Mitigating Overestimation in Stochastic Environments: A Causal Approach to Planning in Reinforcement Learning

Elias F. Fyksen

Master of Science
Artificial Intelligence
School of Informatics
University of Edinburgh
2024

Abstract

This thesis investigates the integration of causal inference techniques into reinforcement learning to enhance planning in stochastic environments. Traditional reinforcement learning approaches often rely on conditional probabilities to estimate future rewards, leading to overestimation and suboptimal decision-making in environments where correlation does not imply causation. To address this, we develop a causally-informed future-value function using the do-operator, which corrects the overestimation problem by accurately accounting for causal dependencies in multi-step planning scenarios. However, this correction introduces a conservative bias, resulting in underestimation of achievable returns. Despite this understimation we show emperically that under several stochastic environments the overstimatiotn tends to harm the policy more then the conservative causally derived estimates.

To mitigate this underestimation, we also propose a novel state-dependent policy transformation method using planning projection matrices, which aligns the future-value function with the optimal value function. Our theoretical analysis shows that solutions must exist that can completely mitigate this problem. However, we are unable to provide convergence guarantees or effective methods for learning these solutions. Consequently, our focus is on empirical experiments that explore the effects of overestimation versus conservative estimates in various stochastic environments. The findings highlight the potential for more accurate and reliable decision-making in reinforcement learning, despite the challenges in fully realizing these theoretical solutions.

Research Ethics Approval

This project was planned in accordance with the Informatics Research Ethics policy. It did not involve any aspects that required approval from the Informatics Research Ethics committee.

Declaration

I declare that this thesis was composed by myself, that the work contained herein is my own except where explicitly stated otherwise in the text, and that this work has not been submitted for any other degree or professional qualification except as specified.

(*Elias F. Fyksen*)

Acknowledgements

I would like to express my gratitude to my supervisor, Dr. Michael Herrmann, for his valuable guidance, support, and encouragement throughout this project.

Table of Contents

Chapter 1

Introduction

1.1 Overview

The rapid advancement in artificial intelligence (AI) has led to significant progress in the field of reinforcement learning (RL), a domain concerned with how agents should take actions in an environment to maximize cumulative reward. Reinforcement learning has proven successful in a wide range of applications, from playing complex board games like Chess [20] and Go [19] to real-world tasks such as robotic control [2], autonomous driving [9], and recommendation systems [1].

In traditional reinforcement learning, agents often rely on model-free approaches, where the focus is on learning a policy directly from experience without explicitly modeling the environment. Techniques such as Q-learning [24, 13] and policy gradient methods [23, 12, 18] have been widely adopted due to their simplicity and effectiveness in various settings. However, these methods typically focus on the next immediate action, often overlooking the broader implications of planning multiple steps ahead. This limitation becomes particularly significant in complex environments where foresight and strategic planning are crucial for achieving optimal performance [17].

Planning, in the context of reinforcement learning, refers to the process of considering sequences of actions over multiple time steps to guide decision-making. This approach has been instrumental in achieving breakthroughs in domains requiring deep strategic thinking, such as in the game of Go, where algorithms like Monte Carlo Tree Search (MCTS) [5, 10] have been employed to search for optimal action sequences. Despite its success, planning in reinforcement learning poses significant challenges, especially in stochastic environments where outcomes are uncertain and depend on both the current state and the agent's actions.

One of the major challenges in planning within stochastic environments is the accurate estimation of future outcomes. Traditional reinforcement learning methods, which often rely on conditional probabilities, may struggle to capture the true causal relationships between actions and outcomes, leading to sub-optimal decision-making.

Causal inference, a field rooted in statistics, provides a framework for understanding cause-and-effect relationships in complex systems. Unlike traditional statistical methods that focus on correlations, causal inference seeks to uncover the underlying mechanisms that generate observed data [15]. This distinction is crucial in reinforcement learning, where the goal is not just to identify actions correlated with high rewards, but to determine the actions that actually cause high rewards.

The intersection of causal inference and reinforcement learning represents a promising direction for addressing the limitations of current planning methods. By incorporating causal models into the planning process, it may be possible to develop algorithms that better account for the uncertainties and complexities of stochastic environments. This approach could have the potential to enhance the effectiveness of reinforcement learning in a wide range of applications, from robotics and autonomous systems to healthcare and finance, where decision-making under uncertainty is a common challenge.

In this thesis, we explore the application of causal inference techniques to improve planning in reinforcement learning. We investigate how the integration of causal reasoning can lead to more accurate future-value estimations and, consequently, more optimal decision-making in stochastic environments. By addressing the limitations of traditional planning approaches, this research aims to contribute to the ongoing development of more intelligent and reliable AI systems.

1.2 Motivation

In the realm of reinforcement learning, planning plays a critical role in enabling agents to make informed decisions by considering the potential outcomes of future actions. Traditional planning methods, such as those employed in complex environments like games or robotic control, typically rely on algorithms like Monte Carlo Tree Search (MCTS). These methods operate by learning the expected reward from sampled experience data, followed by a greedification process that selects the sequence of actions with the highest estimated return. This approach has demonstrated considerable success in various domains; however, it also presents significant limitations that warrant closer

examination.

At the core of these planning methods is the assumption that sampled experience data can be used to accurately estimate the expected value of different action sequences. This estimation process, in essence, relies on conditional probabilities, where the expectation of future rewards is conditioned on the observed states and actions. While this technique is computationally feasible and has been effective in many applications, it implicitly assumes that these conditional expectations provide a reliable approximation of the true value function in the environment.

Our initial observations challenge this assumption, revealing that it is possible to construct environments where these conditional expectations systematically overestimate the achievable returns. In such cases, the greedification process—designed to maximize expected returns—can lead to suboptimal decisions, ultimately degrading the performance of the agent. This phenomenon raises a critical question: why do these planning methods, which have been so successful in many contexts, fail in certain environments?

To address this question, we turn to the tools of causal inference, a framework that allows us to analyze the relationships between actions and outcomes. Unlike traditional statistical methods that focus on correlations, causal inference provides a means to understand the true cause-and-effect mechanisms at play. By applying causal inference to the problem of planning in reinforcement learning, we can investigate the underlying reasons for the observed overestimation of expected returns.

Through this investigation, we derive alternative estimates for future rewards that do not suffer from the same overestimation issues. These causal estimates take into account the intricacies of the environment and the stochastic nature of the decision-making process, providing a more accurate reflection of the expected returns. Our empirical studies further demonstrate that this overestimation problem is not just an isolated issue but a consistent challenge across randomly generated Markov Decision Processes (MDPs).

The motivation for this research is thus twofold: first, to uncover the limitations of current planning methods that rely on conditional expectations, and second, to propose and validate alternative approaches based on causal inference that offer more reliable estimates of expected returns. By doing so, we aim to aid the development of more robust and effective planning algorithms that can be applied across a wide range of complex, real-world environments.

1.3 Related Work

1.3.1 Model-based planning

Model-based planning was shown to have benefits in certain environments as early as the 1990s, with the introduction of the Dyna architecture [21]. This method of planning involves learning a full model of the environment that allows for accurate planning when combined with a model-free learning rule. Later research has combined this with the modelling of complex environments [14, 4, 8]. However, accurate modelling of complex and highly stochastic environments has remained an elusive goal, as it requires computationally expensive methods as well as suffering from compounding errors.

1.3.2 Model-free planning

Many other methods have evolved alongside, such as DRC [6], MuZero [17], EfficientZero [25]. This method has focused on both overall performance as well as sample efficiency. They completely circumvent the problem of model-based planning by directly modelling the future value given a sequence of actions. However, this gives rise to the problem of overestimating the value, which forms the basis of this thesis. While they mention the problem they do not go into detail on the underlying problem and rely on conventional techniques such as regularisation to mitigate the effects.

1.3.3 Stochastic latent models

Other methods use stochastic latent models. The primary example of this is the SimPLe[8] and Dreamer[7] architectures. These architectures attempt to approximate the environment by a stochastic latent model. These models show that you can accurately model many complex environments using a reduced stochastic latent space. They show impressive results, especially in sample efficiency and robustness to many environments. On the other hand, they exclusively do offline planning. This means that they use simulated experiences to train an agent, but there is no planning done during interaction with the environment.

1.3.4 Stochastic planning

Some work has also been done on the extension of the planning process in MuZero[17] called Stochastic MuZero[3]. This work uses VQ-VAE (Vector Quantised Variational

AutoEncoder) as well as introduces the concept of afterstates. These techniques allow them to integrate stochasticity directly into the planning process. This was shown to greatly improve the performance in highly stochastic board games such as Backgammon.

1.3.5 Summary

Although there is a wide variety of planning approaches used we can see that much of the related work has relied on either improving the planning algorithm or introducing stochasticity into models used for planning, rather than exploring the overestimation of these expectations directly and looking at alternative estimates.

Chapter 2

Background

2.1 Notation

Here is some notation that will be used throughout the document. Subscripts will generally refer to timesteps, (e.g. S_t, A_t, R_t refers to state, action, and reward at timestep t. A sequence of variables over multiple timesteps (from a to b) is noted by subscript $a:b$ (e.g. $A_{t:t+k} = (A_t, A_{t+1}, ..., A_{t+k-1})$. Superscript with parenthesis generally refers to a constant value in an MDP (e.g. $S^{(0)}$ refers to state 0 in the MDP, and $A^{(0)}$ refers to action 0).

The expectation given a policy refers to the expected value when evaluating the MDP using a specific policy. E.g.:

$$\mathbb{E}_\pi[f(S_{t+1})|S_t = s_t] = \sum_{s_{t+1}, a_t} f(s_{t+1}) p(s_{t+1}|s_t, a_t) \pi(a_t|s_t)$$

Where p denotes the state-action transition probabilities, and f is any arbitrary function $f : \mathcal{S} \to \mathbb{R}$.

Similarly, we can have multiple-step expectations:

$$\mathbb{E}_\pi[f(S_{t+k})|S_t = s_t, A_{t:t+k} = a_{t:t+k}] = \sum_{s_{t+1:k}} f(s_{t+k}) \prod_{i=t}^{t+k-1} p(s_{i+1}|s_i, a_i) \pi(a_i|s_i)$$

We will also use interventions, in the form of do operators (from causal inference). The following expression represents "given state s_t, if we perform action sequence $a_{t:t+k}$ **by intervention**, what is the expected value of $f(s_{t+k})$ (which in an MDP would be equivalent of ignoring the policy).

$$\mathbb{E}_\pi[f(S_{t+k})|S_t = s_t, \text{do}(A_{t:t+k} = a_{t:t+k})] = \sum_{s_{t+1:k}} f(s_{t+k}) \prod_{i=t}^{t+k-1} p(s_{i+1}|s_i, a_i)$$

2.2 Reinforcement Learning

Throughout this thesis, we will look at reinforcement learning problems. We will primarily focus on the use of planning within these problems and how stochastic environments affect the processes of planning.

Reinforcement learning is the process of learning how to act to maximise a scalar reward signal. This could be learning how to drive a car from point A to B, how to balance one object on top of another, or how to win in board games like Chess or Go.

2.2.1 Markov Decision Process

Throughout this thesis, we will use the standard MDP (Markov Decision Process) formulation of reinforcement learning. This consists of a state space $\mathcal{S}$, action space $\mathcal{A}$, and a probability distribution describing state-action reward and transition probabilities $p(s', r|s, a)$, where $s, s' \in \mathcal{S}$, $a \in \mathcal{A}$, and $r \in \mathbb{R}$. We also assume the Markov property, that given a current state and action pair, all future states and rewards are independent from all past states and actions.

The behaviour of an agent is decided by a policy $\pi(a|s)$, which describes a probability distribution over all actions $\mathcal{A}$ given each state in $\mathcal{S}$.

2.2.2 Planning

In traditional reinforcement learning, we only consider the next action to be executed. This can be done in many different ways such as tabular Q-learning [24], deep Q-networks [11], policy gradient methods [18] etc. However, in recent years the use of planning has shown great success in complex reinforcement learning tasks, such as Chess [20], Go [19], and Atari[17].

In reinforcement learning, planning refers to algorithms that, instead of considering only the next action, find sequences of actions that are used as heuristics for deciding which action to perform.

Planning can be split up into two primary methods. Model-based planning, in which the agent attempts to model the environment directly to use it in the planning process.

While this method can yield accurate results and has been shown at times to improve convergence speed [22], there are some problems with this approach. When we consider finite MDPs, a world model can be easily constructed using sample averages; however, when we consider more complex and/or real-world problems, such as playing visually complex video games and controlling robotic arms using camera input, several problems arise. The two primary problems are that both modelling and sampling continuous state spaces are non-trivial problems to solve.

This brings us to the second method of planning, which is model-free planning [6, 17]. In model-free planning, we do not attempt to model the environment directly. Instead, we derive value functions that take in a state and several actions and try to predict the future value directly without explicitly modelling the environment.

In this thesis, we will focus on model-free planning and use the tools of causal inference to explore and understand how stochastic environments affect such future-value functions.

2.3 Causal Inference

Causal inference is a powerful and increasingly critical area of study in statistics [15]. Unlike traditional statistical methods that focus primarily on associations or correlations between variables, causal inference seeks to understand and quantify the cause-and-effect relationships within data. This distinction is fundamental because, in many real-world applications, simply knowing that two variables are correlated does not provide actionable insight. Instead, decision-makers require knowledge of how one variable causally affects another to make informed decisions.

2.3.1 Do-operator

The first tool of causal inference that we need is the do-operator, introduced by Judea Pearl as part of his structural causal model (SCM) framework [16]. In reinforcement learning, we often encounter conditional probabilities, such as $P(Y|X = x)$, which describes the probability of an outcome Y given that we have observed a particular value x for a variable X. However, this expression reflects only a correlation and does not imply that X causes Y. The do-operator, denoted as $\text{do}(X = x)$, changes this perspective by representing the probability of Y given that X has been actively set to x through an intervention. The resulting probability, written as $P(Y|\,\text{do}(X = x))$, explicitly captures

the causal effect of X on Y. This is a useful concept in reinforcement learning since we are not interested in what actions correlate with high rewards; we are interested in what actions are causing high rewards.

A formal definition of the do-operator is given in Eq 2.1, where Z is the set of confounding variables between X and Y.

$$P(Y|\,\mathrm{do}(X=x)) = \sum_z P(Y|X=x,Z=z)P(Z=z) \tag{2.1}$$

2.3.2 Do-calculus

Judea Pearl developed do-calculus, a formal framework consisting of a set of rules that allows us to manipulate and evaluate expressions involving the do-operator. It is particularly powerful in causal inference because it provides a systematic method for identifying and estimating causal effects from observational data, even in the presence of complex dependencies and confounders.

Do-calculus operates on causal diagrams or causal graphs, where directed edges and causal effects represent the relationships between variables. The main goal of do-calculus is to transform an expression that involves the do-operator into an equivalent expression that does not involve the do-operator, enabling us to estimate causal effects using observational data.

While there are three primary rules of do-calculus, we will only use the 2nd rule in this thesis, and therefore we will not go into depth on the other two rules.

The second rule of do-calculus, often called the action/observation exchange rule, enables us to swap an intervention with observation under specific conditions. Formally, the second rule states:

$$P(Y|\,\mathrm{do}(X),Z) = P(Y|X,Z) \tag{2.2}$$

if X and Y are conditionally independent given Z in the subgraph where outgoing edges from X are removed. An example of this applying and not applying is shown in Fig. 2.1

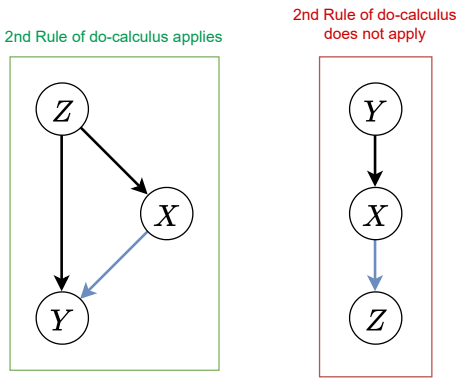

Figure 2.1: Two examples of SCMs where the 2nd rule of do-calculus applies (**left**) and does not apply (**right**). The outgoing edges which are ignored under the modified subgraph is marked in blue.

2.3.3 MDPs as Causal Models

We can represent MDPs as a structured causal model shown in Fig 2.2. In this structured causal model the edge from S_t to A_t represents the policy, and the edges from S_t to S_{t+k} and A_t to S_{t+k} represent the transition probabilities of the MDP.

Using this structured causal model, we can apply the rules and techniques of causal inference to derive estimates about the MDP from observed data. More crucially, the tools of causal inference can tell us whether we can determine the cause-and-effect relationship between actions and rewards from the observed data.

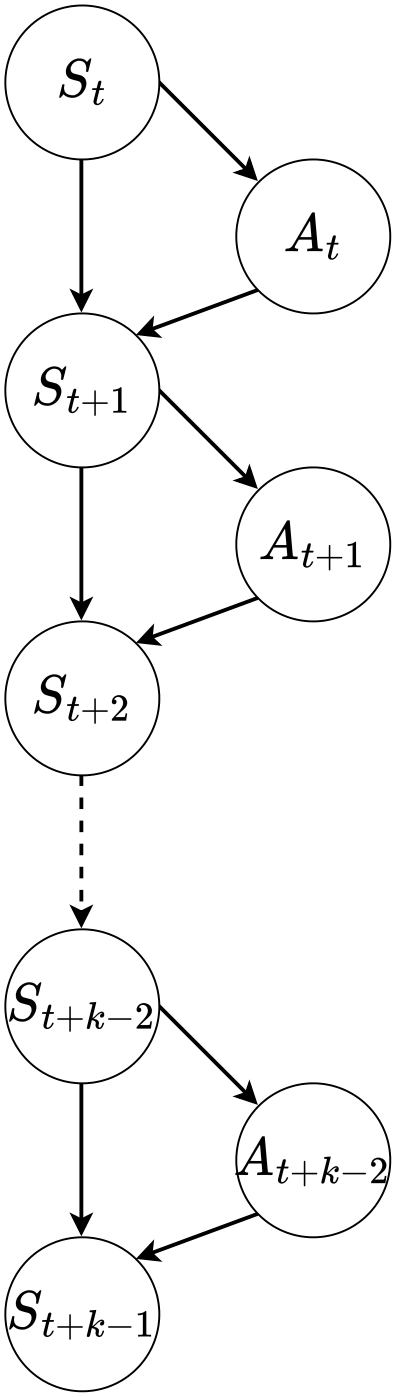

Figure 2.2: MDPs as structured causal models.

Chapter 3

Method

3.1 Causal Planning

It is worthwhile discussing why causal inference has specific benefits to planning in reinforcement learning. It is crucial to note that traditionally when doing reinforcement learning, we do not need to concern ourselves with causal inference. We will now show why this changes when we consider planning with multiple actions.

In this section, we primarily look at how causal inference gives us insight into how intermediate states in planning estimates give rise to a correlation vs causation problem that is not present in traditional single-step reinforcement learning.

3.1.1 Single-step do-operator

When we consider only a single step in an MDP if a specific action has a higher correlation with a certain next state (e.g. Eq 3.1), this must imply that the action is causing the state. This follows directly from the Markov assumption since the only variables governing the next state's probability are the current state and the action. So with the state being the same, the only variable that could be accounting for the change in probability is the action.

$$P(S_{t+1} = s' | S_t = s, A_t = 1) > P(S_{t+1} = s' | S_t = s, A_t = 2) \tag{3.1}$$

We can also show this using causal inference.

$$P(S_{t+1} = s' | S_t = s, \mathrm{do}(A_t = a)) = P(S_{t+1} = s' | S_t = s, A_t = a) \tag{3.2}$$

Eq 3.2 is a direct application of the 2nd rule of do-calculus (Eq 2.2).

This then gives us that:

$$P(S_{t+1} = s' | S_t = s, A_t = 1) > P(S_{t+1} = s' | S_t = s, A_t = 2)$$

Implies that:

$$P(S_{t+1} = s' | S_t = s, \text{do}(A_t = 1)) > P(S_{t+1} = s' | S_t = s, \text{do}(A_t = 2))$$

This implication tells us that if we **observe** a higher probability of s' given action $A_t = 1$, this implies that an **intervention** on the action ($\text{do}(A_t = 1)$) will result in a higher probability of state s'.

We can see from this that when considering a single step, a conditional on the action is always equivalent to a do-operator on the same action. Therefore, a correlation between an action and the next state, given the previous state, always implies that the action is causing the next state.

Note that this applies to any expectation of a function of the next state as well:

$$\mathbb{E}[f(S_{t+1}) | S_t = s, \text{do}(A_t = a)] = \mathbb{E}[f(S_{t+1}) | S_t = s, A_t = a] \tag{3.3}$$

Where $f : \mathcal{S} \to \mathbb{R}$. Notably, this includes any value function, telling us that (for a single step) if the action correlates with a high value, it must be causing the high value.

This is all consistent with traditional reinforcement learning.

3.1.2 Multi-step do-operator

Now, we will look at how this changes when we consider two steps.

Given:

$$P(S_{t+2} | S_t = s, \text{do}(A_t = a_t), \text{do}(A_{t+1} = a_{t+1}))$$

We can still apply the 2nd rule of do-calculus to the first action A_t, giving us:

$$P(S_{t+2} | S_t = s, \text{do}(A_t = a_t), \text{do}(A_{t+1} = a_{t+1})) = P(S_{t+2} | S_t = s, A_t = a_t, \text{do}(A_{t+1} = a_{t+1}))$$

However, we now notice that we can not (in general) apply the same rule to the second action (A_{t+1}). This is because the final state S_{t+2} is not (in general) independent of action A_{t+1} when ignoring the outgoing connections of A_{t+1}. This is because S_{t+2} and A_{t+1} has the confounding variable S_{t+1}. This is what's referred to as a backdoor in

Perl's causal framework [16].

This tells us that a correlation between the final state and the second action does not imply that the second action was causing the final state. It could be that the intermediate action and the final state were both caused by the intermediate state (S_{t+1}).

This has important implications for planning. When planning in reinforcement learning, we generally do not stick to our plan; the plan often changes upon new observations. Therefore, we only have observational data regarding planning, not interventional data. This makes the problem analogous to off-policy learning. Furthermore, causal inference tells us that from the observational data, we can not infer causation as we can with single-step reinforcement learning.

The primary takeaway from this should be that if we maximise any quantity w.r.t multiple actions using conditionals, we are simply maximising for actions that are correlated with the quantity under a certain policy and not necessarily causing the quantity itself. This can, therefore, lead greedification to fail.

3.2 Future-Value function

There have been several attempts at planning in deep reinforcement learning using future value functions (also called model-free planning [6, 17]). Model-free planning is done by approximating a future-value function (which we denote by D_π) given state s_t and action sequence $a_{t:t+k}$:

$$D_\pi(s_t, a_{t:t+k}) = \mathbb{E}_\pi[\gamma^k V_\pi(S_{t+k}) + \sum_{i=0}^{k-1} \gamma^i R_{t+i} | S_t = s_t, A_{t:t+k} = a_{t:t+k}] \tag{3.4}$$

This equation can be interpreted in the following way: "Under policy π, given that we observe state s_t and action sequence $a_{t:t+k}$, what is the expected value of state s_{t+k} and intermediate reward."

Then, in a given state, the agent employs a search strategy over possible future action sequences and uses the result to choose the immediate action to perform. A popular and widely successful strategy employed is MCTS (Monte-Carlo Tree Search). The goal of such search strategies is to maximise the expected future-value function by looking multiple steps ahead. I.e. approach the optimal action a_t^* given the future value

	$A^{(0)}$	$A^{(1)}$	π^*	V^*
$S^{(1)}$	10	0	$A^{(0)}$	10
$S^{(2)}$	0	1	$A^{(1)}$	1
$S^{(0)}$	9.1	6.9	$A^{(0)}$	9.1

Table 3.1: The Q^* table with the optimal policy π^* and the optimal value function V^* of MDP in Fig. 3.1

function (Eq. 3.5):

$$a_t^* = \underset{a_t}{\text{argmax}} \max_{a_{t+1:t+k}} D_\pi(s_t, a_{t:t+k}) \tag{3.5}$$

This can be interpreted as selecting the best sequence of k actions and performing the first action in that sequence.

3.3 Failure in stochastic environments

Here we will introduce a simple stochastic environment which shows a possible failure mode of this approach (Fig. 3.1). We will assume $\gamma = 1$, to simplify this example. The most important things to note are:

- Under the optimal policy $S^{(1)}$ is preferable to $S^{(2)}$
- State-action pair $(S^{(0)}, A^{(0)})$, has **no immediate** reward, but a high probability of transitioning into $S^{(1)}$ (**more preferable**).
- State-action pair $(S^{(0)}, A^{(1)})$, has **immediate** reward, but a high probability of transitioning into $S^{(2)}$ (**less preferable**).

Table 3.1 shows the Q^* values and the optimal policy π^* derived analytically from the MDP. In this table, we can see that the optimal action in $S^{(0)}$ is $A^{(0)}$. Now, let's build the future-value function D_π. First, we assume a behaviour policy ε-greedy (with $\varepsilon = 0.1$) to ensure all trajectories have some probability of occurring. All the trajectories, their probabilities and rewards can be found in Table 3.2. The future-value function for initial state $S^{(0)}$ is calculated in Table 3.3.

If we now perform the maximisation step from Eq 3.5 on the data from Table 3.3 we can see that from the perspective of the future-value function the optimal sequence of actions to take from state $S^{(0)}$ is $(A^{(1)}, A^{(0)})$. This would suggest that the agent should perform the next action $A^{(1)}$, which contradicts the results in the optimal policy and Q^* table (Table 3.1). Furthermore, the max value for $(S^{(0)}, A^{(0)})$ of the future-value

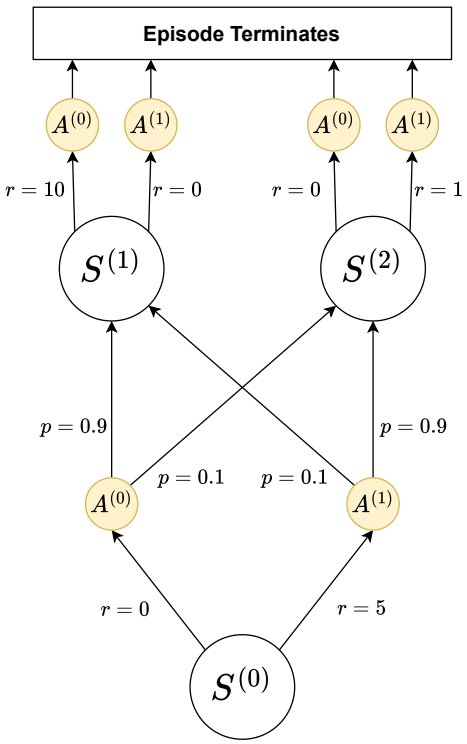

Figure 3.1: MDP of a simple stochastic environment. r represents the immediate reward of a state-action pair, and p represents the state-action-state transition probability.

Trajectory	Probability	Reward
$S^{(0)}, A^{(0)}, S^{(1)}, A^{(0)}$	81.23%	10
$S^{(0)}, A^{(0)}, S^{(1)}, A^{(1)}$	4.28%	0
$S^{(0)}, A^{(0)}, S^{(2)}, A^{(0)}$	0.48%	0
$S^{(0)}, A^{(0)}, S^{(2)}, A^{(1)}$	9.03%	1
$S^{(0)}, A^{(1)}, S^{(1)}, A^{(0)}$	0.48%	15
$S^{(0)}, A^{(1)}, S^{(1)}, A^{(1)}$	0.03%	5
$S^{(0)}, A^{(1)}, S^{(2)}, A^{(0)}$	0.23%	5
$S^{(0)}, A^{(1)}, S^{(2)}, A^{(1)}$	4.28%	6

Table 3.2: The trajectory probabilities and rewards under the behaviour policy outlined in Section 3.3.

Action Sequence ($a_{0:2}$)	$D_\pi(S^{(0)}, a_{0:2})$
$A^{(0)}, A^{(0)}$	9.94
$A^{(0)}, A^{(1)}$	0.68
$A^{(1)}, A^{(0)}$	11.79
$A^{(1)}, A^{(1)}$	5.99

Table 3.3: The future-value function from state $S^{(0)}$ calculated from Eq. 3.4 and Table 3.2.

function is 9.94 and (as discussed) the max value for $(S^{(0)}, A^{(0)})$ is 11.79. Both of these values are in fact overestimates and exceed both their respective Q^* values, and the V^* value of state $S^{(0)}$ (from Table 3.1).

3.4 Conditioning on future actions leads to overestimation

The first clue as to why we arrive at the incorrect results when using the future-value function can be seen in the interpretation of the equation. From the causal interpretation question in Eq 3.4 is "Given that we **observe** s_t and action sequence $a_{t:t+k}$, what is the expected future value of state s_k?". In reality, this is not the question we are after. The question we are after is "Given that we observe s_t and **preform** action sequence $a_{t:t+k}$, what is the expected future value of state s_k?". This is what is referred to in causal inference as an intervention on the action sequence $a_{t:t+k}$, and is noted by the do-operator. Writing the corrected version of Eq 3.4 using the do operator on the action sequence we get:

$$D_\pi(s_t, a_{t:t+k}) = \mathbb{E}_\pi[\gamma^k V_\pi(S_{t+k}) + \sum_{i=0}^{k} \gamma^i R_{t+i} | S_t = s_t, \text{do}(A_{t:t+k} = a_{t:t+k})] \tag{3.6}$$

In causal inference, the difference between a conditional and an intervention is that using the intervention (do-operator) removes all incoming connections from the DAG of the causal model. The causal DAG of a state action chain and the modified DAG under the do-operator can be seen in Figure 3.2. It is important to note here that there exist two cases in which the two DAGs represent the same probability.

- **Case 1:** If the policy is constant and independent of the state, the edge from S_i to A_i in the original DAG is superfluous, and the two DAGs are otherwise equivalent.

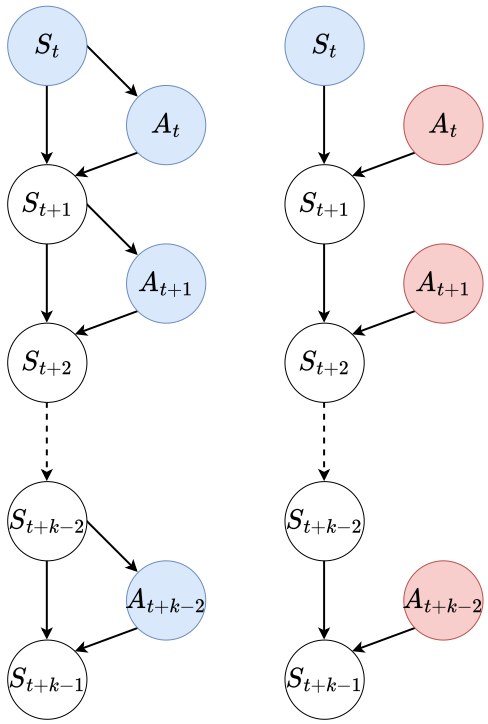

Figure 3.2: The original DAG representing the causal model of a state action chain (**left**) and the modified DAG under the do-operator from Eq 3.6 (**right**). Blue is used to indicate conditionals in Eq 3.4 and red is used to indicate the do-operator in 3.6.

- **Case 2:** If the MDP is completely deterministic then A_{t+k} becomes independent from S_{t+k}, given that we have conditioned on S_t and $A_{t:t+k}$. Therefore, the connection from S_i to A_i again becomes superfluous.

In terms of causal inference, (in general) naively using observational data and conditionals does not give us an accurate estimate of the question of the expected value given some intervention on the actions. In the language of causal inference, this is because the policy allows for a backdoor from A_i to S_i through S_{i-1}, as long as S_{i-1} is not completely determined. Given this, the question becomes if and how we can approximate the future-value function with intervention using only observational data. In the next section, we show how we can write the future-value function in recursive form. Furthermore, we show that when we do this the do-operator can be safely swapped with a conditional from the rules of do-calculus.

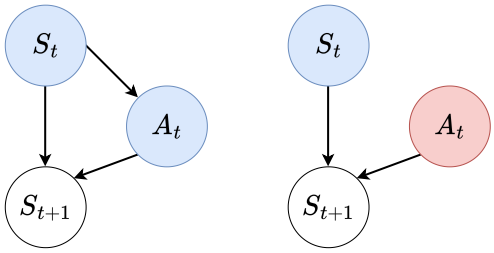

Figure 3.3: The original (**left**) and modified (**right**) DAG of the causal model used in Eq 3.7.

3.5 Future-value function in recursive form

We can define the future-value function in a recursive form, such that it is defined in terms of a shorter time-horizon version of itself:

$$\begin{aligned}
D_\pi(s_t, a_{t:t+k}) =& \mathbb{E}_\pi[\gamma^k V_\pi(S_{t+k}) + \sum_{i=0}^{k} \gamma^i R_{t+i} | S_t = s_t, \\
& \text{do}(A_{t:t+k} = a_{t:t+k})] \\
=& \mathbb{E}_\pi[\gamma(\gamma^{k-1} V_\pi(S_{t+k}) + \sum_{i=1}^{k} \gamma^{i-1} R_{t+i}) + R_t \\
& | S_t = s_t, \text{do}(A_{t:t+k} = a_{t:t+k})] \\
=& \mathbb{E}_\pi[\gamma \mathbb{E}_\pi[\gamma^{k-1} V_\pi(S'_{t+k}) + \sum_{i=1}^{k} \gamma^{i-1} R'_{t+i} \\
& | S'_{t+1} = S_{t+1}, \text{do}(A_{t+1:t+k} = a_{t+1:t+k})] + R_t \\
& | S_t = s_t, \text{do}(A_t = a_t)] \\
=& \mathbb{E}_\pi[\gamma D_\pi(S_{t+1}, a_{t+1:t+k}) + R_t | S_t = s_t, \text{do}(A_t = a_t)]
\end{aligned} \tag{3.7}$$

This new expectation only uses the one step of the state action chain DAG. This DAG can be seen in Figure 3.3. Using the 2nd rule of do-calculus we can safely turn the do-operator on $(A_t = a_t)$ into a regular conditional, since we are conditioning on S_t, so there exists no backdoor from A_t to S_{t+1}. This can also be reasoned by the fact that if we are conditioning on both S_t and A_t, then the edge from S_t to A_t (i.e. the behaviour policy) is irrelevant. E.g. in an MDP if both S_t and A_t is known, then S_{t+1} only depends on the transition probability (not on the policy).

By removing the do-operator using the 2nd rule of do-calculus we get the final equation (Eq. 3.8):

State (s_t)	Action Sequence ($a_{t:t+k}$)	Value ($D_\pi(s_t, a_{t:t+k})$)
$S^{(1)}$	$A^{(0)}$	10
$S^{(1)}$	$A^{(1)}$	0
$S^{(2)}$	$A^{(0)}$	0
$S^{(2)}$	$A^{(1)}$	1
$S^{(0)}$	$A^{(0)}$,$A^{(0)}$	9
$S^{(0)}$	$A^{(0)}$,$A^{(1)}$	0.1
$S^{(0)}$	$A^{(1)}$,$A^{(0)}$	2
$S^{(0)}$	$A^{(1)}$,$A^{(1)}$	1.9

Table 3.4: Evaluation of the future-value function using Eq 3.8

$$D_\pi(s_t, a_{t:t+k}) = \mathbb{E}_\pi[\gamma D_\pi(S_{t+1}, a_{t+1:k+1}) + R_t | S_t = s_t, A_t = a_t] \tag{3.8}$$

Furthermore, we note that the expectation is independent of the policy since the variable S_{t+1} is purely defined by the transition probability when conditioning on both S_t and A_t, and is, therefore, independent of the policy. This expectation can therefore be approximated using observational data from any policy.

3.6 Evaluating the new future-value function

We can now evaluate the new future-value function for the example in Section 3.3, using Eq 3.8. The result of this evaluation can be seen in Table 3.4. If we now apply the max operation from Eq 3.5 in state $S^{(0)}$ we can see that the best action sequence from state $S^{(0)}$ is $(A^{(0)},A^{(0)})$. This suggests that the best action from state $S^{(0)}$ is $A^{(0)}$ which is in agreement with the Q^* table in Table 3.1. We can also see that there are no longer any overestimations that violate the V^* function, and if we calculate the expected return of doing actions $(A^{(0)},A^{(0)})$ from $S^{(0)}$ we can indeed see that the exact expected return is 9 (in agreement with Table 3.4).

3.7 The upper-bound of planning

Although we now have a correct estimation for expected returns when performing k actions in the future we now have a different problem. The max value of $(S^{(0)},A^{(0)})$ from Table 3.4 is 9, while the Q^* value $(S^{(0)},A^{(0)})$ of is actually 9.1. It seems that we have just swapped an overestimation problem for an underestimation problem. The reason for this inconsistency is that when deciding multiple actions ahead in a stochastic

environment we can sometimes not achieve as high an expected reward as the optimal policy. This is because sometimes the optimal action A_{t+i} depends on S_{t+i}, so we can not accurately predict the optimal action multiple steps ahead. For this reason, the max of the future-value function will always be bounded by the optimal plan, which can be lower than the expected value of the optimal policy.

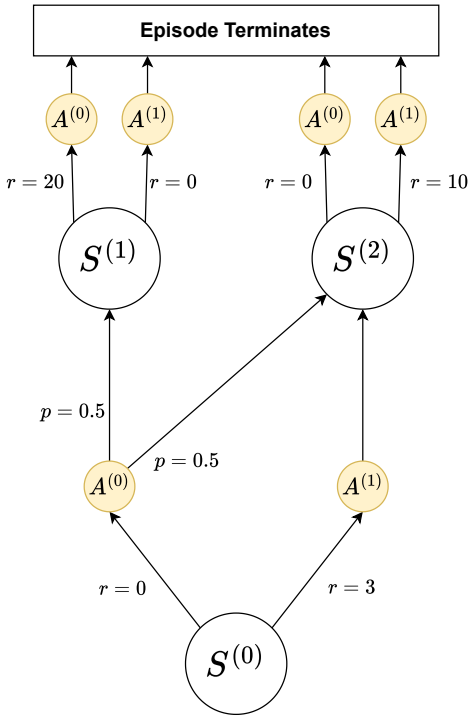

Figure 3.4: MDP where the underestimation leads to incorrect action selection

In the previous MDP this was not a problem because even considering the underestimation the order of the two actions in $S^{(0)}$ did not change. However, this does not always hold true. Figure 3.4 describes a new MDP with the same general structure, but different transition probabilities and rewards. In this new MDP the optimal $Q^{(*)}$ value of action $A^{(0)}$ and $A^{(1)}$ in state $S^{(0)}$ are 15 and 13 respectively. I.e. the optimal action in $S^{(0)}$ is $A^{(0)}$. However, when we do the same calculations for the optimal action sequence as before, we get that the optimal action sequence is $D_\pi(S^{(0)}, (A^{(1)}, A^{(1)})) = 13$ (see Table 3.5), suggesting that the optimal next action is $A^{(1)}$, which is incorrect.

This happens because the trajectories starting in $A^{(1)}$ do not have any stochasticity and therefore do not suffer any underestimation by the future-value function.

State (s_t)	Action Sequence ($a_{t:t+k}$)	Value ($D_\pi(s_t, a_{t:t+k})$
$S^{(1)}$	$A^{(0)}$	20
$S^{(1)}$	$A^{(1)}$	0
$S^{(2)}$	$A^{(0)}$	0
$S^{(2)}$	$A^{(1)}$	10
$S^{(0)}$	$A^{(0)}$,$A^{(0)}$	10
$S^{(0)}$	$A^{(0)}$,$A^{(1)}$	5
$S^{(0)}$	$A^{(1)}$,$A^{(0)}$	3
$S^{(0)}$	$A^{(1)}$,$A^{(1)}$	13

Table 3.5: Evaluation of the future-value function of the MDP in Figure 3.4

3.8 Raising the upper-bound by state-dependent policy transformations

A possible mitigation of the upper-bound problem is to transform the action space in a state-dependent way. We can define a matrix for every state in which every column of the matrix refers to a different distribution of actions for that state. We will refer to this matrix as a planning projection, where each state has a distinct planning projection $\mathcal{B}^{(n)}$ (where n is the state it belongs to). This means that the state-action probabilities for a policy is given by Eq 3.9, and the state-state transition probability is given by Eq 3.10:

$$\mathcal{P}^{\pi,\mathcal{B}}_{s,a} = \mathcal{B}^{(s)}_a \pi(\cdot|s) \tag{3.9}$$

$$\mathcal{P}^{\pi,\mathcal{B}}_{s,s'} = \sum_a p(s'|s,a)\mathcal{B}^{(s)}_a \pi(\cdot|s) \tag{3.10}$$

Doing this we have effectively transformed one MDP into another MDP where each action in the new MDP refers to a distribution over the actions in the original MDP. If we further restrict the planning projection matrices to be doubly stochastic we are guaranteed that a uniform policy in this new MDP corresponds to a uniform policy in the original MDP, which ensures that all states remain reachable.

It must be true that for any MDP there exists a planning projection such that the upper limit of D_{π^*} is V^*. I.e. that:

$$\max_{a_t:a_{t:t+k}} D_{\pi^*}(s_t, a_{t:t+k}) = V^*(s_t) \tag{3.11}$$

And that:

$$\underset{a_t}{\operatorname{argmax}} \max_{a_{t+1:t+k}} D_{\pi^*}(s_t, a_{t:t+k}) = \underset{a_t}{\operatorname{argmax}}\, Q^*(s_t, a_t) \tag{3.12}$$

This follows immediately from the fact that we can define a planning projection such that the first column of the matrix in each state is the optimal policy in that state. I.e.:

$$\mathcal{B}^{(s)}_{*,1} = \pi^*(\cdot|s) \tag{3.13}$$

And the remaining columns are uniformly distributed over the rest of the actions to ensure it is doubly stochastic. Under this transformed MDP the optimal policy is always $A^{(0)}$ (by definition), so even if the original MDP is stochastic the future-value function will give the correct max value since the optimal policy is constant.

3.9 On learning the planning projection

One might learn the planning projection by iterative improvement, by iterating between improving the policy and improving the planning projection. However, we have not been able to prove any convergence guarantees for this.

Even though it is not guaranteed to converge it could still be interesting to attempt to learn both parts by gradient ascent.

There is also the problem of ensuring that the planning matrices retain the doubly stochastic property, which is non-trivial. This property is not strictly necessary for the solution to be optimal, but it ensures that all states always remain reachable and that any ε-soft policy in the transformed MDP, remains an equivalent ε-soft policy in the original MDP, since uniform sampling of a double stochastic matrix results in uniform sampling of the underlying actions.

Due to the difficulties of learning the planning projections, we focus the rest of the thesis on the performance of the causal expectation without planning projections and leave planning projections for future research.

Chapter 4

Experiments

In the previous chapter, we showed by specifically constructed MDPs how naively using conditionals can lead to overestimates and how greedification of these estimates can lead to sub-optimal policies.

We will now show empirically that this overestimation is not unique to highly contrived MDPs. On the contrary, since the only thing required for overestimation to happen is that some future actions are correlated with high rewards without causing them, we should expect this to happen at some rate for most MDPs.

It is important to note that these MDPs are expected to be optimally solved by Q-Learning, which is why we include Q-Learning as a reference result. This is to demonstrate the sacrificed performance of the traditional planning approach.

4.1 Conditional vs Causal Estimates in Random MDPs

4.1.1 Method

First, we create 4 randomly generated MDPs. We then solve the optimal policy of each MDP by iteratively solving the Q function using a linear system of equation, and greedification. We know this will converge to the optimal policy by the policy improvement theorem and the Bellman optimality condition [22]. Now, using this policy, the transition probabilities, and the reward function, we analytically solve both the conditional (Eq. 3.4) and the causal (Eq. 3.8) version of the future value function (for two steps). We then randomly sample states from $\mathcal{S}$ and two actions from $\mathcal{A}$ as a plan. We then evaluate both of the future-value functions using this plan, as well as execute the plan and record the result of executing the plan.

Further hyper-parameters can be found in Appendix A.

4.1.2 Results

Figure 4.1 shows a scatter plot of the expected vs realised return of randomly sampled plans according to the causal and conditional expectations. The top row is the conditional expectations and the lower is the causal expectations. Each of the columns is a distinct randomly generated MDP. The black line is the identity line $(y = x)$. This means that every point above the black line had a higher expected return than what was realised and every point below the black line had a lower expected value than what was realised. For the conditional estimate, 68.7% of the plans had a lower-than-expected return and 31.3% had a higher, while for the causal estimate, 49.6% of the plans had lower-than-expected and 50.4% had a higher.

Since we are interested in the bias of the estimator, we draw a line fitting the optimal bias through the point cloud $(y = x + b)$, where the bias is the mean residual. For an unbiased estimator, the mean residual should be 0, and the optimal biased line should be the same as the black identity line. We see that in the conditional estimates, the biased line is consistently above the black identity line, showing that for the randomly generated MDPs the conditional estimate consistently overestimates the expected value of the plan. For the causal estimates, we can see that the blue biased line fits very closely to the black identity line, indicating that there is little bias in the estimator.

Figure 4.2 shows a bar chart of the mean residuals along with the standard error. We can see that the conditional mean residual is way above zero for all the experiments and the zero value is substantially outside the standard error. For the causal estimates, two of the residuals are below while two of them are above, and for 3 out of 4, the 0 line is within the standard error of the estimate, which is what we should expect for an unbiased estimate.

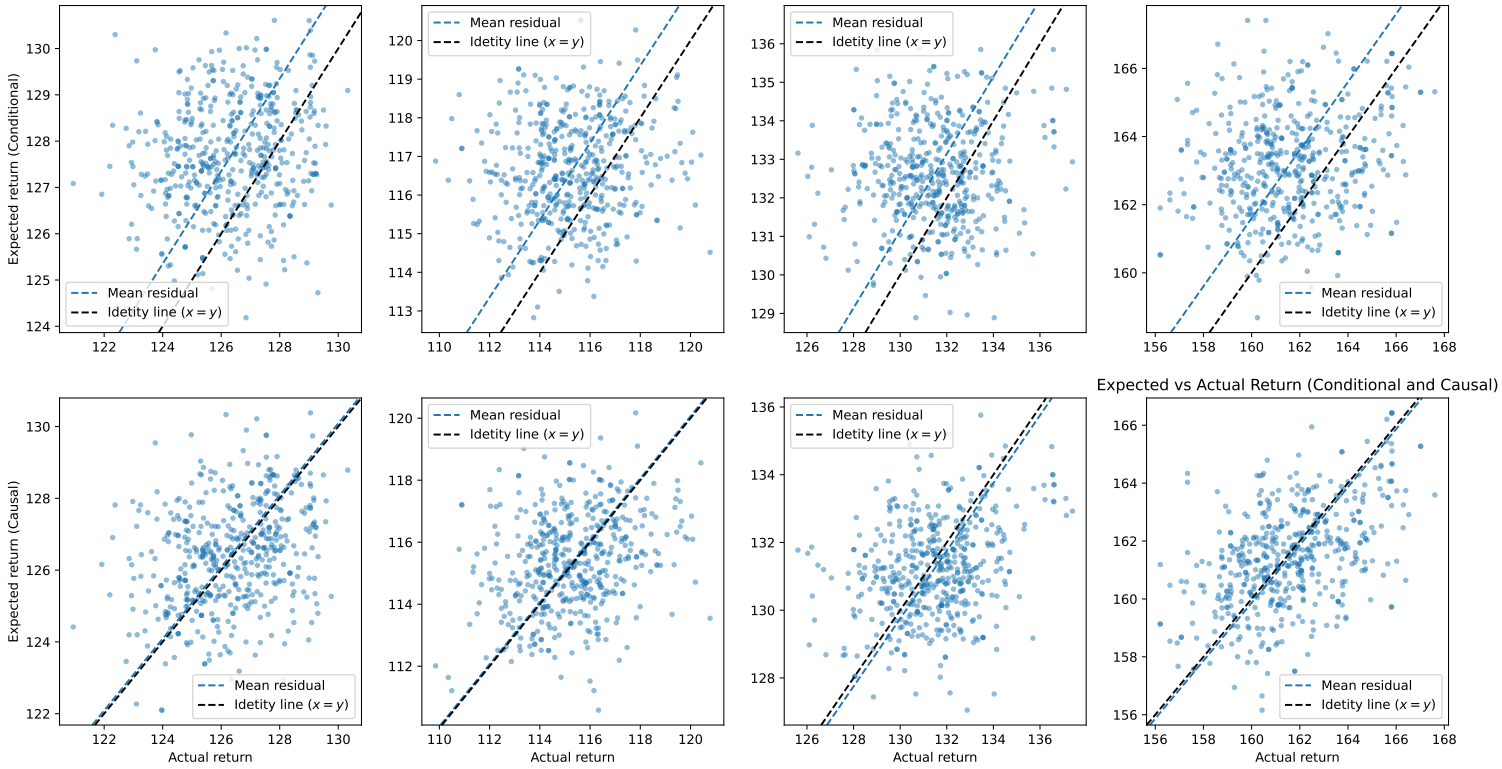

Figure 4.1: Expected vs realised returns of planning using conditional and causal estimates. Each column is a randomly generated MDP. The upper row shows the conditional results, while the lower row shows the causal results. Each graph plots the expected vs realised results of executing a given plan.

Figure 4.2: Residuals of the conditional and causal estimates when compared to the realised returns during simulation.

4.2 Conditional vs Causal Agents in Random MDPs

In this experiment, we look at how agents using different estimates perform in randomly generated MDPs with different amounts of lookahead. We look at both their performance and the expected value as reported by their value function. We compare these results to standard tabular Q-Learning, which should achieve optimal results under these conditions.

4.2.1 Method

Similar to the previous experiment we generate random MDPs. We then train three different agents in all of these MDPs. To ensure exploration we use an ε-greedy behaviour policy, with a linear epsilon decay strategy.

In this experiment, we also vary the lookahead length (k) between 2 and 8.

4.2.1.1 Q-Learning agent

We use the standard Q-Learning update rule [24]:

$$Q(S,A) \leftarrow Q(S,A) + \alpha[R_{t+1} + \gamma \max_{A'} Q(S',A') - Q(S,A)] \tag{4.1}$$

As well as the algorithm 1.

Algorithm 1 Q-Learning

$Q(S,A) \leftarrow 0$ for all $S \in \mathcal{S}$ and $A \in \mathcal{A}$
for each episode **do**
    $S \leftarrow$ sample starting state
    while not done **do**
        $A \leftarrow \text{argmax}_{A'} Q(S,A')$
        $S', R \leftarrow$ perform A in state S
        $Q(S,A) \leftarrow Q(S,A) + \alpha[R + \gamma \max_{A'} Q(S',A') - Q(S,A)]$
        $S \leftarrow S'$
    end while
end for

4.2.1.2 Conditional lookahead agent

For the agent using (Eq. 3.4) the naive conditional future-value function, we apply the update rule (given the state S_t and action sequence $A_{t:t+k}$):

$$D(S_t, A_{t:t+k}) \leftarrow D(S_t, A_{t:t+k}) + \alpha[\sum_{i=0}^{k-1} \gamma^i R_{t+i} + \max_{A_{t+k:t+2k}} D(S_{t+k}, A_{t+k:t+2k}) - D(S_t, A_{t:t+k})] \tag{4.2}$$

We use algorithm 2. The algorithm is similar to algorithm 1, but uses the eq 4.2 for updates and also keeps the state, action, and rewards in a queue to perform the updates on the sequence.

Algorithm 2 Conditional lookahead agent

$D(S, A_{0:K}) \leftarrow 0$ for all $S \in \mathcal{S}$ and $A_{0:K} \in \mathcal{A}^K$
Initialise empty state, action, and reward queue L_S, L_A, L_R
for each episode **do**
    $S \leftarrow$ sample starting state
    while not done **do**
        $A \leftarrow \text{argmax}_{A_{0:1}} \max_{A_{1:K}} D(S, A_{0:K})$
        $S', R \leftarrow$ perform A in state S
        append S, A, R to L_S, L_A, L_R
        if len(L_S) = k **then**
            $D(L_S[0], L_A) \leftarrow D(L_S[0], L_A) + \alpha[\sum_{i=0}^{K-1} \gamma^i L_R[i] + \max_{A_{0:K}} D(S', A_{0:K}) - D(L_S[0], L_A)]$
            Pop first element from L_S, L_A, L_R
        end if
        $S \leftarrow S'$
    end while
end for

4.2.1.3 Causal lookahead agent

For the causal lookahead agent we use the causally derived future-value function (Eq. 3.8) to construct the update rule (Eq. 4.3). Here it is worth noting that the causal future-value function relies on a shorter time horizon version of itself. In reality, we could maintain k different future-value functions of different horizons, but for simplicity, we have assumed that the shorter horizon future-value function can be approximated by greedifying the last step (i.e. $D(S,A_{0:k-1}) = \max_{A_{k-1}} D(S,A_{0:k})$). While this assumption is not necessarily true, preliminary experiments have shown this to work well.

$$D(S_t,A_{t:t+k}) \leftarrow D(S_t,A_{t:t+k}) + \alpha[R_t + \gamma \max_{A_{t+k+1}} D(S_{t+1},A_{t+1:t+k+1}) - D(S_t,A_{t:t+k})] \tag{4.3}$$

We then use a modified version of the Q-Learning algorithm (Alg. 1) to update the future value function. This algorithm is outlined in algorithm 3.

Algorithm 3 Causal lookahead agent

$D(S,A_{0:K}) \leftarrow 0$ for all $S \in \mathcal{S}$ and $A_{0:K} \in \mathcal{A}^K$
for each episode **do**
    $S \leftarrow$ sample starting state
    while not done **do**
        $A_0 \leftarrow \text{argmax}_{A'_0} \max_{A'_{1:K}} D(S,A'_{0:K})$
        $S', R \leftarrow$ perform A_0 in state S
        $D(S,A_{0:k}) \leftarrow D(S,A_{0:k}) + \alpha[R_t + \gamma \max_{A_{t+k}} D(S_{t+1},A_{1:k+1}) - D(S,A_{0:k})]$ for $A_{1:K} \in \mathcal{A}^{K-1}$
        $S \leftarrow S'$
    end while
end for

Further hyper-parameters can be found in Appendix A.

4.2.2 Results

In Figure 4.3 we can see the training results of the different agents in the randomly generated MDPs. We can see in the results that as the lookahead horizon increases the performance of the conditional estimate agent degrades, while the causal estimate agent retains close to the optimal Q-Learning results, showing that the stochasticity introduced by the MDP has little effect on the overall performance.

We can see how the expected result of the conditional agent overshoots the optimal Q-Learning result by a large margin, as expected, and underperforms in realised reward. This clearly shows how the overestimation misleads the greedy policy.

We can also see that the expected value of the causal agent is significantly lower than the Q-Learning result. This is likely due to the underestimation problem discussed in Section 3.7. The more interesting, and unexpected result, is that even though the future-value function underestimates the possible rewards, this has very little effect on the actual return of the agent, and the agent achieves close to the optimal Q-Learning result regardless.

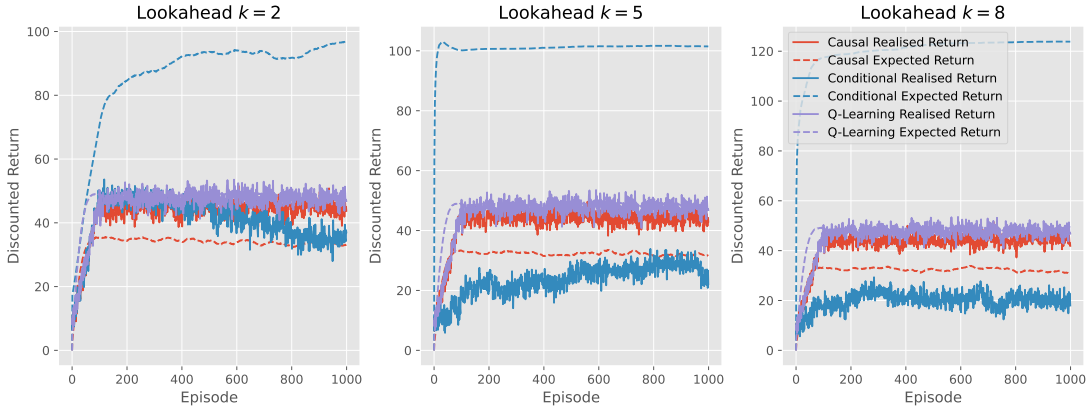

Figure 4.3: Training results from training the different agents on randomly generated MDPs.

4.3 Windy River Environment

So far we have looked at highly contrived MDPs as well as random MDPs in which we see some degrading performance. In this experiment, we will look at how under certain situations this over-estimation can lead to complete failures to learn. To demonstrate this we have created an environment inspired by the CliffWalking environment [22].

4.3.1 Method

The Windy River environment consists of a grid world (See Fig. 4.4) where the agent starts in the lower left corner and the goal is to move to the lower right corner. Along the bottom, there is a river, which if you fall in you are taken back to the start. The agent has 4 actions, up, down, left, and right. At each step in the environment, there is a probability of wind, which if it occurs, your action will be ignored and you will be blown one grid cell down, towards the river. The agent is given a reward of -1 for every step in the environment.

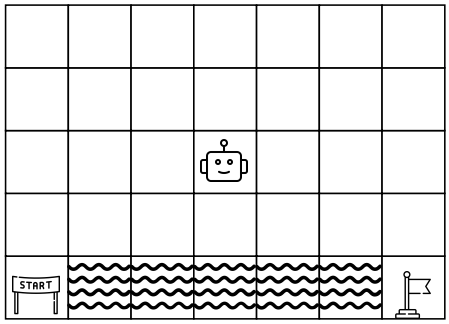

Figure 4.4: An illustration of the Windy River environment

The method is the same as in the previous experiment (Section 4.2), except that we do not vary the lookahead length. In this experiment we very the wind probability, to get insight into how the amount of stochasticity affects the different agents. The results are averaged over 10 runs with different seeds.

Further hyper-parameters can be found in Appendix A.

4.3.2 Results

In Figure 4.5 we can see the results from the Windy River environment. The results are similar to the random MDP results, with Q-Learning and the causal agent achieving a similar performance, while the conditional agent has the highest estimate with the lowest performance. We can also see that when the probability of wind is raised to 50% the conditional agent is unable to learn anything.

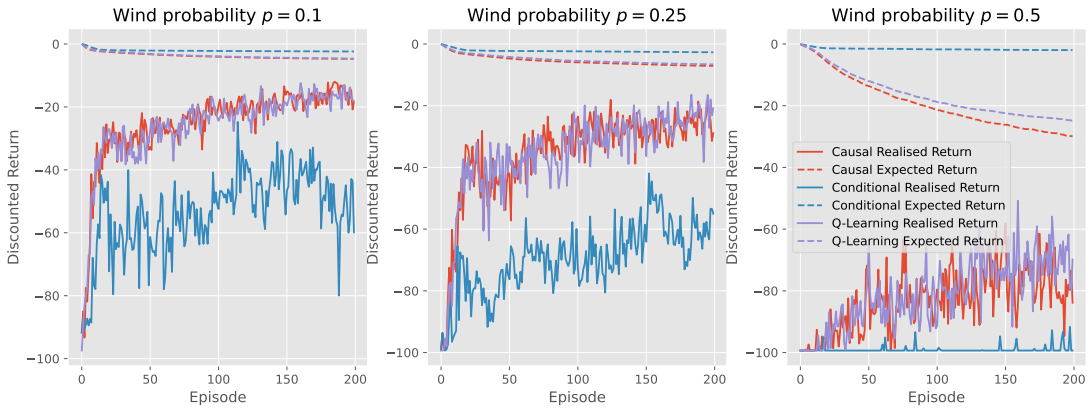

Figure 4.5: Results from the Windy River environment with varying wind probabilities.

Chapter 5

Discussion

The theoretical and empirical findings of this research underscore the utility of incorporating causal inference into reinforcement learning, particularly in the context of planning in stochastic environments. The core of the investigation revolved around the limitations of traditional conditional planning methods, which often rely on maximizing expected returns based on conditional probabilities. This approach, while computationally feasible and effective in many deterministic or near-deterministic scenarios, tends to overestimate future rewards in stochastic settings due to the failure to differentiate between correlation and causation.

5.1 Implications of Overestimation and Underestimation

The identified overestimation problem is particularly problematic in environments where certain actions are correlated with high rewards without being causally responsible for them. This misalignment leads to suboptimal decision-making, as agents may be led astray by spurious correlations that do not reflect the true dynamics of the environment. The empirical evidence from randomly generated MDPs and the Windy River environment highlights how this can lead to degraded performance or even catastrophic failures in highly stochastic scenarios.

Conversely, the introduction of a causally-informed future-value function addresses this overestimation by utilizing the do-operator to derive an estimate that correctly accounts for the causal dependencies in multi-step planning. However, this correction introduces a new challenge: underestimation. The underestimation is upper-bounded by the optimal plan, which can be lower than the optimal policy. While avoiding the pitfalls of overestimation, may lead to conservative strategies that do not fully exploit

the available opportunities in the environment.

5.2 Addressing the Underestimation with Planning Projections

To counteract the underestimation, the research proposes a state-dependent policy transformation using planning projection matrices. This approach transforms the action space such that the future-value function better approximates the optimal value function. The theoretical guarantee that an appropriate planning projection can align the future-value function with the optimal value function is promising, offering a pathway to more accurate and reliable planning strategies.

However, the practical implementation of such planning projections introduces complexity. The need to balance exploration and exploitation presents challenges that require further exploration. Moreover, the potential existence of local optima in the iterative improvement of both policy and planning projection suggests that additional research is needed to develop robust convergence guarantees.

5.3 Broader Impacts and Future Work

The integration of causal inference into decision-making processes has the potential to improve the reliability and robustness of AI systems in various domains, including robotics, autonomous systems, and healthcare. By moving beyond mere correlation and utilising causality, AI systems can make decisions that are more aligned with the underlying realities of the environments they operate in, without requiring to accurately model the environment itself.

Future work should focus on empirically validating the proposed methods in more complex and realistic environments. Additionally, the development of efficient algorithms for learning state-dependent planning projections can prove to be an interesting research avenue. Exploring how these techniques can be scaled to high-dimensional state and action spaces, possibly through the use of deep learning, could significantly enhance their applicability in real-world scenarios.

5.4 Conclusion

This thesis has explored the intersection of causal inference and reinforcement learning, with a particular focus on improving planning in stochastic environments. The research identifies critical limitations in traditional conditional planning methods, which tend to overestimate expected returns due to a failure to distinguish between correlation and causation. By incorporating the do-operator into the planning process, the research introduces a causally informed future-value function that corrects for these overestimations, albeit at the cost of introducing sarcastically conservative estimates.

To mitigate the underestimation problem, the thesis proposes the use of state-dependent planning projections, which can transform the action space to better align with the optimal value function. This approach holds promise for improving the accuracy and reliability of planning in reinforcement learning, particularly in environments characterized by significant stochasticity.

Overall, the findings of this research contribute to the ongoing development of more intelligent and reliable AI systems. By integrating causal reasoning into reinforcement learning, we move closer to the goal of creating agents that can make more informed and effective decisions, even in the face of uncertainty and complexity. Future research should continue to refine these methods and explore their application in increasingly challenging and dynamic environments.

Bibliography

[1] M Mehdi Afsar, Trafford Crump, and Behrouz Far. Reinforcement learning based recommender systems: A survey. *ACM Computing Surveys*, 55(7):1–38, 2022.

[2] Ilge Akkaya, Marcin Andrychowicz, Maciek Chociej, Mateusz Litwin, Bob McGrew, Arthur Petron, Alex Paino, Matthias Plappert, Glenn Powell, Raphael Ribas, et al. Solving rubik's cube with a robot hand. *arXiv preprint arXiv:1910.07113*, 2019.

[3] Ioannis Antonoglou, Julian Schrittwieser, Sherjil Ozair, Thomas K Hubert, and David Silver. Planning in stochastic environments with a learned model. In *International Conference on Learning Representations*, 2021.

[4] Silvia Chiappa, Sébastien Racaniere, Daan Wierstra, and Shakir Mohamed. Recurrent environment simulators. *arXiv preprint arXiv:1704.02254*, 2017.

[5] Rémi Coulom. Efficient selectivity and backup operators in monte-carlo tree search. In *International conference on computers and games*, pages 72–83. Springer, 2006.

[6] Arthur Guez, Mehdi Mirza, Karol Gregor, Rishabh Kabra, Sébastien Racanière, Théophane Weber, David Raposo, Adam Santoro, Laurent Orseau, Tom Eccles, et al. An investigation of model-free planning. In *International Conference on Machine Learning*, pages 2464–2473. PMLR, 2019.

[7] Danijar Hafner, Timothy Lillicrap, Mohammad Norouzi, and Jimmy Ba. Mastering atari with discrete world models. *arXiv preprint arXiv:2010.02193*, 2020.

[8] Lukasz Kaiser, Mohammad Babaeizadeh, Piotr Milos, Blazej Osinski, Roy H Campbell, Konrad Czechowski, Dumitru Erhan, Chelsea Finn, Piotr Kozakowski, Sergey Levine, et al. Model-based reinforcement learning for atari. *arXiv preprint arXiv:1903.00374*, 2019.

[9] B Ravi Kiran, Ibrahim Sobh, Victor Talpaert, Patrick Mannion, Ahmad A Al Sallab, Senthil Yogamani, and Patrick Pérez. Deep reinforcement learning for autonomous driving: A survey. *IEEE Transactions on Intelligent Transportation Systems*, 23(6):4909–4926, 2021.

[10] Levente Kocsis and Csaba Szepesvári. Bandit based monte-carlo planning. In *European conference on machine learning*, pages 282–293. Springer, 2006.

[11] V Mnih. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*, 2013.

[12] 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. PMLR, 2016.

[13] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *nature*, 518(7540):529–533, 2015.

[14] Junhyuk Oh, Xiaoxiao Guo, Honglak Lee, Richard L Lewis, and Satinder Singh. Action-conditional video prediction using deep networks in atari games. *Advances in neural information processing systems*, 28, 2015.

[15] Judea Pearl. Causal inference in statistics: An overview. 2009.

[16] Judea Pearl. The do-calculus revisited. *arXiv preprint arXiv:1210.4852*, 2012.

[17] Julian Schrittwieser, Ioannis Antonoglou, Thomas Hubert, Karen Simonyan, Laurent Sifre, Simon Schmitt, Arthur Guez, Edward Lockhart, Demis Hassabis, Thore Graepel, et al. Mastering atari, go, chess and shogi by planning with a learned model. *Nature*, 588(7839):604–609, 2020.

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

[19] David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al. Mastering the game of go with deep neural networks and tree search. *nature*, 529(7587):484–489, 2016.

[20] David Silver, Thomas Hubert, Julian Schrittwieser, Ioannis Antonoglou, Matthew Lai, Arthur Guez, Marc Lanctot, Laurent Sifre, Dharshan Kumaran, Thore Graepel, et al. Mastering chess and shogi by self-play with a general reinforcement learning algorithm. *arXiv preprint arXiv:1712.01815*, 2017.

[21] Richard S Sutton. Dyna, an integrated architecture for learning, planning, and reacting. *ACM Sigart Bulletin*, 2(4):160–163, 1991.

[22] Richard S Sutton. Reinforcement learning: an introduction. *A Bradford Book*, 2018.

[23] Richard S Sutton, David McAllester, Satinder Singh, and Yishay Mansour. Policy gradient methods for reinforcement learning with function approximation. *Advances in neural information processing systems*, 12, 1999.

[24] Christopher JCH Watkins and Peter Dayan. Q-learning. *Machine learning*, 8:279–292, 1992.

[25] Weirui Ye, Shaohuai Liu, Thanard Kurutach, Pieter Abbeel, and Yang Gao. Mastering atari games with limited data. *Advances in neural information processing systems*, 34:25476–25488, 2021.

Appendix A

Experiment Hyper-parameters

A.1 Conditional vs Causal Estimates in Random MDPs

γ-decay	0.99
Experiments	4
MDP State count	100
MDP Action count	4

Table A.1: Hyper-parameters for the ”Conditional vs Causal Estimates in Random MDPs” experiment

A.2 Conditional vs Causal Agents in Random MDPs

γ-decay	0.99
Random MDPs	10
MDP State count	10
MDP Action count	4

Table A.2: Hyper-parameters for the ”Conditional vs Causal Agents in Random MDPs” experiment

A.3 Conditional vs Causal Agents in Random MDPs

γ-decay	0.99
Experiment Seeds	10

Table A.3: Hyper-parameters for the ”Conditional vs Causal Agents in Random MDPs” experiment