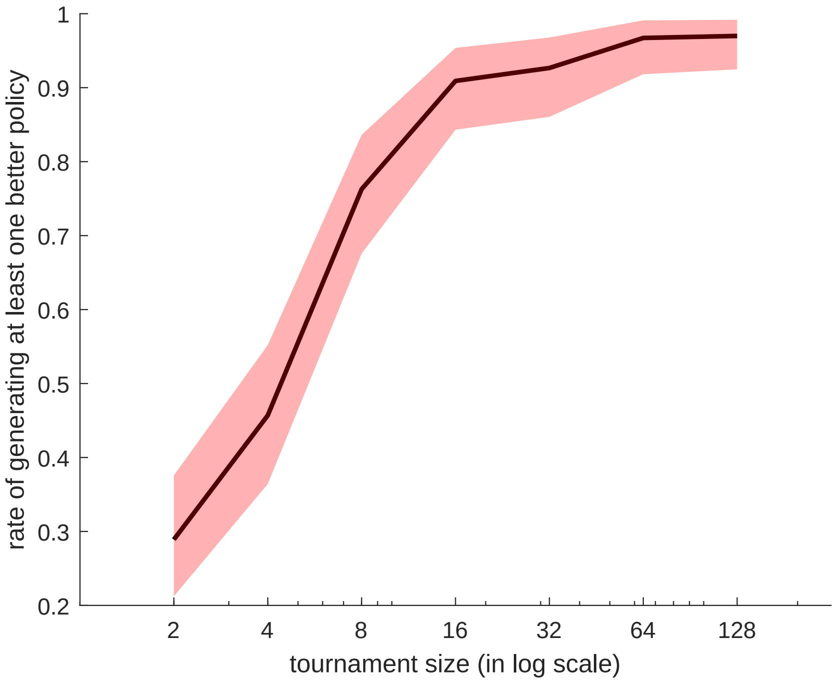

Figure 5.12: Rate of Generating at least one Better policy (RGB) vs tournament size in the modified maze domain.

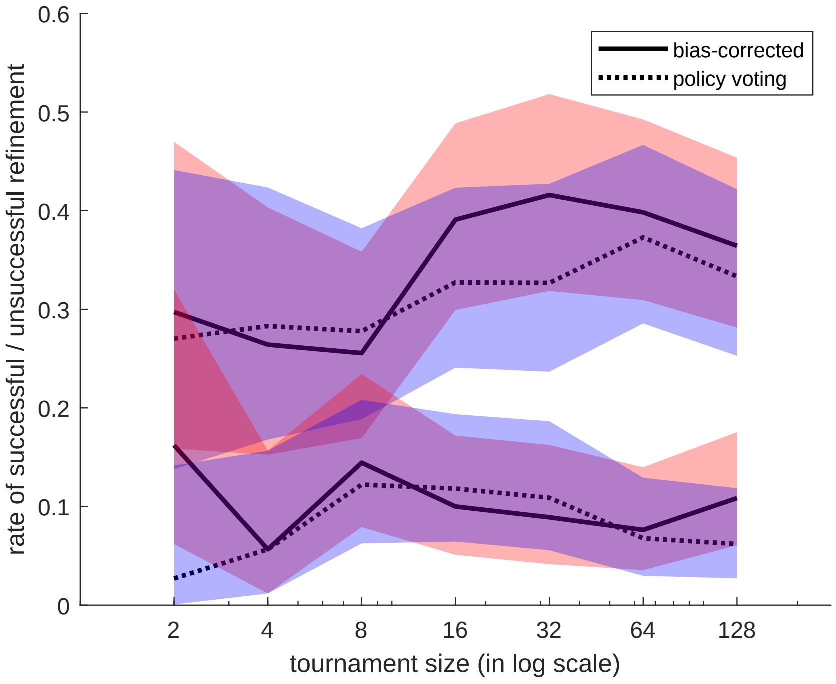

Figure 5.13: Rate of Successful/Unsuccessful Refinement when at least one better policy is generated (RSR, RUR) vs tournament size in the modified maze domain. The pair of curves above $y = 0.2$ are RSR, while the ones below $y = 0.2$ are RUR.

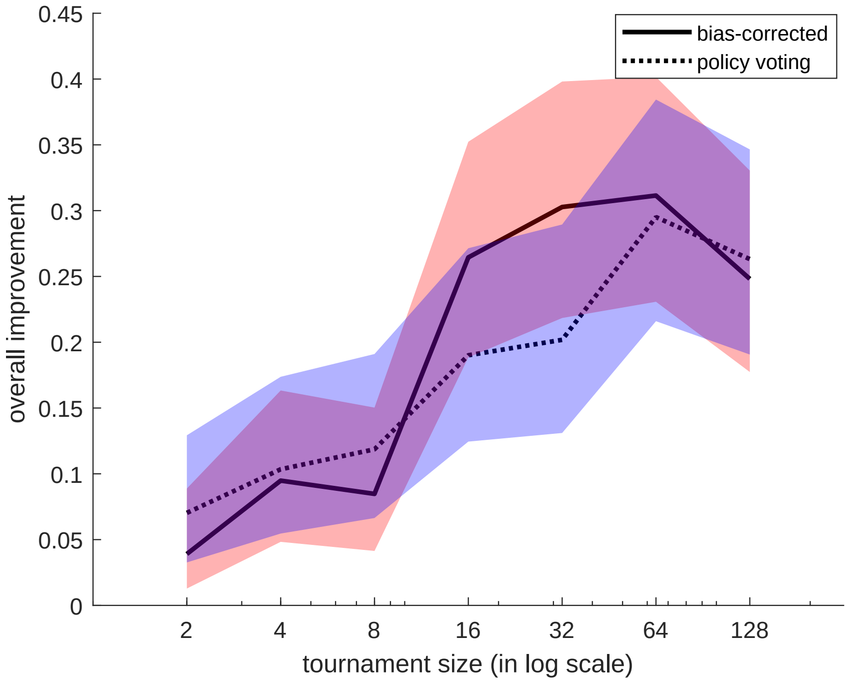

Figure 5.14: Overall Improvement (OI) vs tournament size in the modified maze domain.

Figure 5.13 shows the rates of successful and unsuccessful refinement given that at least one better policy was generated. The curves above $y = 0.2$ are RSR and the ones below $y = 0.2$ are RUR. Red and blue areas show the confidence intervals of results for BCT and PVT, respectively. As can be seen from the curves, the BCT and PVT methods had very similar rate of success and failure in this experiment, though BCT performed slightly better than PVT in terms of this metric when the tournament size is larger. The reason for the better performance of BCT is possibly that the optimal policy (collect all flags and go to the goal) in the modified maze domain involves a much longer cycle than the suboptimal policies (collect less flags and/or go into the traps), and thus its estimated value distribution is more heavily tailed. This gave advantage to the family-wise unfair BTBC selector which has shown to be favouring policies with such a property.

It is also interesting to see that the increased tournament size did not also increase the rate of successful refinement. This is possibly due to the fact that when the tournament becomes larger, although the average number of policies better than the original one increases, the number of matches they need to win in order to become the final champion also increases. Thus, the gain and the loss due to having a larger tournament neutralised with each other, which resulted in an almost fixed rate of successful refinement in this experiment.

Finally, Figure 5.14 shows the overall improvement obtained by using our policy refinement methods. Although increasing the tournament size did not increase the rate of successful refinement given that at least one better policy was generated, the tournament size being larger increased the probability of generating better policies, and thus when the two parts were combined, the overall improvement increased with the tournament size.

Since by definition the value of the overall improvement metric is the probability of outputting a policy better than the original one minus the probability of outputting a worse one given that the original policy is not optimal, an OI score about 0.3 is a remarkable improvement for the overall performance. Considering that the sample size per state action m was only 8 in this experiment, the result showed that our policy refinement methods were highly effective in improving the learning quality when sample size is small.

5.6 Chapter Summary

In this chapter, we have proposed the family-wise policy selection risk and the family-wise unfairness as metrics for policy selection. We have analysed the properties of the two family-wise metrics and have shown that they are more suitable for representing the overall effectiveness of policy selection methods than the instance-wise risk and the strict fairness. We have also shown that the naive model-based policy selector is marginally better than the random guesser in terms of family-wise selection risk and also has non-zero family-wise unfairness in some family of MDPs, and thus needs to be improved. This answers the research question Q3.1 ("How to measure the effectiveness of policy selection methods? Is the naive model-based selector good under this measurement?") in Section

1.3.3 and leads to contribution 5 in Section 1.4.[1]

We have then proposed the Bootstrap-based Policy Voting (BPV) selector, and compared it with the bias-corrected selector based on our BTBC proposed in Chapter 4. We have further proposed two policy refinement methods, Bias-corrected Tournament (BCT) and Policy Voting Tournament (PVT). We have presented our empirical results on these methods, showing that both BTBC and BPV selectors had much less family-wise policy selection risk than the naive model-based selector, that BPV had less family-wise unfairness than BTBC, that both BCT and PVT could significantly improve the overall model-based RL performance even when the sample size is small. This answers the research questions Q3.2 ("How to improve the effectiveness of policy selection?") and Q3.3 ("How to use the improved policy selection methods to further improve overall model-based RL performance?") in Section 1.3.3 and leads to contribution 6 in Section 1.4.[2]

[1]Contribution 5: "We propose two metrics for measuring the effectiveness of policy selection methods, namely family-wise policy selection risk and family-wise unfairness, and show that they are more suitable for measuring the overall effectiveness than the original strict fairness proposed by [Doroudi et al., 2017] and the instance-wise policy selection risk. We then analyse the naive model-based selector to show that it is not satisfactory in terms of these metrics."

[2]Contribution 6: "We propose the Bootstrap-based Policy Voting (BPV) method for policy selection which has significantly better family-wise effectiveness than the naive model-based selector, and also has less family-wise unfairness than the bias-corrected selector based on our BTBC. We then propose two policy refinement methods, Bias-corrected Tournament (BCT) and Policy Voting Tournament (PVT), to improve overall model-based RL performance."

CHAPTER 6

CONCLUSIONS AND FUTURE WORK

This chapter summarises the conclusions of Chapters 3, 4, and 5, then suggests directions for future research.

6.1 Conclusions

The ultimate goal of this thesis is to make reinforcement learning more sample efficient, i.e. work better when the sample size is small due to the limited budget and the expensiveness of data collection. To this end, we have studied the exploration, the transformation bias correction, and the policy selection of model-based reinforcement learning in finite MDPs. Exploration decides how efficient the data is collected, while transformation bias correction and policy selection decide how well the data is used in finding good policies.

Our contributions are summarised in the following list, and are given more details in the subsequent subsections.

- We pointed out that the existing exploration strategies and their analysis are unsuitable in the learning-then-serving workflow due to their connection to learning-process cumulative reward. We proposed the planning for exploration (PFE) problem where the task of fulfilling data demands are formulated as augmented undiscounted MDPs and can be solved by our Value Iteration for Exploration Cost (VIEC) algorithm. We showed that the efficiency of exploration strategies can be better analysed under the PFE formulation.

- We analysed the exploration costs of the optimal exploration scheme, ε-greedy, R-MAX, and MBIE-EB in tower MDPs to expose the weakness of the existing strategies to the distance trap and reward trap, and to show the advantage of explicit planning for exploration.

- We conducted a systematic empirical study on the transformation bias of model-base RL, showing that the relative scale of the bias η increases with the number of states n and decreases with connectivity and sample size per state-action m. For chain MDPs we obtained an empirical formula $\eta = C - K^{n/m}$, where C and K are constants decided by the transition dynamics.

- We proposed the Bootstrap-based Transformation Bias Correction (BTBC) method to correct the transformation bias of model-based RL. Experiments showed that it worked well even with a small sample size.

- We proposed the family-wise policy selection risk and family-wise unfairness metrics to measure the overall effectiveness of policy selection methods. We gave analysis to the effectiveness of some basic selectors, which showed that our metrics are more informative than the strict fairness.

- We proposed the Bootstrap-based Policy Voting (BPV) method that has better effectiveness of policy selection than the naive model-based selector and less family-wise unfairness than our BTBC selector. We then proposed two policy refinement methods, Bias-corrected Tournament (BCT) and Policy Voting Tournament (PVT), that can improve overall model-based RL performance without requiring additional data.

6.1.1 Explicit planning for efficient exploration

In Chapter 3, we considered the exploration efficiency in the learning-then-serving workflow. Most existing researches on exploration assume the learning-while-serving workflow, where learning-process cumulative reward is the objective to be maximised. However, in reality, many applications of RL is made under the learning-then-serving workflow, where the agent learns a policy in a learning process and provides service in a separate serving process, which makes the learning-process cumulative reward completely irrelevant to both the overall learning performance and the sample efficiency. Consequently, most existing methods reduce to mediocre heuristics, and the theoretical guarantees become inapplicable.

In the learning-then-serving workflow, exploration efficiency is decided by how much data an algorithm collects in the learning process in order to produce a policy that can work sufficiently well in the serving process. We showed that since the quality of the final policy can be guaranteed by collecting sufficient data for each state-action pair, the general exploration problem can be translated into a data demand fulfilment problem, where the agent needs to collect the specified amount of data with number of steps as few as possible. A better solution to such a demand fulfilment problem thus collects less amount of redundant data, which leads to better overall sample efficiency.

We formulated the problem of data demand fulfilment for exploration, and showed that the optimal exploration scheme, which fulfils the data demand with minimum exploration cost, can be obtained by solving a corresponding augmented undiscounted MDP. The augmented undiscounted MDP can be solved by our Value Iteration for Exploration Cost (VIEC) algorithm if the transition probabilities of the original MDP are given. The exploration cost can be used as a sample-size-based efficiency metric for exploration, and is more suitable than the existing metrics such as sample complexity and regret that are based on learning-process cumulative reward.

Under our demand fulfilment formulation, we analysed the exploration efficiency of several important exploration strategies, such as ε-greedy, R-MAX, and MBIE-EB, in tower MDPs. By comparing their behaviours and exploration costs with the optimal exploration scheme, we discovered that the existing strategies are weak to distance traps and reward traps when used in the learning-then-serving workflow, and that the exploration scheme obtained from explicit planning for exploration does not suffer from these traps. Thus, explicit planning for exploration can improve the sample efficiency of model-based RL and is more promising than the existing strategies in the learning-then-serving setting.

6.1.2 Transformation bias of model-based RL and its correction

In Chapter 4, we studied the transformation bias of model-based RL. The estimated state/action values are nonlinear functions to the estimated transition probabilities, and thus no matter the estimated transitions are unbiased or not, the estimated state/action values are biased. The correction of the transformation bias was historically considered unimportant, has rarely been studied, and was not conducted in most researches and applications. However, our empirical study showed that the transformation bias can be surprisingly large and has significant impact on the overall RL performance when the sample size is small, and thus should not be ignored.

Through an empirical study, we revealed that the relative bias of the estimated state/action value η, the number of states n, and the sample size per state-action m satisfy the formula $\eta = C - K^{n/m}$ in chain MDPs, where C and K are constants decided by the transition dynamics. This indicates that to prevent the bias from increasing with the number of states, the sample size per state action has to be proportional to the number of states, which means that the total sample size has to increase quadratically with the number of states. Thus, when obtaining data is expensive, correcting the transition bias can be of great help for improving the accuracy of value estimates.

We proposed our Bootstrap-based Transformation Bias Correction (BTBC) method to correct the transformation bias of model-based RL without needing additional data. We showed through empirical results that BTBC can effectively reduce the transformation bias and thus improves the overall sample efficiency of model-based RL. In particular, we showed that our BTBC worked well even when sample size per state-action is notably small. This is not possible for the second-order approximation method proposed by [Mannor et al., 2007], which is affected by its own transformation bias when estimating the transformation bias of state/action values.

6.1.3 Effectiveness of policy selection in model-based RL and its improvement

In Chapter 5, we studied the policy selection process of model-based RL, which, to the best of our knowledge, has not been systematically studied specifically for model-based RL by other researchers. The policy selection process uses the estimated state/action values to decide the output policy, which has direct impact on the quality of the final product of RL. Due to the asymmetry of the distributions of estimated values, some values are more likely to be overestimated than the others, even if the transformation bias is absent/corrected. Thus, the naive policy selection method, which directly compare the estimated values to decide the output policy, is more likely to select the inferior policies that are likely to be overestimated.

To measure the effectiveness of policy selectors, we proposed two metrics, family-wise policy selection risk and family-wise unfairness, that can represent the overall properties of selectors in families of MDPs. We showed that these metrics are more suitable for comparing different selectors than the strict fairness by Doroudi et al. [2017] (which was originally proposed for Importance Sampling based RL) and our instance-wise policy selection risk. We gave analysis to some elementary selectors as well as the naive model-based selector, showing that the naive model-based selector is unsatisfactory in both family-wise policy selection risk and family-wise unfairness.

To improve the effectiveness of policy selection, we proposed our Bootstrap-based Policy Voting (BPV) selector. We compared it with the bias-corrected selector based on our BTBC for transformation bias correction, revealing both their connections and the important property that BPV selector is in general better in terms of family-wise unfairness than BTBC selector.

We also proposed a tournament-based method for policy selection problem that has more than two candidate policies, as well as a bootstrap-based policy generation method that can generate arbitrarily many candidate policies from the obtained data. We combined these two methods with our improved policy selectors to produce the Bias-corrected Tournament (BCT) and Policy Voting Tournament (PVT) methods for policy refinement, which can be used after the basic model-based RL process is ended to further improve the quality of the output policy without needing new data.

We presented our results for both the proposed selectors and the policy refinement methods. The results showed that both BPV and BTBC selectors can have significantly less family-wise policy selection risk and family-wise unfairness than the naive model-based selector, though the BTBC selector has higher family-wise unfairness than the BPV selector. The results also showed that the policy refinement methods based on the two selectors can significantly improve the overall performance of model-based RL with a small sample size.

6.2 Future work

In Section 1.2, we clarified that this thesis focuses on model-based RL in finite MDPs. This is because the model-based RL in finite MDPs is already complicated enough, especially when the sample size is, or required to be, small. On the other hand, in reality, many applications are better formulated as continuous MDPs. Thus, to make our results more widely applicable, generalising them to continuous MDPs is of high importance.

Apart from generalisation to continuous space, there exist many other interesting directions for future research, which are listed as follows.

- Approximation to the computation of the exploration cost. As pointed out in Section 3.3, VIEC requires computation time exponential to the sizes of the state and action spaces. This is unavoidable (unless P=NP) for exact computation because even the simplest case of the planning for exploration problems is NP-hard. However, there might exist both fast and sufficiently good approximation to the exact computation. Similar approximations have been proposed for Bayesian RL, of which the exact computation is intractable. It might be possible that such approximation methods for Bayesian RL, or at least the ideas behind those approximations, can be adapted to planning for exploration to find sufficiently good exploration schemes, which is helpful for improving the exploration efficiency of RL.

- Finite-sample analysis to the quality of the estimated exploration scheme. We did not provide analysis to how good the output exploration scheme by VIEC will be if the true transition probabilities are not available and the estimated ones are used instead. It might be possible to apply concentration inequalities to analysis the (ε, δ)-optimality of the estimated exploration schemes, although such analysis might be only of theoretical interest and is not very useful in practice.

- Extending the transformation bias correction method to cover the modelling error. The model learnt by model-based RL in finite MDPs can exactly represent the transition information contained in the collected data, and thus the bias studied in Chapter 4 comes purely from the nonlinear transformation. When approximation techniques are applied to the model construction and representation, there is an additional modelling error caused by the inability to exactly expressing the information from the data, which cannot be reduced by our BTBC method. However, it might be possible to adapt the bootstrap-based method to either directly reduce the modelling error, or indirectly reduce its impact on the estimation result. By doing so, the collected data can be used in a more efficient way, making RL work better with a smaller sample size.

- Transformation bias in model-free RL. In Section 4.6, we pointed out that because the model-free methods can be seen as approximations to model-based methods, it is possible that model-free methods also implicitly suffer from the transformation bias.

This topic needs further investigation, and if the existence of the transformation bias in model-free RL is confirmed, then a bias correction method can be helpful for improving the sample efficiency of model-free RL.

- Exploiting the family-wise unfairness. As shown in the experiment results presented in Section 5.5.2, the generally more unfair BTBC selector can still lead to results as good as (or even slightly better than) the BPV selector, when used in policy refinement. It will be useful if we can classify the MDPs by whether the policies with heavy tails in their estimated value distributions are better or not. If such classification can be done without actually running the experiment (according to, say, domain knowledge), then we can possibly exploit such information by intentionally choosing selection methods that are family-wise unfair (so that they are more likely to choose the better policies in specific subset of MDPs).

- Extension to action selection. When value comparison is conducted at the action level, it becomes an action selection problem, where the algorithm has to choose one action from the candidates. This problem shares great similarity with policy comparison, and also has strong connection with the maximisation bias of the Bellman equation based algorithms. In fact, since Monte-Carlo methods that do not rely on the Bellman equations also have to choose actions, there should be a general selection problem that applies to all RL approaches. Formulating such a problem and adapting our methods to it can possibly be very helpful for improving the overall RL efficiency.

LIST OF REFERENCES

Pieter Abbeel and Andrew Y. Ng. Apprenticeship learning via inverse reinforcement learning. In *Proceedings of the Twenty-First International Conference on Machine learning*, 2004.

Shipra Agrawal and Randy Jia. Optimistic posterior sampling for reinforcement learning: worst-case regret bounds. In *Advances in Neural Information Processing Systems*, pages 1184–1194, 2017.

David R. Appleton. May the best man win? *Journal of the Royal Statistical Society: Series D (The Statistician)*, 44(4):529–538, 1995.

Peter Auer and Ronald Ortner. Logarithmic online regret bounds for undiscounted reinforcement learning. In *Advances in Neural Information Processing Systems 19*, pages 49–56. MIT Press, Cambridge, MA, USA, 2006.

Peter Auer, Nicolo Cesa-Bianchi, and Paul Fischer. Finite-time analysis of the multiarmed bandit problem. *Machine learning*, 47(2-3):235–256, 2002.

Peter Auer, Thomas Jaksch, and Ronald Ortner. Near-optimal regret bounds for reinforcement learning. In *Advances in Neural Information Processing Systems*, pages 89–96, 2009.

Mohammad Gheshlaghi Azar, Ian Osband, and Rémi Munos. Minimax regret bounds for reinforcement learning. In *Proceedings of the 34th International Conference on Machine Learning*, 2017.

Peter L. Bartlett and Ambuj Tewari. REGAL: A regularization based algorithm for reinforcement learning in weakly communicating MDPs. In *Proceedings of the Twenty-Fifth Conference on Uncertainty in Artificial Intelligence*, pages 35–42, Corvallis, Oregon, USA, 2009. AUAI Press.

Andy G. Barto and Satinder P. Singh. On the computational economics of reinforcement learning. In *Connectionist Models: Proceedings of the 1990 Summer School*, San Francisco, CA, USA, 1990. Morgan Kaufmann Publishers Inc.

BBC. Uber halts self-driving car tests after death, 2018. URL `https://www.bbc.com/news/business-43459156`.

Marc Bellemare, Sriram Srinivasan, Georg Ostrovski, Tom Schaul, David Saxton, and Remi Munos. Unifying count-based exploration and intrinsic motivation. In *Advances in Neural Information Processing Systems*, pages 1471–1479, New York, USA, 2016. Curran Associates, Inc.

Marc G. Bellemare, Yavar Naddaf, Joel Veness, and Michael Bowling. The arcade learning environment: An evaluation platform for general agents. *Journal of Artificial Intelligence Research (JAIR)*, 47:253–279, 2013.

Ronen I. Brafman and Moshe Tennenholtz. R-max–a general polynomial time algorithm for near-optimal reinforcement learning. *Journal of Machine Learning Research*, 3: 213–231, 2002.

Cameron B. Browne, Edward Powley, Daniel Whitehouse, Simon M. Lucas, Peter I. Cowling, Philipp Rohlfshagen, Stephen Tavener, Diego Perez, Spyridon Samothrakis, and Simon Colton. A survey of Monte Carlo tree search methods. *IEEE Transactions on Computational Intelligence and AI in games*, 4(1):1–43, 2012.

Sébastien Bubeck, Rémi Munos, and Gilles Stoltz. Pure exploration in multi-armed bandits problems. In *International Conference on Algorithmic Learning Theory*, pages 23–37. Springer, 2009.

George Casella and Roger L. Berger. *Statistical Inference*. Duxbury Press, Pacific Grove, CA, USA, 2nd edition, 2002.

C. J. Clopper and E. S. Pearson. The use of confidence or fiducial limits illustrated in the case of the binomial. *Biometrika*, 26(4):404–413, 1934.

Christoph Dann and Emma Brunskill. Sample complexity of episodic fixed-horizon reinforcement learning. In *Advances in Neural Information Processing Systems*, pages 2818–2826, New York, USA, 2015. Curran Associates, Inc.

Anthony Christopher Davison and David Victor Hinkley. *Bootstrap methods and their application*, volume 1. Cambridge university press, 1997.

Peter Dayan. The convergence of TD (λ) for general λ. *Machine learning*, 8(3):341–362, 1992.

Richard Dearden, Nir Friedman, and David Andre. Model based Bayesian exploration. In *Proceedings of the Fifteenth Conference on Uncertainty in Artificial Intelligence*, pages 150–159, San Francisco, CA, USA, 1999. Morgan Kaufmann Publishers Inc.

Marc Deisenroth and Carl E. Rasmussen. PILCO: A model-based and data-efficient approach to policy search. In *Proceedings of the 28th International Conference on Machine Learning*, pages 465–472, 2011.

Vali Derhami, Vahid Johari Majd, and Majid Nili Ahmadabadi. Exploration and exploitation balance management in fuzzy reinforcement learning. *Fuzzy sets and systems*, 161 (4):578–595, 2010.

Shayan Doroudi, Philip S. Thomas, and Emma Brunskill. Importance sampling for fair policy selection. In *Proceedings of the Thirty-Third Conference on Uncertainty in Artificial Intelligence*, 2017.

Bradley Efron. Bootstrap methods: another look at the jackknife. In *The Annals of Statistics*, pages Vol. 7, No. 1 (Jan., 1979), pp. 1–26. Institute of Mathematical Statistics, 1979.

Horst A. Eiselt, Michel Gendreau, and Gilbert Laporte. Arc routing problems, part II: The rural postman problem. *Operations research*, 43(3):399–414, 1995.

Logan Engstrom, Andrew Ilyas, Shibani Santurkar, Dimitris Tsipras, Firdaus Janoos, Larry Rudolph, and Aleksander Madry. Implementation matters in deep RL: A case study on PPO and TRPO. In *International Conference on Learning Representations*, 2019.

Claude-Nicolas Fiechter. Efficient reinforcement learning. In *Proceedings of the Seventh Annual Conference on Computational Learning Theory*, pages 88–97, New York, USA, 1994. ACM.

Roy Fox, Ari Pakman, and Naftali Tishby. Taming the noise in reinforcement learning via soft updates. In *Proceedings of the Thirty-Second Conference on Uncertainty in Artificial Intelligence*, pages 202–211, 2016.

Ronan Fruit, Matteo Pirotta, Alessandro Lazaric, and Ronald Ortner. Efficient bias-span-constrained exploration-exploitation in reinforcement learning. In *Proceedings of the 35th International Conference on Machine Learning*, 2018.

Scott Fujimoto, Herke Hoof, and David Meger. Addressing function approximation error in actor-critic methods. In *Proceedings of the 35th International Conference on Machine Learning*, pages 1587–1596, 2018.

Stuart Geman, Elie Bienenstock, and René Doursat. Neural networks and the bias/variance dilemma. *Neural computation*, 4(1):1–58, 1992.

Mohammad Ghavamzadeh, Shie Mannor, Joelle Pineau, and Aviv Tamar. Bayesian reinforcement learning: A survey. *arXiv preprint arXiv:1609.04436*, 2016.

Aditya Grover, Jiaming Song, Ashish Kapoor, Kenneth Tran, Alekh Agarwal, Eric J. Horvitz, and Stefano Ermon. Bias correction of learned generative models using likelihood-free importance weighting. In *Advances in Neural Information Processing Systems*, pages 11058–11070, 2019.

Steffen Grünewälder and Klaus Obermayer. The optimal unbiased value estimator and its relation to LSTD, TD and MC. *Machine learning*, 83(3):289–330, 2011.

Shixiang Gu, Ethan Holly, Timothy Lillicrap, and Sergey Levine. Deep reinforcement learning for robotic manipulation with asynchronous off-policy updates. In *IEEE International Conference on Robotics and Automation (ICRA'17)*, pages 3389–3396, New York, USA, 2017. IEEE.

Arthur Guez, David Silver, and Peter Dayan. Efficient Bayes-adaptive reinforcement learning using sample-based search. In *Advances in Neural Information Processing Systems*, pages 1025–1033, 2012.

Arthur Guez, David Silver, and Peter Dayan. Scalable and efficient Bayes-adaptive reinforcement learning based on Monte-Carlo tree search. *Journal of Artificial Intelligence Research*, 48:841–883, 2013.

Tuomas Haarnoja, Haoran Tang, Pieter Abbeel, and Sergey Levine. Reinforcement learning with deep energy-based policies. In *Proceedings of the 34th International Conference on Machine Learning*, 2017.

Peter Hall. *The bootstrap and Edgeworth expansion.* Springer Science & Business Media, 1992.

Josiah Paul Hanna. *Data efficient reinforcement learning with off-policy and simulated data.* PhD thesis, The University of Texas at Austin, 2019.

Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The elements of statistical* *learning: data mining, inference, and prediction.* Springer Science & Business Media, 2009.

J. Stuart Hunter. The exponentially weighted moving average. *Journal of quality technology*, 18(4):203–210, 1986.

Jemin Hwangbo, Joonho Lee, Alexey Dosovitskiy, Dario Bellicoso, Vassilios Tsounis, Vladlen Koltun, and Marco Hutter. Learning agile and dynamic motor skills for legged robots. *arXiv preprint arXiv:1901.08652*, 2019.

Hiroyuki Iida, Makoto Sakuta, and Jeff Rollason. Computer shogi. *Artificial Intelligence*, 134(1-2):121–144, 2002.

Shahin Jabbari, Matthew Joseph, Michael Kearns, Jamie Morgenstern, and Aaron Roth. Fairness in reinforcement learning. In *Proceedings of the 34th International Conference on Machine Learning*, pages 1617–1626. PMLR, 2017.

Thomas Jaksch, Ronald Ortner, and Peter Auer. Near-optimal regret bounds for reinforcement learning. *Journal of Machine Learning Research*, 11(Apr):1563–1600, 2010.

Michael Janner, Justin Fu, Marvin Zhang, and Sergey Levine. When to trust your model: Model-based policy optimization. In *NeurIPS 2019 : Thirty-third Conference on Neural Information Processing Systems*, pages 12519–12530, 2019.

Nan Jiang and Lihong Li. Doubly robust off-policy value evaluation for reinforcement learning. In *Proceedings of the 33rd International Conference on Machine Learning*, pages 652–661, 2016.

Jiantao Jiao and Yanjun Han. Bias correction with jackknife, bootstrap, and Taylor series. *IEEE Transactions on Information Theory*, 2020.

Leslie P. Kaelbling, Michael L. Littman, and Andrew W. Moore. Reinforcement learning: A survey. *Journal of Artificial Intelligence Research*, pages 237–285, 1996.

Leslie Pack Kaelbling. *Learning in embedded systems.* MIT Press, Cambridge, MA, USA, 1993.

Sham M. Kakade. *On the sample complexity of reinforcement learning.* PhD thesis, University College London, London, U.K., 2003.

Michael Kearns and Satinder Singh. Near-optimal reinforcement learning in polynomial time. *Machine Learning*, 49(2-3):209–232, 2002.

Jens Kober, J. Andrew Bagnell, and Jan Peters. Reinforcement learning in robotics: A survey. *The International Journal of Robotics Research*, 32(11):1238–1274, 2013.

Levente Kocsis and Csaba Szepesvári. Bandit based Monte-Carlo planning. In *European Conference on Machine Learning*, pages 282–293, New York, USA, 2006. Springer.

J. Zico Kolter and Andrew Y. Ng. Near-Bayesian exploration in polynomial time. In *Proceedings of the 26th International Conference on Machine Learning*, pages 513–520, New York, USA, 2009. ACM.

Akshay Krishnamurthy, Alekh Agarwal, and John Langford. PAC reinforcement learning with rich observations. In *Advances in Neural Information Processing Systems*, pages 1840–1848, New York, USA, 2016. Curran Associates, Inc.

Volodymyr Kuleshov and Doina Precup. Algorithms for multi-armed bandit problems. *arXiv preprint: arXiv:1402.6028*, 2014.

Thanard Kurutach, Ignasi Clavera, Yan Duan, Aviv Tamar, and Pieter Abbeel. Model-ensemble trust-region policy optimization. In *International Conference on Learning Representations 2018*, 2018.

Tor Lattimore and Marcus Hutter. Near-optimal PAC bounds for discounted MDPs. *Theoretical Computer Science*, 558:125–143, 2014.

Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *Nature*, 521(7553): 436–444, 2015.

Jiwei Li, Will Monroe, Alan Ritter, Michel Galley, Jianfeng Gao, and Dan Jurafsky. Deep reinforcement learning for dialogue generation. *arXiv preprint arXiv:1606.01541*, 2016.

Lihong Li. Sample complexity bounds of exploration. In *Reinforcement Learning*, pages 175–204. Springer, New York, USA, 2012.

Lihong Li, Michael L. Littman, Thomas J. Walsh, and Alexander L. Strehl. Knows what it knows: a framework for self-aware learning. *Machine Learning*, 82(3):399–443, 2011.

Timothy P. Lillicrap, Jonathan J. Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning. *International Conference on Learning Representations 2016*, 2016.

Michael L. Littman, Thomas L. Dean, and Leslie P. Kaelbling. On the complexity of solving Markov decision problems. In *Proceedings of the Eleventh Conference on Uncertainty in Artificial Intelligence*, pages 394–402, San Francisco, CA, USA, 1995. Morgan Kaufmann Publishers Inc.

Yuping Luo, Huazhe Xu, Yuanzhi Li, Yuandong Tian, Trevor Darrell, and Tengyu Ma. Algorithmic framework for model-based deep reinforcement learning with theoretical guarantees. In *ICLR 2019 : 7th International Conference on Learning Representations*, 2019.

Nguyen Cong Luong, Dinh Thai Hoang, Shimin Gong, Dusit Niyato, Ping Wang, Ying-Chang Liang, and Dong In Kim. Applications of deep reinforcement learning in communications and networking: A survey. *IEEE Communications Surveys and Tutorials*, 21(4):3133–3174, 2019.

Marlos C. Machado, Sriram Srinivasan, and Michael Bowling. Domain-independent optimistic initialization for reinforcement learning. *AAAI Workshop on Learning for General Competency in Video Games*, 2015.

Marlos C. Machado, Marc G. Bellemare, and Michael Bowling. Count-based exploration with the successor representation. In *Proceedings of the 34th AAAI Conference on Artificial Intelligence*, 2020.

Mufti Mahmud, Mohammed Shamim Kaiser, Amir Hussain, and Stefano Vassanelli. Applications of deep learning and reinforcement learning to biological data. *IEEE Transactions on Neural Networks*, 29(6):2063–2079, 2018.

Horia Mania, Aurelia Guy, and Benjamin Recht. Simple random search provides a competitive approach to reinforcement learning. *arXiv preprint arXiv:1803.07055*, 2018.

Shie Mannor, Duncan Simester, Peng Sun, and John N. Tsitsiklis. Bias and variance in value function estimation. In *Proceedings of the Twenty-First International Conference on Machine Learning*, page 72, 2004. doi: 10.1145/1015330.1015402.

Shie Mannor, Duncan Simester, Peng Sun, and John N. Tsitsiklis. Bias and variance approximation in value function estimates. *Management Science*, 53(2):308–322, 2007. doi: 10.1287/mnsc.1060.0614.

Jarryd Martin, Suraj Narayanan Sasikumar, Tom Everitt, and Marcus Hutter. Count-based exploration in feature space for reinforcement learning. *arXiv preprint: arXiv:1706.08090*, 2017.

Rupert G. Miller. The jackknife – a review. *Biometrika*, 61(1):1–15, 1974.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A. Rusu, Joel Veness, Marc G. Bellemare, Alex Graves, Martin Riedmiller, Andreas K. Fidjeland, Georg Ostrovski, Stig Petersen, Charles Beattie, Amir Sadik, Ioannis Antonoglou, Helen King, Dharshan Kumaran, Daan Wierstra, Shane Legg, and Demis Hassabis. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, 2015.

Volodymyr Mnih, Adria Puigdomenech Badia, Mehdi Mirza, Alex Graves, Timothy Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. Asynchronous methods for deep reinforcement learning. In *International Conference on Machine Learning*, pages 1928–1937, New York, USA, 2016. ACM.

Andrew William Moore. *Efficient memory-based learning for robot control.* PhD thesis, University of Cambridge, Cambridge, U.K., 1990.

Rémi Munos, Tom Stepleton, Anna Harutyunyan, and Marc Bellemare. Safe and efficient off-policy reinforcement learning. In *Advances in Neural Information Processing Systems*, pages 1054–1062, 2016.

Anusha Nagabandi, Gregory Kahn, Ronald S. Fearing, and Sergey Levine. Neural network dynamics for model-based deep reinforcement learning with model-free fine-tuning. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 7559–7566. IEEE, 2018.

Rui Nian, Jinfeng Liu, and Biao Huang. A review on reinforcement learning: Introduction and applications in industrial process control. *Computers & Chemical Engineering*, page 106886, 2020.

Brendan O'Donoghue, Ian Osband, Remi Munos, and Volodymyr Mnih. The uncertainty Bellman equation and exploration. In *International Conference on Machine Learning*, pages 3836–3845, 2018.

Min-hwan Oh and Garud Iyengar. Directed exploration in PAC model-free reinforcement learning. *arXiv preprint arXiv:1808.10552*, 2018.

OpenAI, Christopher Berner, Greg Brockman, Brooke Chan, Vicki Cheung, Przemysław Dębiak, Christy Dennison, David Farhi, Quirin Fischer, Shariq Hashme, Chris Hesse, Rafal Józefowicz, Scott Gray, Catherine Olsson, Jakub Pachocki, Michael Petrov, Henrique Pondé de Oliveira Pinto, Jonathan Raiman, Tim Salimans, Jeremy Schlatter, Jonas Schneider, Szymon Sidor, Ilya Sutskever, Jie Tang, Filip Wolski, and Susan Zhang. Dota 2 with large scale deep reinforcement learning. *arXiv preprint arXiv:1912.06680*, 2019.

Dirk Ormoneit and Śaunak Sen. Kernel-based reinforcement learning. *Machine learning*, 49(2-3):161–178, 2002.

Ronald Ortner and Daniil Ryabko. Online regret bounds for undiscounted continuous reinforcement learning. In *Advances in Neural Information Processing Systems*, pages 1763–1771, New York, USA, 2012. Curran Associates, Inc.

Ronald Ortner, Daniil Ryabko, Peter Auer, and Rémi Munos. Regret bounds for restless Markov bandits. *Theoretical Computer Science*, 558:62–76, 2014.

Ian Osband and Benjamin Van Roy. Near-optimal reinforcement learning in factored MDPs. In *Advances in Neural Information Processing Systems*, pages 604–612, New York, USA, 2014. Curran Associates, Inc.

Ian Osband, Charles Blundell, Alexander Pritzel, and Benjamin Van Roy. Deep exploration via bootstrapped DQN. In *Advances in Neural Information Processing Systems*, pages 4026–4034, 2016.

Cosmin Păduraru, Doina Precup, Joelle Pineau, and Gheorghe Comănici. An empirical analysis of off-policy learning in discrete MDPs. In *Proceedings of the Tenth European Workshop on Reinforcement Learning*, pages 89–102, 2013.

Deepak Pathak, Pulkit Agrawal, Alexei A. Efros, and Trevor Darrell. Curiosity-driven exploration by self-supervised prediction. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 16–17, 2017.

Jason Pazis and Ronald Parr. Efficient PAC-optimal exploration in concurrent, continuous state MDPs with delayed updates. In *Proceedings of AAAI'16*, pages 1977–1985, Palo Alto, CA, USA, 2016. AAAI Press.

Jason Pazis and Ronald E. Parr. PAC optimal exploration in continuous space Markov decision processes. In *Proceedings of the 27th AAAI Conference on Artificial Intelligence*, pages 774–781, Palo Alto, CA, USA, 2013. AAAI Press.

Marcus B. Perry. The exponentially weighted moving average. In *Wiley Encyclopedia of Operations Research and Management Science*. American Cancer Society, 2011.

Pascal Poupart, Nikos Vlassis, Jesse Hoey, and Kevin Regan. An analytic solution to discrete Bayesian reinforcement learning. In *Proceedings of the 23rd international conference on Machine learning*, pages 697–704, New York, USA, 2006. ACM.

Doina Precup, Richard S. Sutton, and Satinder Singh. Eligibility traces for off-policy policy evaluation. In *Proceedings of the Seventeenth International Conference on Machine Learning*, page 80, 2000.

Martin Puterman. *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. Wiley-Interscience, Hoboken, NJ, USA, 1994.

Maurice H. Quenouille. Notes on bias in estimation. *Biometrika*, 43(3/4):353–360, 1956.

Karun Rao and Shimon Whiteson. V-max: tempered optimism for better PAC reinforcement learning. In *Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems*, volume 1, pages 375–382, 2012.

Martin Riedmiller, Thomas Gabel, Roland Hafner, and Sascha Lange. Reinforcement learning for robot soccer. *Autonomous Robots*, 27(1):55–73, 2009.

Joseph Lee Rodgers. The bootstrap, the jackknife, and the randomization test: A sampling taxonomy. *Multivariate Behavioral Research*, 34(4):441–456, 1999.

Gavin Adrian Rummery. *Problem solving with reinforcement learning*. PhD thesis, University of Cambridge Ph. D. dissertation, 1995.

Daniel Russo, Benjamin Van Roy, Abbas Kazerouni, Ian Osband, and Zheng Wen. A tutorial on Thompson sampling. *arXiv preprint arXiv:1707.02038*, 2017.

Ahmad El Sallab, Mohammed Abdou, Etienne Perot, and Senthil Kumar Yogamani. Deep reinforcement learning framework for autonomous driving. *Electronic Imaging*, 2017(19):70–76, 2017.

Jürgen Schmidhuber. Adaptive confidence and adaptive curiosity. Technical report, Institut fur Informatik, Technische Universitat Munchen, Munich, Germany, 1991.

John Schulman, Sergey Levine, Pieter Abbeel, Michael Jordan, and Philipp Moritz. Trust region policy optimization. In *Proceedings of the 32nd International Conference on Machine Learning*, pages 1889–1897, New York, USA, 2015. ACM.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

Iulian V. Serban, Chinnadhurai Sankar, Mathieu Germain, Saizheng Zhang, Zhouhan Lin, Sandeep Subramanian, Taesup Kim, Michael Pieper, Sarath Chandar, Nan Rosemary Ke, Sai Rajeshwar, Alexandre de Brebisson, Jose M. R. Sotelo, Dendi Suhubdy, Vincent Michalski, Alexandre Nguyen, Joelle Pineau, and Yoshua Bengio. A deep reinforcement learning chatbot. *arXiv preprint arXiv:1709.02349*, 2017.

Shai Shalev-Shwartz, Shaked Shammah, and Amnon Shashua. Safe, multi-agent, reinforcement learning for autonomous driving. *arXiv preprint arXiv:1610.03295*, 2016.

Jun Shao and Dongsheng Tu. *The jackknife and bootstrap.* Springer Science & Business Media, 2012.

David Silver, Aja Huang, Chris J. Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, Sander Dieleman, Dominik Grewe, John Nham, Nal Kalchbrenner, Ilya Sutskever, Timothy Lillicrap, Madeleine Leach, Koray Kavukcuoglu, Thore Graepel, and Demis Hassabis. Mastering the game of go with deep neural networks and tree search. *Nature*, 529(7587):484–489, 2016.

David Silver, Thomas Hubert, Julian Schrittwieser, Ioannis Antonoglou, Matthew Lai, Arthur Guez, Marc Lanctot, Laurent Sifre, Dharshan Kumaran, Thore Graepel, Timothy Lillicrap, Karen Simonyan, and Demis Hassabis. A general reinforcement learning algorithm that masters chess, shogi, and go through self-play. *Science*, 362(6419):1140–1144, 2018.

Alexander L. Strehl and Michael L. Littman. A theoretical analysis of model-based interval estimation. In *Proceedings of the 22nd International Conference on Machine learning*, pages 856–863, New York, USA, 2005. ACM.

Alexander L. Strehl and Michael L. Littman. An analysis of model-based interval estimation for Markov decision processes. *Journal of Computer and System Sciences*, 74(8): 1309–1331, 2008.

Alexander L. Strehl, Lihong Li, Eric Wiewiora, John Langford, and Michael L. Littman. PAC model-free reinforcement learning. In *Proceedings of the 23rd International Conference on Machine learning*, pages 881–888, New York, USA, 2006. ACM.

Alexander L. Strehl, Carlos Diuk, and Michael L. Littman. Efficient structure learning in factored-state MDPs. In *Proceedings of AAAI'07*, volume 7, pages 645–650, Palo Alto, CA, USA, 2007. AAAI Press.

Alexander L. Strehl, Lihong Li, and Michael L. Littman. Reinforcement learning in finite MDPs: PAC analysis. *Journal of Machine Learning Research*, 10:2413–2444, 2009.

Malcolm Strens. A Bayesian framework for reinforcement learning. In *International Conference on Machine Learning*, volume 2000, pages 943–950, 2000.

Richard S. Sutton. Temporal credit assignment in reinforcement learning. *University of Massachusetts, Department of Computer and Information Science*, 1984.

Richard S. Sutton. Integrated architectures for learning, planning, and reacting based on approximating dynamic programming. In *Proceedings of the seventh international conference on machine learning*, pages 216–224, New York, USA, 1990. ACM.

Richard S. Sutton and Andrew G. Barto. *Reinforcement learning: An introduction.* MIT press, 2nd edition, 2018.

Csaba Szepesvári. Algorithms for reinforcement learning. *Synthesis lectures on artificial intelligence and machine learning*, 4(1):1–103, 2010.

István Szita and András Lőrincz. The many faces of optimism: A unifying approach. In *Proceedings of the 25th International Conference on Machine Learning*, pages 1048–1055, New York, USA, 2008. ACM. ISBN 978-1-60558-205-4.

István Szita and Csaba Szepesvári. Model-based reinforcement learning with nearly tight exploration complexity bounds. In *Proceedings of the 27th International Conference on Machine Learning*, pages 1031–1038, New York, USA, 2010. ACM.

Adrien Ali Taiga, William Fedus, Marlos C. Machado, Aaron Courville, and Marc G. Bellemare. On bonus based exploration methods in the arcade learning environment. In *ICLR 2020 : Eighth International Conference on Learning Representations*, 2020.

Jie Tan, Tingnan Zhang, Erwin Coumans, Atil Iscen, Yunfei Bai, Danijar Hafner, Steven Bohez, and Vincent Vanhoucke. Sim-to-real: Learning agile locomotion for quadruped robots. In *Robotics: Science and Systems XIV*, volume 14, 2018.

Haoran Tang, Rein Houthooft, Davis Foote, Adam Stooke, Xi Chen, Yan Duan, John Schulman, Filip DeTurck, and Pieter Abbeel. # Exploration: A study of count-based exploration for deep reinforcement learning. In *Advances in Neural Information Processing Systems*, pages 2753–2762, 2017.

Philip Thomas and Emma Brunskill. Data-efficient off-policy policy evaluation for reinforcement learning. In *International Conference on Machine Learning*, pages 2139–2148, 2016.

Philip Thomas, Georgios Theocharous, and Mohammad Ghavamzadeh. High confidence policy improvement. In *Proceedings of the 32nd International Conference on Machine Learning*, pages 2380–2388, 2015.

Sebastian B. Thrun and Knut Möller. Active exploration in dynamic environments. In *Advances in Neural Information Processing Systems*, pages 531–538, San Francisco, CA, USA, 1992. Morgan Kaufmann Publishers Inc.

Arryon D. Tijsma, Madalina M. Drugan, and Marco A. Wiering. Comparing exploration strategies for Q-learning in random stochastic mazes. In *2016 IEEE Symposium Series on Computational Intelligence (SSCI)*, pages 1–8. IEEE, 2016.

Michel Tokic and Günther Palm. Value-difference based exploration: adaptive control between epsilon-greedy and softmax. *KI 2011: Advances in Artificial Intelligence*, pages 335–346, 2011.

Aristide Tossou, Debabrota Basu, and Christos Dimitrakakis. Near-optimal optimistic reinforcement learning using empirical bernstein inequalities. *arXiv preprint arXiv:1905.12425*, 2019.

Łukasz Kaiser, Mohammad Babaeizadeh, Piotr Miłos, Błażej Osiński, Roy H. Campbell, Konrad Czechowski, Dumitru Erhan, Chelsea Finn, Piotr Kozakowski, Sergey Levine, Afroz Mohiuddin, Ryan Sepassi, George Tucker, and Henryk Michalewski. Model based reinforcement learning for Atari. In *ICLR 2020 : Eighth International Conference on Learning Representations*, 2020.

Leslie G. Valiant. A theory of the learnable. *Communications of the ACM*, 27(11):1134–1142, 1984.

Hado Van Hasselt. Double Q-learning. In *Advances in Neural Information Processing Systems*, pages 2613–2621, New York, USA, 2010. Curran Associates, Inc.

Hado Van Hasselt, Arthur Guez, and David Silver. Deep reinforcement learning with double Q-learning. In *Proceedings of AAAI'16*, pages 2094–2100, Palo Alto, CA, USA, 2016. AAAI Press.

Oriol Vinyals, Igor Babuschkin, Wojciech M. Czarnecki, Michaël Mathieu, Andrew Dudzik, Junyoung Chung, David H. Choi, Richard Powell, Timo Ewalds, Petko Georgiev, Junhyuk Oh, Dan Horgan, Manuel Kroiss, Ivo Danihelka, Aja Huang, Laurent Sifre, Trevor Cai, John P. Agapiou, Max Jaderberg, Alexander S. Vezhnevets, Rémi Leblond, Tobias Pohlen, Valentin Dalibard, David Budden, Yury Sulsky, James Molloy, Tom L. Paine, Caglar Gulcehre, Ziyu Wang, Tobias Pfaff, Yuhuai Wu, Roman Ring, Dani Yogatama, Dario Wünsch, Katrina McKinney, Oliver Smith, Tom Schaul, Timothy Lillicrap, Koray Kavukcuoglu, Demis Hassabis, Chris Apps, and David Silver. Grandmaster level in StarCraft II using multi-agent reinforcement learning. *Nature*, 575(7782):350–354, 2019.

Tao Wang, Daniel Lizotte, Michael Bowling, and Dale Schuurmans. Bayesian sparse sampling for on-line reward optimization. In *Proceedings of the 22nd international conference on Machine learning*, pages 956–963, New York, USA, 2005. ACM.

Christopher John Cornish Hellaby Watkins. *Learning from delayed rewards.* PhD thesis, King's College, Cambridge, Cambridge, U.K., 1989.

Martha White and Adam White. Interval estimation for reinforcement-learning algorithms in continuous-state domains. In *Advances in Neural Information Processing Systems 23*, pages 2433–2441, 2010.

Steven D. Whitehead. Complexity and cooperation in Q-learning. In *Proceedings of the Eighth International Workshop on Machine Learning*, pages 363–367, 1991.

Shimon Whiteson, Matthew E. Taylor, and Peter Stone. Empirical studies in action selection with reinforcement learning. *Adaptive Behavior*, 15(1):33–50, 2007.

Marco Wiering and Martijn Van Otterlo. *Reinforcement Learning: State-of-the-Art.* Springer-Verlag Berlin Heidelberg, Berlin, Germany, 2012.

Jeremy Wyatt. *Exploration and inference in learning from reinforcement.* PhD thesis, University of Edinburgh, Edinburgh, U.K., 1998.

Liangpeng Zhang, Ke Tang, and Xin Yao. Increasingly cautious optimism for practical PAC-MDP exploration. In *Proceedings of the 24th International Joint Conference on Artificial Intelligence*, pages 4033–4040, Palo Alto, CA, USA, 2015. AAAI Press.

Liangpeng Zhang, Ke Tang, and Xin Yao. Explicit planning for efficient exploration in reinforcement learning. In *NeurIPS 2019 : Thirty-third Conference on Neural Information Processing Systems*, pages 7488–7497, 2019.

Yuke Zhu, Roozbeh Mottaghi, Eric Kolve, Joseph J. Lim, Abhinav Gupta, Li Fei-Fei, and Ali Farhadi. Target-driven visual navigation in indoor scenes using deep reinforcement learning. In *IEEE International Conference on Robotics and Automation (ICRA'17)*, pages 3357–3364, New York, USA, 2017. IEEE.

Shaofeng Zou, Tengyu Xu, and Yingbin Liang. Finite-sample analysis for sarsa with linear function approximation. In *Advances in Neural Information Processing Systems*, pages 8668–8678, 2019.