Lost in abstraction

Exploring our way to efficient reinforcement learning

Starre, R.A.N.

DOI
10.4233/uuid:4b156db1-bdb9-4c1a-b704-5befe072c2cf

Publication date
2025

Document Version
Final published version

Citation (APA)
Starre, R. A. N. (2025). *Lost in abstraction: Exploring our way to efficient reinforcement learning*. [Dissertation (TU Delft), Delft University of Technology]. https://doi.org/10.4233/uuid:4b156db1-bdb9-4c1a-b704-5befe072c2cf

Lost in Abstraction

Exploring Our Way to Efficient Reinforcement Learning

Rolf A. N. Starre

Lost in Abstraction

Exploring Our Way to Efficient Reinforcement Learning

LOST IN ABSTRACTION

EXPLORING OUR WAY TO EFFICIENT REINFORCEMENT LEARNING

Dissertation

for the purpose of obtaining the degree of doctor
at Delft University of Technology
by the authority of the Rector Magnificus, prof. dr. ir. T.H.J.J. van der Hagen,
chair of the Board for Doctorates
to be defended publicly on Tuesday 11, November 2025 at 15:00 o'clock

by

Rolf Andries Nicodemo STARRE

Master of Science in Computer Science,
Delft University of Technology, the Netherlands,
Master of Science in Human Movement Sciences,
Vrije Universiteit Amsterdam, the Netherlands,
born in Leiden, the Netherlands.

This dissertation has been approved by the promotors.

Composition of the doctoral committee:

Rector Magnificus,	chairperson
Dr. F.A. Oliehoek,	Delft University of Technology, *promotor*
Prof.dr. M. Loog,	TU Delft / Radboud U. Nijmegen, *promotor*
Independent members:	
Prof. dr. M.T.J. Spaan	Delft University of Technology
Dr.-Ing J. Kober	Delft University of Technology
Dr. J.S.L. Junges	Radboud U. Nijmegen
Dr. C. Amato	Northeastern University, US
Prof. dr. M.M. de Weerdt,	Delft University of Technology, reserve member

Cover by: S.F.M. Starre

ISBN 978-94-6518-145-5

An electronic copy of this dissertation is available at
https://repository.tudelft.nl/.

All that other folk can do,
Why, with patience, should not you?
Only keep this rule in view,
Try, try again.

William Edward Hickson

CONTENTS

SUMMARY

Reinforcement Learning (RL) methods aim to find near-optimal solutions to sequential decision-making problems with initially unknown dynamics. These methods learn by interacting with the environment and observing the outcomes of their actions. RL methods have made significant progress in recent years and good solutions to difficult problems have been found in rapid succession. However, these successes often rely on access to a simulator, which makes it possible to generate a lot of experience cheaply and safely. In contrast, there are many real-world applications of RL where learning must occur solely through experience obtained in the environment itself. This is often time-consuming and expensive, with risks such as damage to equipment. This makes efficiently collecting and using experience of crucial importance. The thesis focuses on improving the learning efficiency of RL methods.

Two methods to improve learning efficiency are Model-based Reinforcement Learning (MBRL) and state abstraction. MBRL methods learn a model and use it for planning and learning, which drives efficient learning by directing exploration to unknown areas of a problem. On the other hand, state abstraction reduces the size of a problem, which achieves efficient learning in an alternative way.

This thesis focuses on combining these two methods, aiming to achieve even greater learning efficiency. We first survey methods that have previously combined MBRL and abstraction, including approaches ranging from state aggregation to abstractions based on deep learning. We identify challenges resulting from the combination of MBRL and abstraction, particularly focusing on the view of RL plus abstraction as a partially observable problem. From this perspective, we demonstrate how this combination leads to *perceptual aliasing*, where different states are perceived as the same state. This implies the observed behavior is no longer guaranteed to adhere to the assumptions required for most analyses.

Next, this thesis addresses the issue of perceptual aliasing with a theoretical analysis of the combination of MBRL and abstracted observations. While there are many algorithms with performance guarantees without abstraction, it may come as a surprise that no such guarantees are available when combining MBRL and abstraction, where MBRL merely observes abstract states. We prove that, even in this context, it is still possible to guarantee that an accurate model can be learned. Based on this result, we extend the performance guarantees of MBRL methods to learning with abstract observations.

Finally, we shift our focus to partially observable problems. Previously, we assumed the problems were fully observable and it was only the abstraction that rendered them partially observable. However, many complex problems are partially observable by nature. A difficulty in these problems is the belief space the agent needs to reason about, which is typically too large to find an exact solution. Online planning, which involves choosing actions within a limited amount of time, is often used as an alternative for find-

ing solutions. In this setting, abstraction can provide additional benefits by potentially increasing the planning speed, since it reduces the size of the model.

We propose and investigate an abstraction method that uses the structure of the problem to define different levels of abstraction. We evaluate our approach empirically in several domains and find that abstract models can indeed enable faster planning which can increase performance, even when the abstraction leads to a loss of information. Further, we show that abstractions can improve performance even under a fixed number of simulations. This occurs because abstract models can aggregate multiple samples that the original model treats independently, thereby using experience more efficiently.

This thesis theoretically and empirically shows that we can learn efficiently by combining MBRL and abstraction. The results of this investigation advance our understanding of this combination, furthering knowledge in this important area of research and providing a foundation that can support effective learning in complex real-world problems.

SAMENVATTING

Het doel van Reinforcement Learning (RL) methodes is om bijna-optimale oplossingen te vinden voor sequentiële besluitvormingsproblemen met aanvankelijk onbekende dynamica. Deze methodes leren door hun interactie met de omgeving en het observeren van de resultaten. RL methodes hebben de laatste jaren aanzienlijke vooruitgang geboekt en goede oplossingen voor moeilijke problemen werden snel na elkaar gevonden. Deze successen zijn echter vaak afhankelijk van toegang tot een simulator, waardoor het mogelijk is om op een goedkope en veilige manier veel ervaring op te doen. Daarentegen zijn er veel toepassingen van RL in de echte wereld waarbij het leren uitsluitend moet plaatsvinden op basis van ervaring die direct in de omgeving wordt opgedaan. Dit is vaak tijdrovend en duur, met risico's zoals schade aan apparatuur. Dit maakt het efficiënt verzamelen en gebruiken van ervaring van cruciaal belang. Deze dissertatie richt zich op het verbeteren van de leerefficiëntie van RL methodes.

Twee methodes om de leerefficiëntie te verbeteren zijn Model-based Reinforcement Learning (MBRL) en toestandsabstractie. MBRL methodes leren een model en gebruiken dit voor planning en leren, wat efficiënt leren stimuleert door exploratie te richten op onbekende gebieden van een probleem. Aan de andere kant verkleint toestandsabstractie de omvang van een probleem, waardoor efficiënt leren op een andere manier wordt bereikt.

Deze dissertatie richt zich op het combineren van deze twee methodes, met als doel een nog grotere leerefficiëntie te bereiken. We geven eerst een overzicht van methodes die eerder MBRL en abstractie hebben gecombineerd, inclusief benaderingen die variëren van toestandsaggregatie tot abstracties gebaseerd op diep leren. We identificeren uitdagingen die voortkomen uit de combinatie van MBRL en abstractie, waarbij we ons vooral richten op de kijk op MBRL plus abstractie als een gedeeltelijk waarneembaar probleem. Vanuit dit perspectief laten we zien hoe deze combinatie leidt tot *perceptuele aliasing*, waarbij verschillende toestanden worden waargenomen als dezelfde toestand. Dit houdt in dat het waargenomen gedrag niet langer gegarandeerd voldoet aan de aannames die nodig zijn voor de meeste analyses.

Vervolgens wordt in dit proefschrift het probleem van perceptuele aliasing aangepakt met een theoretische analyse van de combinatie van MBRL en geabstraheerde waarnemingen. Hoewel er veel algoritmen zijn met prestatiegaranties zonder abstractie, kan het als een verrassing komen dat zulke garanties niet beschikbaar zijn bij het combineren van MBRL en abstractie, waarbij MBRL alleen abstracte toestanden observeert. We bewijzen dat het zelfs in deze context nog steeds mogelijk is om te garanderen dat er een accuraat model geleerd kan worden. Op basis van dit resultaat breiden we de prestatiegaranties van MBRL methodes uit naar leren met abstracte waarnemingen.

Tot slot verschuiven we onze aandacht naar gedeeltelijk waarneembare problemen. Eerder gingen we ervan uit dat de problemen volledig waarneembaar waren en dat het

alleen de abstractie was die ze gedeeltelijk waarneembaar maakte. Veel complexe problemen zijn echter van nature gedeeltelijk waarneembaar. Een moeilijkheid bij deze problemen is de enorme hoeveelheid mogelijkheden waarover de agent moet redeneren, die meestal te groot is om een exacte oplossing te vinden. Online planning, waarbij acties worden gekozen binnen een beperkte tijd, wordt vaak gebruikt als alternatief voor het vinden van oplossingen. In deze setting kan abstractie extra voordelen bieden door mogelijk de planningssnelheid te verhogen, omdat het de grootte van het model verkleint.

We creëren en onderzoeken een abstractiemethode die de structuur van het probleem gebruikt om verschillende abstractieniveaus te definiëren. We evalueren onze aanpak empirisch in verschillende domeinen en bevinden dat abstracte modellen inderdaad snellere planning mogelijk maken, wat de prestaties kan verhogen, zelfs als de abstractie leidt tot een verlies van informatie. Verder laten we zien dat abstracties de prestaties kunnen verbeteren, zelfs bij een vast aantal simulaties. Dit komt doordat abstracte modellen meerdere ervaringen kunnen samenvoegen die het oorspronkelijke model onafhankelijk behandelt, waardoor ervaring efficiënter wordt gebruikt.

Dit proefschrift toont zowel theoretisch als empirisch aan dat we efficiënt kunnen leren door MBRL te combineren met abstractie. De resultaten van dit onderzoek vergroten ons begrip van deze combinatie, verdiepen de kennis op dit belangrijke onderzoeksgebied en bieden een basis die effectief leren in complexe toepassingen in de echte wereld kan ondersteunen.

1

INTRODUCTION

Intelligence can be defined as the ability to learn, understand, or adapt to new or challenging situations [1]. In artificial intelligence, Reinforcement Learning (RL) embodies this concept by enabling systems to discover optimal solutions autonomously. I find this notion particularly fascinating, and recent advancements have demonstrated that RL can achieve performance levels surpassing those of top human players in strategic games like chess, Go, Starcraft, and Stratego [2–5]. In a practical application, researchers have employed RL to improve data center cooling, reducing costs and energy consumption [6].

Looking to the future, RL holds promise for addressing challenges such as space exploration. The vast distances involved create lengthy travel times, making human exploration impractical. Moreover, the significant delays in communication mean that robots must autonomously adapt to unforeseen challenges, such as a malfunctioning motor or a sudden sandstorm. In such scenarios, the ability to learn and adapt is crucial, underscoring the potential of RL.

A key challenge in RL is the trade-off between exploration and exploitation [7]. Exploration is crucial in gaining the knowledge required for employing near-optimal solutions. In real-world problems, RL methods must be efficient in terms of the experience they require, as experience can be costly in terms of time and money. However, current methods are often not provably efficient or rely on simulators that may not be available in unknown environments [2–6, 8–12].

This thesis investigates two methods and their integration to enhance the efficiency of RL. We focus on Model-based Reinforcement Learning (MBRL), which learns and utilizes a model to guide exploration [13, 14], and state abstraction, which creates compact problem representations for faster learning [15, 16]. Most importantly, we explore how combining these methods can result in more efficient learning. In the following sections, we give an overview of these techniques and the challenges associated with their integration. We assume the reader is familiar with the main principles of RL and planning, such as the Markov assumption, Markov decision processs (MDPs), (optimal) policies [7, 17, 18], and Partially Observable Markov Decision Processes (POMDPs) [19].

1

1.1. REINFORCEMENT LEARNING

We are interested addressing sequential decision-making problems through RL. As the term suggests, these problems require a sequence of actions to reach a desired outcome. For example, consider the example in Figure 1.1. Here, the goal is for an exhausted researcher to receive coffee. To achieve this, the robot must navigate towards the researcher and hand over the coffee. It can take several actions, such as moving right or left and handing over the coffee. Here it can accomplish its goal by moving right twice and then handing over the coffee.

Of course, this is a simple example. Real-world problems are often more complex and have additional challenges such as stochasticity. Stochasticity is often present as randomness in the outcomes of actions. In the coffee robot example, this could mean that an attempt to move right only succeeds 70% of the time, for instance, due to the difficulty of navigating the floor surface.

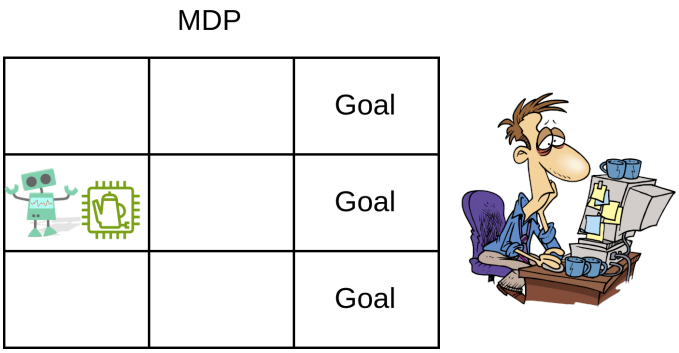


Figure 1.1: A robot that has to bring coffee to the vicinity of an exhausted researcher.

Due to the stochastic nature of most problems, the solution to these problems is typically not simply a fixed sequence of actions. Instead, solutions specify the best action to take in any given situation that could arise. This kind of solution is called a *policy*. For example, in the case of the coffee robot, For the coffee robot, a policy might state: move right if the robot is in one of the first two columns, and hand over the coffee if it is in the third column. Two cases can be distinguished when finding policies for sequential decision-making problems.

The first case concerns applications where we know how the world works and thus can construct or estimate a sufficiently accurate model. In this scenario, *planning* methods can be applied to find the best policy for the problem. Planning uses the model to compute a policy before taking any real-world actions.

The second case applies to problems where we do not know how the world works, and thus do not have access to a model of the problem. We can turn to *RL* methods to find a policy in these situations. This approach requires learning (the dynamics of the problem) through interaction with the real world. Learning happens through trying out actions and observing the outcomes.

RL has made significant progress in the last years, particularly in Atari games [8],

and games like chess [3], poker [10], and other simulation-based problems [9]. More difficult problems with a huge state space are being mastered all the time, for example, the games Unreal [11], Dota [12] and Stratego [5]. However, these problems also have a simulator, meaning that generating a lot of experience is not a problem. The created methods require a lot of experience to learn, which could be challenging in the real world for several reasons: the number of different situations that could be encountered might be huge, collecting experience can be time-consuming, and it might be expensive (equipment could break). Thus, being efficient is important in RL.

In RL, the objective is to learn efficiently and quickly find a good policy. This task is difficult because, initially, the environment is unknown. The challenge of learning to act optimally with minimal experience can be illustrated through the example of the coffee robot. At first, the robot will not know the outcomes of any actions it can choose and must experiment to observe the outcomes. Imagine that the first path the robot finds is the one in Figure 1.2. While this path achieves its goal of delivering coffee to the tired researcher, it is suboptimal since the researcher would like his coffee as quickly as possible (time is often an important component).

For instance, moving to the right from the starting position would be faster, but the robot is unaware of this possibility. It has to try out the action first to observe the outcome. This also means that the robot does not know whether or not there is any faster route, as it has only followed the path in Figure 1.2. When the robot starts again from the initial position, it must choose whether to exploit its current knowledge by following the path it has learned will lead to the goal (though this may not always succeed due to stochasticity), or to explore by trying out alternative actions in order to learn more and possibly discover a better policy.

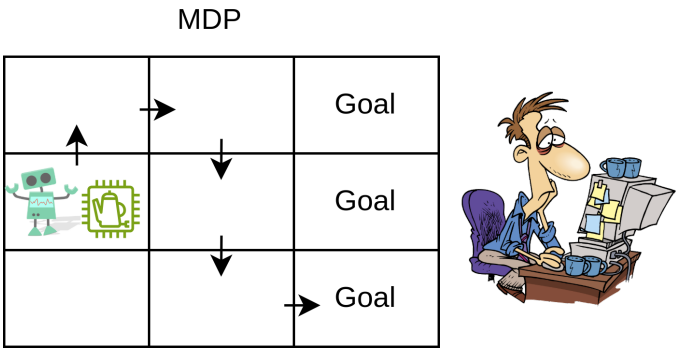


Figure 1.2: Example initial path followed by the robot.

Having to balance visiting the states and taking the actions you believe are most rewarding (exploiting) with learning about what you do not know (exploration) is called the exploration-exploitation trade-off, and is one of the main challenges in RL [7]. As illustrated by the example, exploring generally comes at the expense of exploitation, and vice versa. Finding the right balance is important in RL, especially

for efficient learning.

There are multiple definitions of efficient learning in RL [14, 20]. Efficiency is typically measured in terms of *sample efficiency*, related to the number of samples required to learn, and *computational efficiency*, related to the amount of storage and computation required. In some cases, we might be concerned with only one of these aspects. For example, if a fast simulator of the problem is available, generating samples might be very cheap, making computational efficiency the primary concern. In real-world settings, such as operating with robots, collecting many samples can be time-consuming or expensive, so sample efficiency may be the priority.

1.2. MODEL-BASED RL AND EXPLORATION

A way to learn efficiently is to maintain a (learned) model of the environment along with a measure of uncertainty about this model, as this allows for balancing exploration and exploitation. When you learn a model of the environment, you can use it to plan to reach any state you want. At the same time, you can also use it to take the actions you believe are most rewarding. This is the idea on which MBRL methods are based. They try to learn an accurate model of the environment, which allows these methods to do targeted exploration. They can purposefully take actions that have not been taken yet, or attempt to reach a certain state of a problem that has not been visited often, to explore that state.

For example, the coffee robot from the earlier example could keep track of the outcomes of its actions in each state while following the path in Figure 1.2. When it starts again from the beginning, the robot can prioritize actions it has not tried before. This targeted exploration helps the robot learn more about the environment and more quickly discover a better policy.

Exploration methods are important for efficiently navigating and learning within an environment, and several near-optimal methods for exploration in MBRL have been developed [13, 14, 20–25]. In this thesis, we focus on two such methods: interval estimation [14] and Bayesian learning [21, 23]. Both maintain a model of the environment together with a measure of uncertainty around that model, and use this uncertainty to guide action selection. These methods will be discussed in more detail in Chapters 3 and 4.

A limitation of the mentioned exploration methods is that they typically require visiting all the states, which makes them impractical for large problems. This is both because exploration can become too time-consuming, and because the learned model could exceed memory capacity. Most of the methods discussed above relate to tabular methods. In model-based RL, tabular methods are methods were we store the whole model in memory. However, as problems grow larger this might not be feasible. It might simply not be possible because it does not fit into the memory, or finding a solution will take too long. Two alternatives to tabular learning are learning factored representations and deep RL, which will be discussed in Chapters 2 and 4.

1.3. STATE ABSTRACTION

We are interested in finding other ways to learn in large problems. One such alternative is *abstraction*. There are many forms of abstraction in reinforcement learning. For a recent survey, see [26]. Key categories include temporal abstraction [27–29], action abstraction [30, 31], influence-based abstraction [32, 33], and state abstraction [15, 16, 34, 35]. These approaches generally remove (or abstract away) information from the environment, particularly information that is not (as) relevant for finding the optimal policy. For example, depending on the type of abstraction, the removed information could relate to actions, state information, or other components of the MDP. We focus on state abstraction [15, 16], which partitions the state space by grouping states into abstract states. We present a high-level overview here and provide a more detailed description in the following chapters.

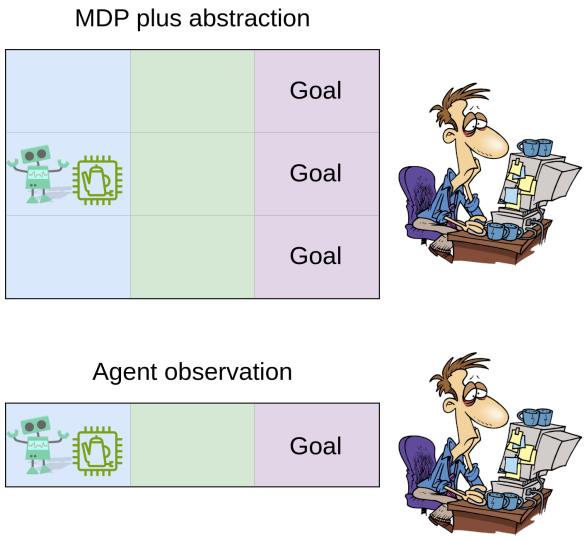


Figure 1.3: The coffee robot problem, with abstraction. A representation of the problem and the view of the robot.

As an example of abstraction, consider the coffee robot problem shown in Figure 1.1 on page 2. In this case, an abstract representation can be created by removing information about the robot's row position, as illustrated in Figure 1.3. Using this representation, the robot cannot distinguish the row it is in. This is no issue since the optimal action is to move right and then deliver the coffee, regardless of the row the robot is in. Importantly, the abstraction reduces the size of the problem from nine distinct states to only three. This makes the problem significantly easier since it reduces the amount of states the robot needs to learn the dynamics of to find the optimal policy.

As the example shows, abstraction can reduce the size of the state space. Many

different types of state abstraction have been studied, see [15] for an excellent survey. In state abstractions, states in the MDP are grouped, based on some criteria, creating an abstract MDP.

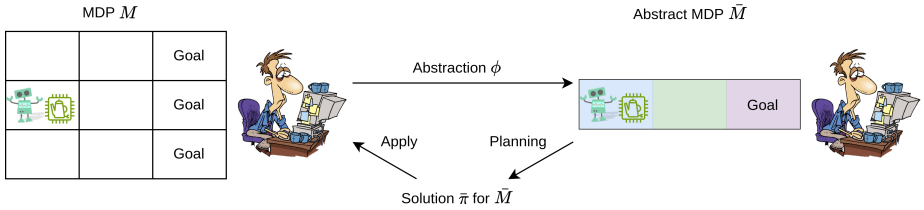


Figure 1.4: Illustrating the idea of applying a solution for an abstract MDP to the original problem. Image inspired by [36].

Ultimately, we want a policy that performs well in the original problem. This idea is displayed in Figure 1.4. First, we construct an abstract model by using the abstraction function, and then we compute an optimal policy in this abstract model. When this policy is applied to the original problem, some value may be lost, as the abstraction is often only an approximate representation of the dynamics. Nevertheless, for several classes of approximate abstractions, this loss can be bounded and depends on the *quality* of the abstraction [16]. The quality of an abstraction depends on the type of state abstraction and a parameter that measures the closeness of the approximation to the true model. The closer the approximation is to the real model, the closer one can get to the optimal solution. Conversely, when the abstraction is far away from the real model, there is no guarantee of learning anything useful.

A challenge arises from the use of abstraction when the abstract model is not close to the true model. This issue can be illustrated by the example in Figure 1.5, a modified version of the examples in Figures 1.1 and 1.3. In the previous setup, there was no difference between the rows. In the current version, the middle location is blocked by a trashcan, making it impossible for the robot to pass through. Instead, the robot has to navigate around it to reach the goal. If the robot uses the abstract representation it only knows at which abstract location (the column) it is. This means it cannot distinguish whether it is in the top, middle, or bottom row. Without the information about which row it is in, it is difficult for the robot to navigate around this blockade. This demonstrates how abstraction can result in the loss of information important for estimating the exact dynamics of a problem.

1.4. CHALLENGES IN COMBINING ABSTRACTION WITH MBRL

Both abstraction and MBRL can improve learning efficiency. Our goal is to combine these methods to increase learning efficiency. Specifically, we aim to optimally balance exploration and exploitation through quickly learning a compact abstract

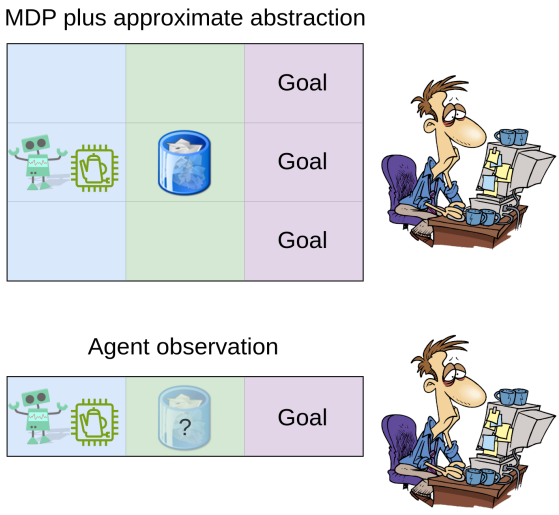


Figure 1.5: The coffee robot problem, with approximate abstraction. A representation of the problem and the view of the robot.

model. We call this combination MBRL from Abstracted Observations (MBRLAO), where the agent acts in an MDP but receives observations of the abstract states instead of the regular states. This combination is not trivial, and a lot of work has been done that has combined abstraction and RL in some shape or form [34, 35, 37–42].

To illustrate the difficulties arising from this combination, consider a simple case in which we aim to find a solution in an MDP and are provided with an abstraction function. An MBRL algorithm could use this abstraction function to observe abstract states instead of raw states. However, as illustrated in Section 1.3, this approach can make learning the problem much more difficult. As a result of using the abstraction function, the efficiency of the MBRL methods may be reduced in this setting, as they typically rely on the *Markov* assumption [13, 14, 20]. The Markov assumption states that the transition from the current state to the next one depends only on the current state and the chosen action. However, this assumption may no longer hold in MBRLAO. This raises an important question: *is it possible to directly transfer the results for MBRL without abstraction to MBRLAO?*

So far, we have focused on the situation where an abstraction is *given*, meaning we use an existing abstraction during the learning process without creating or learning one ourselves. A good abstraction generally groups states that behave similarly and still allows learning a near-optimal policy. However, constructing such an abstraction requires knowledge of the dynamics of the problem. Since RL problems typically involve environments that are (largely) unknown, this can be a challenge. This leads to an important question: *how could we create a good abstraction before we start learning?*

Finally, we have not addressed yet that in many problems the state is not fully *observable*. This is similar to the example in Figure 1.5, where the robot does not observe the full state. However, the problem was fully observable, and abstraction made it partially observable in that example. In other problems, the state is inherently partially observable, even without abstraction. This complicates learning and generally leads to a much larger state space, as the history of observations and actions becomes important in finding a good policy.

1.5. CONTRIBUTIONS OF THIS THESIS

This thesis investigates the intersection of MBRL and abstraction, called MBRLAO. We provide an overview of work and open questions in this area and address, theoretically and empirically, some of the identified open questions. Theoretically, we study and prove how to provide efficient learning guarantees in MBRLAO. Empirically, we investigate using the structure of environments to create good abstractions and leveraging these abstractions in online planning and learning in partially observable environments.

CHAPTER 2: MODEL-BASED RL WITH STATE ABSTRACTION: A SURVEY [43]

In Chapter 2, we focus on previous work in the MBRLAO setting. MBRLAO is of interest for improving the sample efficiency of learning methods that aim to solve sequential decision-making problems. Even though MBRL and abstraction have received considerable attention, e.g., [15, 41, 44, 45], there has been no comprehensive overview of the intersection between these fields.

Chapter 2 addresses this gap by reviewing existing MBRLAO approaches and introducing a novel framework for interpreting these methods. The framework categorizes the work into two groups: those that utilize one or more predefined abstraction functions and those that simultaneously learn both the environment and an abstraction function. The lens provided by this framework helps to establish a clear overview of various facets of integrating abstraction and MBRL, as well as to identify problems and opportunities for further research.

An important insight from our analysis is that MBRLAO can be viewed as changing the problem from fully observable to partially observable. This transformation from an MDP to a POMDP means that guarantees for MBRL methods, which assume an MDP, may no longer hold in MBRLAO. This presents an important open question: can results for MBRL in MDPs be directly applied to MBRLAO?

We also highlight a promising direction for online planning using abstractions and MBRL. Research has demonstrated that, with limited planning time, planning with a compact learned model outperformed planning with the actual model of the environment [46]. Employing abstraction to create a smaller model could enhance performance through quicker planning and accelerated learning. However, there may be a trade-off in learning with abstraction, as a coarser model could yield better results with shorter planning times but worse outcomes with longer ones. An interesting open question is: under what conditions could an abstract model

facilitate improved performance by balancing computational efficiency, learning speed, and model accuracy?

CHAPTER 3: AN ANALYSIS OF MODEL-BASED RL FROM ABSTRACTED OBSERVATIONS [47]

Chapter 3 presents a theoretical perspective on MBRLAO and focuses on approximate state abstractions, particularly the approximate model-similarity abstraction. Approximate abstractions are interesting because they allow for a larger reduction in the problem size compared to exact abstractions. However, using abstractions can lead to a loss of information. The complications associated with this have largely been avoided by work in MBRLAO [37, 48–52], leaving the open question posed before: whether results for MBRL without abstraction can be directly transferred to MBRLAO.

As a first contribution, we investigate the challenges of combining MBRL and abstraction. Using the insight from Chapter 2 that MBRLAO makes the problem partially observable, we analyze how the characteristics of the observed examples are affected by an abstraction function. We construct a counterexample to demonstrate that samples obtained in MBRLAO are no longer Markovian: they cannot be guaranteed to be independent nor identically distributed. This is important since it means guarantees of MBRL methods in MDPs do not directly transfer to MBRLAO since the Markov assumption is key in establishing performance guarantees of MBRL methods.

We then focus on a way around this negative result, to still provide guarantees in MBRLAO. Key in establishing performance guarantees of MBRL is showing that an accurate model can be learned, which normally relies on the Markov assumption. While we cannot rely on the Markov assumption, we show that samples in MBRLAO are only weakly dependent and that the learning process constitutes a martingale difference sequence [53, 54]. We use this insight and the properties of martingale processes to establish a theoretical result that proves we can accurately learn a model in MBRLAO.

Proving that we can still learn an accurate model in MBRLAO is a significant result, as this establishes, for the first time, that it is possible to transfer the guarantees of MBRL to MBRLAO. Finally, we demonstrate this explicitly through transferring the results of the R-max algorithm [13] to MBRLAO using an approximate model-similarity abstraction. These results exemplify that MBRL and abstraction can be combined in a way that leads to efficient learning.

CHAPTER 4: ABSTRACTION FOR BAYESIAN RL IN FACTORED POMDPS [55]

Chapter 4 presents a more practical perspective, focusing on creating abstractions and empirically utilizing them in partially observable environments. From this perspective, it is important to focus on sample complexity, especially in real-world applications where data collection is expensive, difficult, or dangerous. Many real-world applications offer prior knowledge, and incorporating this knowledge into the learning process is crucial for efficient learning [56–58]. Model-based Bayesian RL (BRL) [21, 23, 59] offers a way to incorporate prior knowledge via Bayesian priors.

We build on the Factored Bayes-Adaptive POMDP (FBA-POMDP) framework [60, 61], a model-based BRL approach that combines partial observability and structured factored models. It uses factorized representations of the dynamics of the environment, allowing agents to exploit problem structure for improved scalability. While this framework is promising as factorization enables better generalization, irrelevant state factors in the model can lead to unnecessarily large model spaces, making planning and learning more difficult. Abstraction can play an important role by removing less important factors, which can simultaneously improve both planning and learning. We incorporate abstraction into the FBA-POMDP framework to enhance scalability and learning efficiency. In doing so, Chapter 4 addresses how to create effective abstractions before learning begins and how to leverage them in domains that require both learning and online planning.

As a first contribution, we introduce a method to create effective abstractions in RL. This is challenging due to the need to group states with similar characteristics, requiring knowledge of the problem domain, which is often limited. The factored Bayesian model enables us to incorporate prior knowledge and leverage the structural characteristics of the problem to inform the abstraction process. The structural characteristics help identify which factors can be considered less relevant and safely removed, minimizing the loss of essential details. Our method creates abstractions automatically based on the problem's structure, enabling agents to plan and learn more effectively. Further, by maintaining a belief in the structure and dynamics of the problem, we can automatically adapt the abstraction based on observations. This represents a novel step toward combining abstraction with BRL in Factored POMDPs (F-POMDPs).

Empirically, we used several domains to evaluate our approach and investigate the benefits of abstraction for planning and learning, yielding several important insights. We demonstrate that abstract models can improve performance through faster online planning due to the reduction in the model size. Interestingly, this also occurred when the abstract model did not accurately represent the original problem, where information crucial for optimal performance was removed, and the increased simulation speed provided by the abstraction compensated for the loss of model accuracy.

Furthermore, additional experiments under a fixed number of simulations show that abstractions can improve performance due to their greater statistical strength. Abstract models can use experience more efficiently because they aggregate multiple samples that the original model treats independently. As we gather more data, the full model may eventually surpass the performance of the abstract model. Nevertheless, this can require a lot of observations, especially when a reduction of the model size due to abstraction results in substantial planning speed improvements. Before the full model catches up, the performance of the abstract models can be significantly better, highlighting their importance in learning.

1.6. OTHER RESEARCH CONTRIBUTIONS

Outside the scope of this thesis, I have contributed to research on influence-based abstraction and have supervised several bachelor's and master's students on projects and theses. One of these bachelor projects focused on exploration approaches in deep RL, which resulted in a publication.

INFLUENCE-BASED ABSTRACTION

I collaborated on a project to improve efficiency in deep RL, led by Miguel Suau [33, 62]. My involvement was in discussions during the project's ideation and designing initial experiments. The work was inspired by the influence-based abstraction framework [32]. This type of abstraction captures a smaller part of the environment, the local problem. To accurately capture the dynamics of the local part, a predictor that quantifies the influence of variables outside the local problem on the local problem is used. Our work uses the influence-based abstraction idea in the form of an influence-aware memory, a novel neural network architecture. It improves learning in partially observable environments by filtering out observation variables that do not influence and are not influenced by hidden states. This approach makes learning easier for the network and improves training speed and policy performance compared to standard methods.

EXPLORATION IN DEEP RL

Yaniv Oren, a former bachelor's student, investigated the importance of exploration in traffic light control tasks by comparing several deep exploration methods [63]. Reducing traffic congestion could lead to a large saving in costs as it is estimated to be 1% of the GDP in the EU [64]. RL could be a promising method for reducing traffic congestion by providing better traffic light control, and research applying deep RL has been carried out in this area [65]. We explored the importance of efficient exploration in the traffic control setting to improve traffic light control. Specifically, we used a standard approach of ϵ-greedy exploration in deep Q-networks [8]. We compared this to two state-of-the-art deep exploration methods, bootstrapped deep Q-networks [66] and randomized prior functions [67], and their combination. We used three different traffic scenarios of varying complexity to investigate these methods. The results suggest that the gain of efficient exploration becomes more important the more complex the scenario and the larger the observation space of the agent.

2

Model-Based RL with State Abstraction: A Survey

Model-based reinforcement learning methods are promising since they can increase sample efficiency while simultaneously improving generalizability. Learning can also be made more efficient through state abstraction, which delivers more compact models. Model-based reinforcement learning methods have been combined with learning abstract models to profit from both effects. We consider a wide range of state abstractions that have been covered in the literature, from straightforward state aggregation to deep learned representations, and sketch challenges that arise when combining model-based reinforcement learning with abstraction. We further show how various methods deal with these challenges and point to open questions and opportunities for further research.

Parts of this chapter have been published in Artificial Intelligence and Machine Learning, 34th Joint Benelux Conference, BNAIC/Benelearn 2022, Mechelen, Belgium, November 7–9, 2022, Revised Selected Papers (2023) [43].

2.1. INTRODUCTION

With roots in sequential analysis [68], Reinforcement Learning (RL) is a general framework for learning how to act near-optimally in sequential decision-making problems. A key challenge for RL is sample efficiency. Sample efficiency is important because, in many problems, it can be expensive, in time or monetary costs, to collect samples. The combination of Model-based Reinforcement Learning (MBRL) and abstraction is of interest for improving the sample efficiency of learning methods that aim to find solutions for sequential decision-making problems. We define MBRL as an RL method that explicitly learns a model of the environment. MBRL provides a way to find solutions to complex problems efficiently [14] and allows for transfer in shifting or related tasks [41, 69]. The state representation, the input to RL methods, plays an essential role in the learning process. A state representation will often contain irrelevant details, e.g., when the input is an image, a large amount can consist of a background that has no direct relevance to the task. Abstracting the state representation to remove irrelevant parts for optimal decision-making allows RL methods to learn much faster. Learning to decide which parts of the state representation are relevant is a key aspect of abstraction learning.

State abstraction can be carried out in various ways, ranging from state aggregation [15, 16] to deep learned representations [45, 46]. We provide a high-level view of the promising research in the field, covering a wide range of different types of state abstractions known from the literature.

Recently MBRL, abstraction learning, and related topics have received much attention. There are surveys of decision-making under uncertainty [70], MBRL in general [69], deep MBRL [71], and representation learning in both robotics [72] and MBRL [73]. Our work takes a broad view of abstraction and focuses on the additional challenges that arise when combining MBRL and abstraction. For these additional challenges, see also [34, 38, 41] and Chapter 3. The contributions of this work are the following: We detail challenges that arise from the combination of MBRL with abstraction using the view of abstraction plus RL as a Partially Observable Markov Decision Process (POMDP). We show how different approaches for MBRL with state abstraction deal with these challenges, providing a unified view of a wide range of approaches in the process. We identify open questions and opportunities for further research.

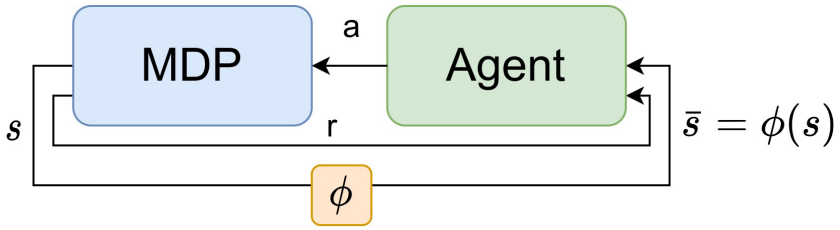


Figure 2.1: RL with abstraction, the agent observes $\bar{s} = \phi(s)$ instead of s. Image based on Abel *et al.* [41].

2.2. AN OVERVIEW OF STATE ABSTRACTION FOR RL

We consider RL in sequential decision-making problems, which can be defined as a Markov decision process (MDP) [74]: $\langle S, A, T, R, \gamma \rangle$, where S is a set of states $s \in S$, A a set of actions $a \in A$, T a transition function $T(s'|s,a) = \Pr(s'|s,a)$, R a reward function $R(s,a)$ which gives the reward received when the agent executes action a in state s, and γ the discount factor ($0 \leq \gamma < 1$). For realistic problems, the state space of the MDP representation is often too large to tackle directly. One way to reduce the size is to use compact representations such as state abstractions. Section 2.2.1 characterizes different state abstractions methods and briefly describes some of their properties. Section 2.2.2 describes how abstraction in an MDP can be viewed as a POMDP and the resulting challenge.

2.2.1. CHARACTERIZATION OF ABSTRACTIONS

State abstraction can be used to reduce the problem size by clustering states into abstract states. This clustering can be defined by using an abstraction function ϕ, which maps (or aggregates) ground states s to abstract states $\bar{s}$, where the bar notation denotes objects in the abstract space. Here we consider a discrete state space and write this mapping as $\phi(s) = \bar{s}$, such that the abstract state space can be written as $\bar{S} = \{\phi(s) \mid s \in S\}$. The agent then uses the abstract states $\bar{s}$ and the rewards for learning transitions and rewards over the abstract state space. State abstraction can result in an abstract state space that is much smaller than the original state space, $|\bar{S}| \ll |S|$, which can make learning easier.

In the planning setting, where we have access to the model of a problem, many different abstraction functions have been considered [15, 16]. Abstractions group states based on specific criteria of the state or state-action pairs. An example is the (stochastic) bisimulation [75], also known as model-irrelevance abstraction [15]. In this abstraction, states are only grouped if their reward and transition functions in the abstract space are the same, i.e., $\phi(s_1) = \phi(s_2)$ iff

$$\forall_{a \in A} \; R(s_1, a) = R(s_2, a), \tag{2.1}$$

$$\text{and } \forall_{\bar{s}' \in \bar{S}} \; T(\bar{s}'|s_1, a) = T(\bar{s}'|s_2, a). \tag{2.2}$$

Here $T(\bar{s}'|s,a)$ is the transition to an abstract state $\bar{s}'$ which is defined as

$$T(\bar{s}'|s,a) := \sum_{s' \in \bar{s}'} T(s'|s,a). \tag{2.3}$$

If we have access to the MDP, we can compute a more compact abstract MDP [76] and find a solution for this smaller problem. An important aspect of these abstractions is whether or not (near) optimal policies for the original policy can be obtained when the policy is learned from the abstract problem. Several results showing that this is possible have been obtained for multiple forms of abstraction [15, 16]. These results make abstractions interesting for RL as they show that it is possible to significantly reduce the problem size while still being able to obtain (near) optimal policies for the original problem.

To allow for further reduction in the problem size, approximate versions of abstractions, such as the ϵ-bisimulation, have been considered [15, 16]. In the approximate versions, the grouping criteria are relaxed. E.g., in the ϵ-bisimulation, the transition and reward functions for grouped states will be close but not necessarily the same, i.e., $\phi(s_1) = \phi(s_2)$ iff

$$\forall_{a \in A} \ |R(s_1, a) - R(s_2, a)| \leq \epsilon, \tag{2.4}$$

$$\text{and } \forall_{\bar{s}' \in \bar{S}} \ |T(\bar{s}'|s_1, a) - T(\bar{s}'|s_2, a)| \leq \epsilon, \tag{2.5}$$

where $T(\bar{s}'|s, a)$ is defined as in (2.3). Several other examples of exact and approximate state abstraction functions can be found in the literature [15, 16]. For a given MDP, it is possible to build an abstract MDP using ϵ-bisimulation criteria [77]. Recent work has introduced transitive state abstractions, which can be computed efficiently [41]. If we have a compact model, the goal is to find a good policy. A potential issue is that if a learned model only approximates the true model, minor errors can compound when planning for long horizons [78, 79]. Results for planning have shown that for particular approximate state representations, such as ϵ-bisimulation, the learned policy can still be approximately optimal [16]. There is a similar result for using RL in an abstract MDP [80]. However, these results assume that we have access to the MDP or an abstract MDP, which requires the problem to be known, and this is typically not the case in RL.

2.2.2. ABSTRACTION IN AN MDP AS A POMDP

In the general case of MBRL in an unknown MDP with an abstraction ϕ, the situation will be as depicted in Figure 2.1. Without abstraction, the agent receives a state s as an observation. With abstraction, the agent instead observes an abstract state $\bar{s} = \phi(s)$ through the abstraction function ϕ. In this case, the agent will no longer know precisely which state it is in, making the environment (a special case of) a POMDP, see [30, 34, 35, 73, 81] and Chapter 3. Abstraction can be seen as a special case of POMDPs because the observation results from *perceptual aliasing*, i.e., multiple states are perceived as the same. Perceptual aliasing may not be a problem when the resulting problem behaves as an MDP, as for a bisimulation [75, 82], but this is often not the case as shown in [38, 41] and Chapter 3.

To formalize the combination of abstraction and RL in an MDP as a special case of a POMDP, we first give the general definition of an infinite horizon POMDP [19], which can be described by the tuple $\langle S, A, T, R, \Omega, O, \gamma \rangle$, where S, A, T, R, and γ are the same as in the MDP. The Ω is a finite set of observations $o \in \Omega$ that an agent can receive, and O is an observation function $O(o|a, s') = \Pr(o|a, s')$ that gives the probability of receiving an observation o after taking an action a and ending in state s'. Now, when an RL agent acts in an MDP but receives observations through ϕ, the uncertainty is only due to perceptual aliasing, which means that the observation is a deterministic function of the state: $O(o|a, s') = \Pr(o = \phi(s')|a, s') = \Pr(o = \phi(s')|s')$. For deterministic functions ϕ, this is 1 iff $o = \phi(s')$. The abstraction function ϕ has taken the role of the observation function O, with the observation space being $\bar{S}$.

Since we can view the combination of abstraction and RL in an MDP as a special case of a POMDP, RL methods for POMDPs could be used to find a solution. A

common approach to finding solutions in POMDPs is through Bayesian RL, for which the Bayes-Adaptive POMDP (BA-POMDP) provides a framework [59]. Extensions of Bayesian RL for POMDPs are covered in the survey of Ghavamzadeh *et al.* [23]. In Deep RL, using recurrent neural networks is one way in which partial observability has been addressed [83, 84]. Specific focus has been on using variational inference methods [85, 86] and belief tracking [84, 87, 88]. However, these POMDP approaches are often general solutions for any POMDP, and they are not necessarily optimal for the special case of the POMDP induced by abstraction.

Instead of applying POMDP solution methods, it can be tempting to treat the resulting problem as a Markov problem and try to find a solution in this way. For instance, this could be tempting when the abstraction clusters together states with similar transition and reward functions in the abstract space, such as with a ϵ-bisimulation abstraction. However, as noted by [38, 41] and discussed in Chapter 3, treating this problem as a Markov process can lead to policies that are far from optimal, and there could be no guarantee of finding an optimal solution. In general, non-stationarity of the collected data, due to changing behavior of the policy, has been shown to lead to worse performance in Deep RL [89], and non-stationarity due to perceptual aliasing can lead to similar problems when not addressed. Therefore, to find good solutions, methods that combine RL and abstraction should take into account perceptual aliasing.

2.3. UTILIZING GIVEN ABSTRACTION FUNCTIONS

This section presents an overview of the literature that utilizes an abstraction function for MBRL. First, Section 2.3.1 discusses the relation between abstract MDPs and Robust MDPs (RMDPs) and how solution methods for RMDP can allow for obtaining better policies when using an abstract learned model. Section 2.3.2 considers the RL setting where we do not have such a model, but we are given some abstraction function ϕ and see how abstraction can be leveraged to improve performance. Section 2.3.3 deals with the setting where we are given a set of abstractions and have to learn which one leads to optimal performance. Afterward, Section 2.4 deals with the setting where we do not have an abstraction function ϕ and have to learn one online.

2.3.1. ROBUST OPTIMIZATION

The RMDP [90] and the related Bounded Parameter MDP (BPMDP) [91] extend the MDP definition by allowing for uncertainty in the transition and reward functions, as quantified by intervals. This uncertainty is generally motivated by not having enough data to be sure about the transition functions but still being able to give some confidence intervals. Another motivation is inherent uncertainty, for instance caused by having a ϵ-bisimulation, where the uncertainty intervals are ϵ wide. If we learn an ϵ-bisimulation model and can estimate ϵ, we could apply solution methods for RMDPs, this makes solution methods for RMDPs interesting for RL with abstraction. For a comprehensive introduction and overview of RMDPs, see the survey by Suilen *et al.* [92].

To solve problems with inherent uncertainty, the RMDP extends the MDP definition by including an extra set of outcomes B. The transition probabilities and reward function are then modeled as functions of both an action $a \in A$ and an outcome $b \in B$. From a game-theoretic perspective, B can be interpreted as the actions of an adversary [93]. A solution to an RMDP therefore also includes the policy of the adversary.

In the RMDP setting, it has been shown that the problem of finding an optimal robust policy is computationally intractable. In particular, early work established that the problem is strongly NP-hard [90], and later research strengthened this result by proving that analyzing non-rectangular RMDPs is ETR-complete [94]. To make the problem tractable, different assumptions about the policy of the adversary have been considered [90, 93]. In one case, the adversary can independently choose an outcome for each state s. In another case, outcomes are chosen independently for each state s and action a pair. More recent work has studied less restrictive adversarial policies [95, 96]. Overall, accounting for uncertainty with robust optimization can lead to policies that perform better in real environments [90, 97].

There has also been work that combines abstraction with RMDP [93, 97]. The RAAM algorithm [93] receives an abstraction function and an MDP as input. It first constructs an RMDP and uses this to compute an approximately optimal policy for the original MDP. It is shown that this can be beneficial in the limit; bounds on the performance are given that are similar to the bounds for ϵ-bisimulation abstractions in planning [16]. The RAAM approach was later extended by Lim and Autef [97], who use a kernel-based approach, of which state abstraction can be seen as a special case.

The work in this section shows that uncertainty about the transition and reward functions can be dealt with in a principled way, given some uncertainty intervals. While some work connects this work to abstraction, it only focuses on results in the limit.

2.3.2. LEVERAGING AN ABSTRACTION FUNCTION

In this setting, we make use of a given abstraction function ϕ, which maps the original state space into a smaller abstract space. This ϕ could, for instance, come from a domain expert or result from the discretization of a continuous problem. Rather than learning a model of the full environment, the idea is to learn a model directly in the abstract state space created by the mapping of ϕ. This is typically done by collecting data and then constructing a maximum-likelihood estimate of the transition and reward functions over the abstract states.

If we learn a correct abstract model and find the optimal policy for this abstract model, this policy can be near-optimal in the true MDP, depending on the abstraction used [16]. Learning in this way could be more sample-efficient than learning a model of the full MDP because the abstract space is smaller than the original state space.

One difficulty in this setting is learning a correct abstract model in the first place. In RL, samples can usually be considered independent, and this is used to show that an accurate model can be learned. In the combination of RL and abstraction,

samples can no longer be considered independent due to perceptual aliasing as observed by [34, 98] and detailed in Chapter 3. In order to give sample efficiency results for RL plus abstraction, some work assumes that the collected samples are independent [37, 99]. The work by Paduraru *et al.* [37] assumes that they receive a data set with independent and identically distributed (i.i.d.) samples and show a trade-off between the quality of the abstraction and the quality of the transition model. The quality of the abstraction is measured in terms of the ϵ of ϵ-bisimulation. A larger ϵ means a coarser abstraction and a larger error. The second error relates to the number of samples we can get for a state-action pair, where a coarser abstraction gives more samples per state-action pair and a lower error. Like the work by Paduraru *et al.* [37], other work has also shown that the error of the agent can be decomposed into multiple components, which are based on the asymptotic bias of the representation and overfitting due to limited data (variance) [99, 100]. This bias-variance trade-off indicates that using abstractions can be especially beneficial when the available data is limited while being less beneficial when much data is available, which has been illustrated in experiments [99].

The assumption that the generated data consists of independent samples does not hold in general. Another way to show that we can learn an accurate abstract model is by looking at convergence in the limit. The convergence to an accurate estimation of the abstract model is possible under several conditions, e.g., when the policy is fixed or when the abstraction is a bisimulation [34, 98]. Having to use a fixed policy can be seen as a downside because a changing policy that explores helps to learn efficiently [14]. Another downside is that, in the limit, using the full model will be better than using an abstract model since only the error introduced by the bias remains, which is zero for the full model.

In Chapter 2, we show that an accurate abstract model can still be learned by applying martingale theory [101]. We give the first finite-sample performance analysis for model-based RL plus abstraction by extending the results of an existing algorithm (R-MAX [13]) with the use of an ϵ-bisimulation abstraction.

This section shows that abstractions can lead to better performance with fewer data, trading it off with less accuracy when much data is available. For these methods to work, it is required to already have a good abstraction function, which can be challenging.

2.3.3. ABSTRACTION SELECTION

While the work in the previous section mainly focused on the case where we have one particular abstraction function, there is also a considerable amount that has focused on state representation selection, where the agent is provided with a set of state representations (or abstraction functions). It is usually assumed that a domain expert provides these representations, and the goal is to select the best representation, often in terms of regret.

Most of this work focuses on finding representations that make the problem Markov instead of focusing on finding good approximate abstractions. In order to deal with perceptual aliasing, most work assumes that the provided set contains a Markov model of the environment [48, 49, 51, 102, 103]. In order to find a

correct representation in the online setting, these algorithms eliminate non-Markov models by comparing the obtained rewards during execution with a threshold based on a Markov model. The work by Lattimore, Hutter, and Sunehag [104] considers a similar setting where the dynamics of the true environment depend arbitrarily on the history of actions, rewards, and observations. Instead of getting a set of representation functions, they assume access to a given set of models, one of which is a correct model of the true environment. In this way, they can compare the calculated expected reward for the given model with the rewards obtained during the process and eliminate the unlikely models.

Other work does not assume that a Markov representation is available [39, 40], these both use an ϵ-bisimulation type abstraction. The work of Ortner, Maillard, and Ryabko [39] builds on the work of Maillard *et al.* [49] by removing the necessity of having a Markov representation in the set of available representations. However, it has been shown that the regret proof of Maillard *et al.* [49] assumes a certain difference is always positive, which is not guaranteed Fruit, Pirotta, and Lazaric [105]. This makes the bound used in the analysis incorrect, and the claimed guarantees no longer hold. They also do not take into perceptual aliasing since they use a concentration inequality that requires i.i.d. samples. The work by Jiang, Kulesza, and Singh [40] deals with perceptual aliasing by explicitly assuming in their analysis that a data set consisting of samples that are i.i.d. is available. They give a performance bound for policies based on a learned abstract model and split the error into two components, similar to some of the work mentioned in Section 2.3.2 [37, 99]. These two components are used to create an algorithm that decides which representation should be used based on the available data.

The methods in this section show that we can learn to select a correct (Markov) representation, given an initial set of representations. Most of these methods are not very scalable, as they are tabular, and finding a good (Markov) representation/abstraction in larger problems can be challenging.

2.4. ONLINE ABSTRACTION LEARNING

The previously discussed works have mostly assumed that an abstract representation (or a set thereof) is readily available. However, this is not always possible. In this section, we consider the situation where such an abstraction is unavailable and has to be learned first while simultaneously learning about the environment. Two early studies on this topic provided promising experimental results [81, 106]. Section 2.4.1 covers tabular approaches, which have mostly been more theoretical, and Section 2.4.2 covers deep learned representations focused on scaling up.

2.4.1. TABULAR APPROACHES

The combination of MBRL and abstraction has also been approached theoretically. The work by Bernstein and Shimkin [107] gives results for online abstraction when the transition functions are deterministic. The work by Ortner Ortner [38] explores the more general case of stochastic transition functions when trying to learn a ϵ-bisimulation. To learn a ϵ-bisimulation they maintain an interval on the estimation

of the transition and reward functions for each state-action pair, which is used to create a BPMDP [91]. Subsequently, the BPMDP is abstracted by clustering the states that overlap in the transition and reward function for all actions, but only if they have a similar amount of samples. They give an example to show that clustered states must have a similar amount of samples for all the actions to obtain good performance. This is an interesting observation since it points out a problem that should be taken into account when learning an abstraction in combination with MBRL. A downside of the method is that it focuses on the computational benefit abstraction can bring; from the perspective of sample efficiency, a method that utilizes abstraction to learn more efficiently is desirable.

In the Bayesian RL setting, the work by Mandel *et al.* [108] proposes an algorithm that does online clustering and exploration. The clustering is done over state-action pairs rather than only over states. State-action abstractions allow for a broader class of abstractions since state abstractions can be considered a subset of state-action abstractions while potentially still being optimality preserving. This gives additional power in doing the abstraction since, in some domains, there could be no similar states while similar state-action pairs exist. State-action pairs are grouped when the relative outcomes are likely to be the same. Relative outcomes are similar to observations. Given a relative outcome, the agent knows both the transition and reward. However, it needs to learn the distribution over relative outcomes for each state.

Work in block MDPs, or MDPs with rich observations, is a related approach where the assumption is that each state can generate multiple different observations [52, 109–112]. Instead of having multiple states that generate the same observation (due to the abstraction function), each type of observation is only generated by one state, but each state can generate multiple observations. This is similar to representation learning, specifically to learning a bisimulation [52, 109, 112]. A common approach in this setting is to use spectral methods [109–111]. For these to work, it is necessary to be able to uniquely identify states from the observation function. While this is possible for model-irrelevance abstractions, this is generally not possible in the abstraction setting.

The focus of tabular approaches has been on block MDPs, which can lead to a considerable reduction in the state space in suitable problems. However, this does require the problem to have many states with the same behavior in an abstract space, i.e., there needs to be a bisimulation abstraction. This restricts the number of problems to which these methods can be applied.

2.4.2. DEEP LEARNED REPRESENTATIONS

There have also been several Deep RL approaches that focus on learning compact state representations, which can be viewed as an instance of state abstraction. For instance, see the approaches by [35, 45, 46, 79, 113–116]. One crucial notion for abstraction in deep RL is a collapse of the latent representation [35, 45, 57, 116]. When considering only the transition function, it would be optimal to cluster all states into exactly one abstract state. It has been shown that losses that require both the transition and reward function of grouped states to be the same can avoid

this collapse [117], making it essential to group states based on both transitions and rewards.

Recently, multiple contrastive methods have been used to learn compact representations for predicting the next state [118–120]. Their representation learning tries to maximize the mutual information between the present and future samples. To train the network, they use positive and negative next-state samples, where the positive samples are transitions that occurred, while the negative samples are transitions that did not occur. These negative samples should help prevent the potential collapse of the state representation. Their methods do not use the model to plan the policy but instead use actor-critic and policy optimization methods on top of the representation. The proposed representation learning method was able to help improve the performance of these methods.

Other work has focused on learning deep representations for robotics [56, 57, 121]. This has investigated adding several types of robotic priors to bias the representation learning, which are added to the network as an auxiliary loss [83]. These priors encode knowledge about physics, e.g., that changes in the state are often gradual rather than abrupt. The state-representation objectives were instrumental in generalizing, as they significantly improved the results in the test domain. This shows that learning a compact model of the environment can be beneficial even if the model itself is not directly used for planning. Other methods for robotics focus on finding compact linear representations of a problem and finding a policy for this smaller model [122–124]. This has shown promising results for robotics, where many of the essential state features could be approximately linear.

Most of the work in this section focused on learning exact abstractions. They try to reduce the problem so that the resulting latent representation still makes the problem an MDP. This can be difficult to ensure, especially in Deep RL, so it is likely that the resulting representation is an approximate abstraction. Since most work does not acknowledge this, they do not consider the resulting perceptual aliasing, and algorithms can experience the problem illustrated by [38]: when states with a different number of visitations are grouped, this can lead to suboptimal policies. When this is not taken into account, this can lead an agent to be stuck in a suboptimal loop.

2.5. DISCUSSION AND CONCLUSION

We summarize our overview in Table 2.1, which compares the approaches on the type of environment, whether or not a model is given, how an abstraction ϕ is obtained, what kind of abstraction is used, available theoretical support, scalability, and how they deal with perceptual aliasing.

The methods in Sections 2.2 and 2.3 generally have strong theoretical support (V) in the form of bounded loss (e.g., [16, 90]), finite-sample guarantees (e.g., Chapter 3, [37]), or regret bounds (e.g., [102]). Most of these methods are not (X) scalable due to being tabular or only somewhat scalable (~) due to needing to be given a model, which in many cases is not possible. In most of these works, the problem of perceptual aliasing does not arise, either because of assumptions on

Section	Method	Environment	Model	Abstraction ϕ	Abstraction Type	Theory	Scalability	Perceptual Aliasing
2.2.1	Planning	MDP	Given	Constructed	Many	V	~	Not an issue
	Tabular RL	Abstract MDP	Given	Build-in	Bisimulation related	V	X	Not an issue
2.3.1	Robust Optimization	MDP	Given (interval)	Build-in	Bisimulation	V	~	Assumption on uncertainty
	Robust Optimization	MDP + ϕ	Given	Given	Bisimulation related	~	X	Not an issue
2.3.2	Tabular RL	MDP + ϕ	Unknown	Given	Bisimulation related	V	X	Assumptions on data gathering
2.3.3	Abstraction Selection	MDP + $[\phi_1, \cdots, \phi_n]$	Unknown	Given	Several	V	X	Markov representation, assumption on data gathering
2.4.1	Tabular RL	MDP + ϕ	Unknown	Learned	Bisimulation related	V	X	Markov representation, specific check
	Bayesian RL	MDP + ϕ	Unknown	Learned	(s,a)-abstraction	V	X	Markov representation
	Spectral Methods	Block-MDP	Unknown	Learned	Bisimulation related	V	~	Markov representation
2.4.2	Deep RL	MDP + ϕ	Unknown	Learned	Bisimulation related	~	V	Markov representation, potential problem
	Contrastive Loss	MDP + ϕ	Unknown	Learned	Bisimulation related	X	V	Markov representation, potential problem
	Linear Latent Representations	MDP + ϕ	Unknown	Learned	Linear Function	X	V	Markov representation, potential problem

Table 2.1: Characterization of MBRL methods in combination with a type of state abstraction.

data gathering or because an MDP, or MDP representation, is provided. We show in Chapter 2 that finite-sample bounds for MBRL in an MDP with an ϵ-bisimulation can be obtained, without assuming that samples are independent. *Extending these results to other types of abstractions is still an open question.*

In Section 2.3.2, we saw a bias-variance trade-off with abstractions [37, 40, 99, 100]. Because of this trade-off, *an interesting direction would be to combine learning multiple representations with abstraction selection to decide which representation to use at which time.*

As discussed in Section 2.3.1, results for optimization under uncertainty could make it interesting to maintain confidence intervals for the learned models and use robust optimization to find policies. Since the model will generally not be completely accurate during learning, robust optimization could improve performance [97]. Tabular work discussed in Sections 2.3.1 and 2.4.1 investigated this idea [38, 93], *scaling such approaches to larger problems is an interesting future direction.*

Most of the focus has been on abstractions related to bisimulation. As touched upon in Section 2.4.1, abstractions that aggregate state-action pairs can be more potent than state abstractions [108]. An open question is *what are the best types of abstraction to use?* Non-deterministic abstraction [34], temporal abstraction, or combinations of abstractions could be powerful but have not been as well studied [125].

In [46], there is some indication that, in online planning, using a coarser learned model rather than the true model can be beneficial. With limited planning time, planning with a compact learned model outperformed planning with the true model of the environment. *There could be a trade-off for learning between the coarseness of the model and the allotted planning time; a coarser model could perform better with a shorter planning time but worse with a longer planning time.*

The methods in Section 2.4.2 focus on learning abstractions that result in a Markov representation, e.g., bisimulation abstractions. However, *during learning, when the abstraction is likely not a Markov representation, perceptual aliasing occurs. How can the resulting non-stationarity be addressed?* In Section 2.4.1, we saw that the tabular work by Ortner [38] deals with perceptual aliasing, but to do so, it maintains visitation counts for all state-action pairs. Methods that can maintain counts in an approximate way, such as pseudo-counts [80], could enable a scalable version of the approach by Ortner [38]. Another approach to deal with perceptual aliasing in a more sample-efficient way could be using an algorithm such as ITER [89], which tackles the general non-stationarity of the data distribution caused by the RL algorithm. The idea of the algorithm is to frequently transfer the knowledge of the trained network to a new network and then use the new network for training. The knowledge is transferred through samples that are obtained from the collected data set as if they had been generated with the final policy of the trained network.

In multi-agent RL, the challenge is to behave optimally in the presence of other agents whose behavior may be non-stationary [126]. *Approaches for the multi-agent RL problem that address non-stationarity could be insightful for the combination of RL and abstraction.* One approach that could be relevant is trying to capture the non-stationarity that is the result of perceptual aliasing, which could, for instance,

be done by using influence-based abstraction [32]. Influence-based abstraction aims to abstract a problem into a smaller local problem with a predictor that quantifies the influence of variables outside the local problem on the local problem. Given an accurate predictor, this results in a Markov problem. Such a predictor could capture the non-stationarity due to perceptual aliasing and improve performance. Influence-based abstraction has been applied together with Deep model-free RL, using a recurrent neural network to capture the influence, which has shown promising results [62].

Other approaches in multi-agent RL do not deal with the non-stationarity but simply ignore it by abstracting away the internal states of the other agents. Since this can be seen as a special case of the non-stationarity in the combination of RL and abstraction, insights from this combination on how to deal with non-stationarity as a result of perceptual aliasing could provide interesting directions for these multi-agent RL approaches.

3

AN ANALYSIS OF MODEL-BASED RL FROM ABSTRACTED OBSERVATIONS

Many methods for Model-based Reinforcement learning (MBRL) in Markov decision processes (MDPs) provide guarantees for both the accuracy of the model they can deliver and the learning efficiency. At the same time, state abstraction techniques allow for a reduction of the size of an MDP while maintaining a bounded loss with respect to the original problem. Therefore, it may come as a surprise that no such guarantees are available when combining both techniques, i.e., where MBRL merely observes abstract states. Our theoretical analysis shows that abstraction can introduce a dependence between samples collected online (e.g., in the real world). That means that, without taking this dependence into account, results for MBRL do not directly extend to this setting. Our result shows that we can use concentration inequalities for martingales to overcome this problem. This result makes it possible to extend the guarantees of existing MBRL algorithms to the setting with abstraction. We illustrate this by combining R-MAX, a prototypical MBRL algorithm, with abstraction, thus producing the first performance guarantees for model-based 'RL from Abstracted Observations': model-based reinforcement learning with an abstract model.

Parts of this chapter have been published in Transactions on Machine Learning Research [47].

3.1. INTRODUCTION

Tabular Model-based Reinforcement Learning (MBRL) methods provide guarantees that show they can learn efficiently in Markov decision processs (MDPs) [13, 14, 20, 127–130]. They do this by finding solutions to a fundamental problem for Reinforcement Learning (RL), the exploration-exploitation dilemma: when to take actions to obtain more information (explore) and when to take actions that maximize reward based on the current knowledge (exploit). However, MDPs can be huge, which can be problematic for tabular methods. One way to deal with large problems is by using abstractions, such as the mainstream state abstractions [16, 82]. State abstractions reduce the size of the problem by aggregating together states according to different criteria, depending on the specific type of abstraction. We can view state abstraction as a special case of function approximation, where every state maps to its abstract state [131], and we can roughly divide them into *exact* and *approximate* abstractions [16, 82].

Approximate abstractions relax the criteria of exact abstractions, and therefore allow for a larger reduction in the state space. Typically, this approximation leads to a trade-off between performance and the amount of required data [37, 40]. In this paper, we will assume the use of abstraction as a given, e.g., because the complete state space is too large to deal with. Nevertheless, we explore the trade-off in Section 3.4, where we compare the performance of the prototypical R-MAX algorithm [13] with and without abstraction.

In our setting, the agent acts in an MDP that returns states s, but instead of observing the true state s, the agent only observes abstract states $\phi(s)$ (see Figure 3.1). This setting, which has been considered before [39, 41],[1] is what we call *RL from Abstracted Observations (RLAO)*. Surprisingly, there are relatively few results for RLAO, even though many results for the planning setting are available [15, 16]. The main difference between these two settings is that in planning with abstraction the resulting problem can still be considered an MDP, but in RLAO, while the underlying problem is still an MDP, the observed problem is not.

The observation that the observed problem is not an MDP can be understood when we realize that RLAO corresponds to RL in a Partially Observable Markov

[1] We refer to Section 3.5 for a comparison with the related work.

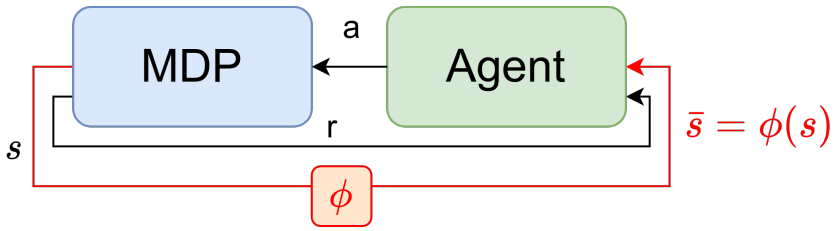


Figure 3.1: RL from Abstracted Observations, the agent receives the abstract state $\bar{s} = \phi(s)$ as an observation instead of the state s. Image based on Abel *et al.* [41].

Decision Process (POMDP) [19], as previously described [30]. Specifically, the abstraction function serves as an observation function. Rather than observing its true state s, the agent observes the abstract state $\phi(s)$ and its policy chooses an action based on this abstract state. It is well known that policies for POMDPs that only base their action on the last observation can be arbitrarily bad [132]. Fortunately, there is also good news, as this worst-case does not apply when ϕ is an *exact model similarity abstraction* [2] [82], because the resulting problem can be considered an MDP; this abstraction maps states to the same abstract state only when their reward and transition functions in the abstract space are the same [15]. We focus on the related *approximate model similarity abstraction* [16], which maps states to the same abstract state only when their reward and transition functions in the abstract space are close. Intuitively, because of its connection to the exact model similarity, one could expect that for this abstraction the worst-case also does not apply. However, as we discuss in detail in Section 3.2.2, MBRL methods typically use results that rely on the assumption of independent and identically distributed (i.i.d.) samples to prove efficient learning [14, 20, 127, 130]. This is not appropriate in RLAO: with abstraction, the transitions between abstract states need not be Markov, and the samples may depend on the history.

We analyze collecting samples in RLAO and prove that, with abstraction, samples are not guaranteed to be independent. This means that *most guarantees of existing MBRL methods do not hold in the RLAO setting.* [3] The primary technical result in this work shows that we can still learn an accurate model in RLAO by replacing concentration inequalities that rely on independent samples with a well-known concentration inequality for martingales [54]. This result allows us to extend the guarantees of MBRL methods to RLAO. We illustrate such an extension for the prototypical R-MAX algorithm [13], thus producing the first performance guarantees for model-based methods in RLAO. These results are important for the often adopted state abstraction framework, as they allow us to conclude under what cases performance guarantees in MBRL can be transferred to settings with state abstraction.

3.2. BACKGROUND

Section 3.3 will cover the combination of MBRL and abstraction in MDPs, in this section we introduce the required background.

3.2.1. MODEL-BASED RL

As is typical for RL problems, we assume the environment the agent is acting in can be represented by an infinite horizon MDP $M \triangleq \langle S, A, T, R, \gamma \rangle$ [74]. Here S is a finite set of states $s \in S$, A a finite set of actions $a \in A$, T a transition function

[2] Also known as stochastic bisimulation [75].

[3] Of course, certain guarantees on the combination of abstraction and RL are known. However, in most related work in abstraction settings (e.g., abstraction selection), the complication of samples not being independent does not occur due to particular assumptions [37, 48–52]. Section 3.5 gives details for individual papers.

$T(s'|s,a) = \Pr(s'|s,a)$, R a reward function $R(s,a)$ which gives the reward received when the agent executes action a in state s, and γ is a discount factor with $0 \leq \gamma \leq 1$ that determines the importance of future rewards. We use $R_{\max}$ to denote the maximum reward the agent can obtain in one step. The agent's goal is to find an optimal policy $\pi^* : S \to A$, i.e., a policy that maximizes the expectation of the cumulative reward in the MDP. $V^\pi(s)$ denotes the expected value of the cumulative reward under policy π starting from state s. Similarly, $Q^\pi(s,a)$ denotes the expected value of the cumulative reward when first taking action a from state s and then following policy π afterward.

MBRL methods learn a model from the experience that the agent gains by taking actions and observing the rewards it gets and the states it reaches. For a fixed state-action pair (s,a), we let $\tau_1, \tau_2, \cdots, \tau_{N(s,a)}$ be the first $N(s,a)$ time steps at which the agent took action a in state s. The first $N(s,a)$ states s' that the agent reached after taking action a in state s are stored as the sequence $Y_{s,a} \triangleq (s'^{(\tau_1+1)}, s'^{(\tau_2+1)}, \cdots, s'^{(\tau_{N(s,a)}+1)})$. We use Y to refer to the collection of all $Y_{s,a}$. Typically, in MBRL, the obtained experience is used to construct the empirical model T_Y [13, 14, 20, 127–130]. This model is constructed simply by counting how often the agent reached a particular next state s' and normalizing the obtained quantity by the total count:

$$\forall s' \in S: \quad T_Y(s'|s,a) \triangleq \frac{1}{N(s,a)} \sum_{i=1}^{N(s,a)} \mathbb{1}\{Y_{s,a}^{(\tau_i+1)} = s'\}. \tag{3.1}$$

Here $\mathbb{1}\{\cdot\}$ denotes the indicator function of the specified event, i.e., $\mathbb{1}\{Y_{s,a}^{(\tau_i+1)} = s'\}$ is 1 if $Y_{s,a}^{(\tau_i+1)} = s'$ and 0 otherwise.

3.2.2. GUARANTEES FOR MBRL

The quality of the empirical model T_Y is crucial for performance guarantees, irrespective of the form of the guarantee, e.g., PAC-MDP [14] or regret [20]. The quality of the empirical model is high when the distance between $T_Y(\cdot|s,a)$ and the ground truth $T(\cdot|s,a)$ is small. We can, for instance, measure this distance with the L_1 norm, defined as follows:

$$||T_Y(\cdot|s,a) - T(\cdot|s,a)||_1 \triangleq \sum_{s' \in S} |T_Y(s'|s,a) - T(s'|s,a)|. \tag{3.2}$$

Concentration inequalities are often used to guarantee that, with enough samples, this distance will be small, e.g.:

Lemma 3.1 (L_1 inequality [133])**.** Let $Y_{s,a} = Y_{s,a}^{(1)}, Y_{s,a}^{(2)}, \cdots, Y_{s,a}^{(N(s,a))}$ be i.i.d. random variables distributed according to $T(\cdot|s,a)$. Then, for all $\epsilon > 0$,

$$\Pr(||T_Y(\cdot|s,a) - T(\cdot|s,a)||_1 \geq \epsilon) \leq (2^{|S|} - 2) e^{-\frac{1}{2} N(s,a) \epsilon^2}. \tag{3.3}$$

These inequalities typically make use of the fact that samples are i.i.d. It is not necessarily evident that these bounds can be applied without problem. Let us

explore the transitions from a particular state, say state 42, in a Markov chain (we can ignore actions for this argument). Let k and l denote the time steps of two different visits to state 42. Without abstraction, the conditional distributions from which next states are sampled are identical. So the question now is if these are independent. That is, is it the case that:

$$P(S_{k+1}, S_{l+1}|S_k = 42, S_l = 42) = P(S_{k+1}|S_k = 42) * P(S_{l+1}|S_l = 42)? \tag{3.4}$$

We have that

$$P(S_{k+1}, S_{l+1}|S_k = 42, S_l = 42) = P(S_{k+1}|S_k = 42, S_l = 42)P(S_{l+1}|S_k = 42, S_k + 1, S_l = 42) \tag{3.5}$$

$$= P(S_{k+1}|S_k = 42, S_l = 42)P(S_{l+1}|S_l = 42) \text{ (due to the Markov property)} \tag{3.6}$$

So the question is if $P(S_{k+1}|S_k = 42, S_l = 42) = P(S_{k+1}|S_k = 42)$? In general, this is not the case, since the information that $S_l = 42$ gives information about what S_{k+1} was.

However, as shown for instance by Strehl and Littman [14], concentration inequalities for i.i.d. samples, such as Hoeffding's Inequality, can still be used as an upper bound in this case, because of the Markov property and the identical distributions of the samples. In this way, MBRL can upper bound the probability that the empirical model $T_Y(\cdot|s,a)$ will be far away ($\geq \epsilon$) from the actual model $T(\cdot|s,a)$. When the empirical model is accurate, a policy based on this model leads to near-optimal performance in the MDP M [13, 14, 20, 130].

3.2.3. STATE ABSTRACTION FOR KNOWN MODELS

We can formulate state abstraction as a mapping from states to abstract states [15]. This mapping is done with an abstraction function ϕ, a surjective function that maps from states $s \in S$ to abstract states $\bar{s} \in \bar{S}$: $\phi(s) : S \rightarrow \bar{S}$. We use the $\bar{}$ notation to refer to the abstract space and define $\bar{S}$ as $\bar{S} = \{\phi(s)|s \in S\}$. We slightly overload the definition of $\bar{s}$ to be able to write $s \in \bar{s}$. In this case, $\bar{s}$ is the set of states that map to $\bar{s}$, i.e., $\bar{s} = \{s \in S \mid \phi(s) = \bar{s}\}$. This form of state abstraction is general, and clusters states with different dynamics into abstract states. We assume that the state abstraction deterministically maps states to an abstract state. Since each state maps to precisely one abstract state and multiple states can map to the same abstract state, the abstract state space is typically (much) smaller than the original state space, $|\bar{S}| \leq |S|$.

We focus on a type of abstraction *approximate model similarity abstraction* [16], also known as approximate stochastic bisimulation [75, 134]. In this abstraction, two states can map to the same abstract state only if their behavior is similar in the abstract space, i.e., when the reward function and the transitions to abstract states are close. We can determine the transition probability to an abstract state $T(\bar{s}'|s,a)$ as:

$$T(\bar{s}'|s,a) = \sum_{s' \in \bar{s}'} T(s'|s,a). \tag{3.7}$$

Then, we can use (3.7) to define approximate model similarity abstraction:

Definition 3.1. An approximate model similarity abstraction, $\phi_{model,\eta_R,\eta_T}$, for fixed η_R, η_T, satisfies

$$\begin{aligned} \phi_{model,\eta}(s_1) = \phi_{model,\eta}(s_2) \implies \forall a \in A: \quad & |R(s_1,a) - R(s_2,a)| \leq \eta_R, \\ \forall \bar{s}' \in \bar{S}, a \in A: \quad & |T(\bar{s}'|s_1,a) - T(\bar{s}'|s_2,a)| \leq \eta_T. \end{aligned} \tag{3.8}$$

From now on, we will refer to $\phi_{\text{model},\eta_R,\eta_T}$ as ϕ. We note that this abstraction is still quite generic. It can cluster together states that have different transition and reward functions.

3

3.2.4. PLANNING WITH ABSTRACT MDPS

In the planning setting, where the model is known a priori, we can use the abstraction function ϕ to construct an abstract MDP. An abstract MDP can be helpful because it is smaller, making it easier to find a solution, and a solution for the abstract MDP can work well in the original MDP [15, 16]. We construct an abstract MDP $\bar{M}_\omega$ from the model of an MDP M, an abstraction function ϕ, and an action-specific weighting function ω. [4] The weighting function ω gives a weight to every state-action pair: $\forall s \in S,\ a \in A: 0 \leq \omega(s,a) \leq 1$. The weights of the state-action pairs associated with an abstract state $\bar{s}$ sum up to 1: $\sum_{s' \in \phi(s)} \omega(s',a) = 1$. We can use the weighting function to create an abstract transition and reward function, which are weighted averages of the original transition and reward functions. In this way, from M, ϕ, and any ω, we can *construct* an abstract MDP $\bar{M}_\omega$:

Definition 3.2 (Abstract MDP)**.** Given an MDP M, ϕ, and ω, an abstract MDP $\bar{M}_\omega = \langle \bar{S}, A, \bar{T}_\omega, \bar{R}_\omega \rangle$ is constructed as: $\bar{S} = \{\phi(s) \mid s \in S\}, A = A$,

$$\forall \bar{s} \in \bar{S},\ a \in A: \quad \bar{R}_\omega(\bar{s},a) \triangleq \sum_{s \in \bar{s}} \omega(s,a) R(s,a), \tag{3.9}$$

$$\forall \bar{s}, \bar{s}' \in \bar{S},\ a \in A: \quad \bar{T}_\omega(\bar{s}'|\bar{s},a) \triangleq \sum_{s \in \bar{s}} \sum_{s' \in \bar{s}'} \omega(s,a) T(s'|s,a). \tag{3.10}$$

Note that the abstract MDP $\bar{M}_\omega$ itself is an MDP. So we can use planning methods for MDPs to find an optimal policy $\bar{\pi}^*$ for $\bar{M}_\omega$. A desirable property of the approximate model similarity abstraction is that we can upper bound the difference between the optimal value V^* in M and the value $V^{\bar{\pi}^*}$ obtained when following the policy $\bar{\pi}^*$ in M. These bounds exists in different forms [16, 80, 135]. For completeness, we give these bounds for both the undiscounted finite horizon and the discounted infinite horizon:

Theorem 3.1. Let $M = \langle S, A, T, R \rangle$ be an MDP and $\bar{M} = \langle \bar{S}, A, \bar{T}, \bar{R} \rangle$ an abstract MDP, for some defined abstract transitions and rewards. We assume that

$$\forall \bar{s}, \bar{s}' \in \bar{S},\ s \in \bar{s},\ a \in A: |\bar{T}(\bar{s}'|\bar{s},a) - \Pr(\bar{s}'|s,a)| \leq \eta_T \tag{3.11}$$

$$\text{and } |\bar{R}(\bar{s},a) - R(s,a)| \leq \eta_R. \tag{3.12}$$

[4] The action-specific weighting function is more general than the typically used weighting function, which is not action-specific and only depends on the state s [15]. More formally, it is the case where $\forall a, a' \in A,\ s \in S: \omega(s,a) = \omega(s,a')$.

Then, for a finite horizon problem with horizon h we have:

$$V^*(s) - V^{\bar{\pi}^*}(s) \leq 2h\eta_R + (h+1)h\eta_T|\bar{S}|R_{\max}. \tag{3.13}$$

And for a discounted infinite horizon problem with discount γ we have:

$$V^*(s) - V^{\bar{\pi}^*}(s) \leq \frac{2\eta_R}{1-\gamma} + \frac{2\gamma\eta_T|\bar{S}|R_{\max}}{(1-\gamma)^2}. \tag{3.14}$$

The proof of Theorem 3.1 is in Appendix 3.8.1. These bounds show that an optimal abstract policy $\bar{\pi}^*$ for $\bar{M}$ can also perform well in the original problem M when the approximate errors η_R and η_T are small. They hold for any abstract MDP $\bar{M}$ created from an approximate model similarity abstraction ϕ and any valid weighting function ω.

3.3. MBRL From Abstracted Observations

In RLAO, we have an abstraction function ϕ and instead of observing the true state s, the agent observes the abstract state $\phi(s)$. In contrast to the planning setting in Section 3.2.3, here we act in an MDP M of which we do *not* know the transition and reward functions. As mentioned in the introduction, there are surprisingly few results for the RLAO setting (Section 3.5 discusses special cases people have considered). Specifically, results of MBRL from Abstracted Observations (MBRLAO) are lacking. Section 3.3.2 explains why this is by analyzing how abstraction leads to dependence between samples, which means that the methods for dealing with Markov transitions, as covered in Section 3.2.2, no longer suffice. Then, in Section 3.3.4, we show how concentration inequalities for martingales can be used to still learn an accurate model in RLAO. To illustrate how this result can be used to extend the results of MBRL methods to RLAO, we extend the results of the R-MAX algorithm [13]. R-MAX is a well-known and straightforward method that guarantees sample efficient learning.

3.3.1. The General MBRL From Abstracted Observations Approach

In RLAO, the agent collects data for every abstract state-action pair $(\bar{s}, a)$, stored as sequences $\bar{Y}_{\bar{s},a}$:

$$\bar{Y}_{\bar{s},a} : \{\bar{s}'^{(\tau_1+1)}, \bar{s}'^{(\tau_2+1)}, \cdots, \bar{s}'^{(\tau_{N(\bar{s},a)}+1)}\}. \tag{3.15}$$

Like in (3.1), we construct an empirical model $\bar{T}_Y$, now looking at the abstract next-states that the agent reached:

$$\bar{T}_Y(\bar{s}'|\bar{s}, a) \triangleq \frac{1}{N(\bar{s}, a)} \sum_{i=1}^{N(\bar{s},a)} \mathbb{1}\{\bar{Y}_{\bar{s},a}^{(i)} = \bar{s}'\}. \tag{3.16}$$

Suppose we could guarantee that the empirical model $\bar{T}_Y$ was equal, or close, to the transition function $\bar{T}_\omega$ of an abstract MDP $\bar{M}_\omega$ constructed from the true MDP

with ϕ and a valid ω. In that case, we could bound the loss in performance due to applying the learned policy $\bar{\pi}^*$ to M instead of applying the optimal policy π^* [16, 80]. Our main question is: do the finite-sample model learning guarantees of MBRL algorithms still hold in the RLAO setting?

3.3.2. Requirements for guarantees for MBRL From Abstracted Observations

In order to give guarantees, we need to show that the empirical model $\bar{T}_Y$ is close to the transition model of an abstract MDP $\bar{M}_\omega$. Before defining this transition model of $\bar{M}_\omega$, we examine the data collection. In the online data collection, the agent obtains a sample for $\bar{Y}_{\bar{s},a}$ when it is in a state $s \in \bar{s}$ and takes action a. Specifically, the agent obtains the i-th sample $\bar{Y}^{(i)}_{\bar{s},a} = \bar{s}'^{\tau_1+1}$ from state $X^{(i)}_{\bar{s},a} = s^{\tau_i} \in \bar{s}$:

$$\bar{Y}^{(i)}_{\bar{s},a} \sim T(\cdot | X^{(i)}_{\bar{s},a} = s^{\tau_i}, a). \tag{3.17}$$

Let $X_{\bar{s},a} = (X^{(i)}_{\bar{s},a})_{i=1}^{N(\bar{s},a)}$ denote the sequence of states $s \in \bar{s}$ from which the agent took action a. Each state s gets a weight according to how often it appears in $X_{\bar{s},a}$, which we formalize with the weighting function ω_X:

$$\forall s \in \bar{s}, a \in A: \quad \omega_X(s,a) \triangleq \frac{1}{N(\bar{s},a)} \sum_{i=1}^{N(\bar{s},a)} \mathbb{1}\{X^{(i)}_{\bar{s},a} = s\}. \tag{3.18}$$

We use ω_X to define $\bar{T}_{\omega_X}$ analogous to (3.10):

$$\forall \bar{s}, \bar{s}' \in \bar{S}, a \in A: \quad \bar{T}_{\omega_X}(\bar{s}' | \bar{s}, a) \triangleq \sum_{s \in \bar{s}} \omega_X(s,a) \sum_{s' \in \bar{s}'} T(s' | s, a). \tag{3.19}$$

To highlight the close connection between $\bar{T}_{\omega_X}$ and $\bar{T}_Y$ (build of samples from $T(\cdot | X^{(i)}_{\bar{s},a} = s^{\tau_i}, a)$), we give a second, but equivalent, [5] definition of $\bar{T}_{\omega_X}$:

$$\forall (\bar{s}, a), \bar{s}': \quad \bar{T}_{\omega_X}(\bar{s}' | \bar{s}, a) \triangleq \frac{1}{N(\bar{s},a)} \sum_{i=1}^{N(\bar{s},a)} T(\bar{s}' | X^{(i)}_{\bar{s},a}, a). \tag{3.20}$$

Note that ω_X and thus $\bar{T}_{\omega_X}$ are not fixed a priori. Instead, like $\bar{T}_Y$, they are empirical quantities that change at every time step and depend on the policy and the (stochastic) outcomes. Importantly, by its definition, ω_X is a valid ω at every timestep. It is not a problem that ω_X and $\bar{T}_{\omega_X}$ change over time, as long as the empirical model $\bar{T}_Y$ can be shown to be close to $\bar{T}_{\omega_X}$. For this, we want a concentration inequality to provide bounds on the deviation of the empirical model $\bar{T}_Y$ from $\bar{T}_{\omega_X}$; we refer to this inequality as the abstract L1 inequality, similar in form to (3.3):

$$P(|\bar{T}_Y(\cdot | \bar{s}, a) - \bar{T}_{\omega_X}(\cdot | \bar{s}, a)|_1 \geq \epsilon) \leq \delta, \tag{3.21}$$

where $\bar{T}_Y(\cdot | \bar{s}, a)$ is defined according to (3.16) and $\bar{T}_{\omega_X}$ according to (3.19).

[5] In the proof of Theorem 3.2 we show that these two definitions are equivalent.

3.3.3. WHY THE PREVIOUS STRATEGY FAILS: DEPENDENT SAMPLES THAT ARE NOT IDENTICALLY DISTRIBUTED

Suppose we could directly obtain i.i.d. samples from $\bar{T}_{\omega_X}$ and base our empirical model $\bar{T}_Y$ on the obtained samples. In that case, we could show that the abstract L1 inequality holds by applying Lemma 3.1. This lemma would be applicable because we could obtain a number $N(\bar{s}, a)$ of i.i.d. samples per abstract state-action pair, distributed according to $\bar{T}_{\omega_X}(\cdot|\bar{s}, a)$. However, the samples are not i.i.d. in RLAO: the samples are neither identically distributed nor independent, and this combination means that previous techniques fail. We will first cover the distribution of the samples and show that samples not being identically distributed is not a problem. Then we prove that samples are not guaranteed to be independent. Afterward, Section 3.3.4 shows that we can still learn when the samples are dependent.

WHY WE CAN NOT USE LEMMA 3.1: DEPENDENT SAMPLES.

The samples are not necessarily identically distributed in RLAO since the agent obtains a sample $\bar{Y}^{(i)}$ when taking action a from state $X^{(i)}_{\bar{s},a} = s \in \bar{s}$, as in (3.17). If $X^{(i)}_{\bar{s},a} \neq X^{(j)}_{\bar{s},a}$, these states can have different transition distributions. This implies that in general we might not be able to apply Lemma 3.1, because it assumes identically distributed random variables. However, different distributions by themselves need not be a problem; we show that the result also holds when the random variables are not identically distributed:

Lemma 3.2. Let $X_{\bar{s},a} = s_1, \cdots, s_m$ be a sequence of states $s \in \bar{s}$ and let $\bar{\boldsymbol{Y}}_{\bar{s},a} = \bar{Y}^{(1)}, \bar{Y}^{(2)}, \cdots, \bar{Y}^{(m)}$ be independent random variables distributed according to $\Pr(\cdot|s_1, a), \cdots, \Pr(\cdot|s_m, a)$ (3.7). Then, for all $\epsilon > 0$,

$$\Pr(||\bar{T}_Y(\cdot|\bar{s}, a) - \bar{T}_{\omega_X}(\cdot|\bar{s}, a)||_1 \geq \epsilon) \leq (2^{|\bar{S}|} - 2)e^{-\frac{1}{2}m\epsilon^2}. \tag{3.22}$$

The proof can be found in Appendix 3.8.2. Therefore, if the samples in RLAO were independent, then we could apply Lemma 3.2 to guarantee an accurate model.

Independence. One could be tempted to assume the samples are independent, i.e.,

$$\forall \bar{s}'_1, \cdots, \bar{s}'_m \in (\bar{S})^m : \quad \Pr(\bar{Y}^{(1)}_{\bar{s},a} = \bar{s}'_1, \cdots, \bar{Y}^{(m)}_{\bar{s},a} = \bar{s}'_m) = \Pr(\bar{Y}^{(1)}_{\bar{s},a} = \bar{s}'_1) \cdots P(\bar{Y}^{(m)}_{\bar{s},a} = \bar{s}'_m). \tag{3.23}$$

However, this is not true in general in RLAO:

Observation 3.1. When collecting samples online using an abstraction function, such samples are not necessarily independent.

Samples can be dependent when 1) samples are collected online in the real environment, of which we do not know the transitions, and 2) the samples are collected for abstract states $\bar{s}$. Observation 3.1 can be understood from the perspective that the RLAO problem corresponds to RL in a POMDP [30]. The corresponding POMDP uses the abstraction function as the observation function and the abstract states as observations. Since the transitions between observations

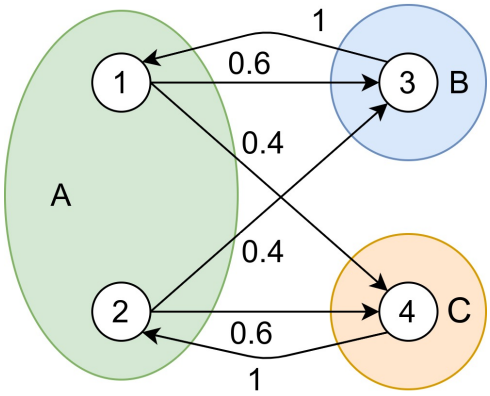


Figure 3.2: Simple MDP, with only 1 action, and abstraction. The small circles are states (1,2,3,4). A, B and C are the abstract states. The arrows show the transition probabilities, e.g. $P(3|1) = 0.6$.

need not be Markov in POMDPs, the samples from abstract states can depend on the history. While this observation may be clear from the POMDP perspective, work in RLAO regularly assumes (explicitly or implicitly) that independent samples can somehow be obtained [37, 39, 40, 51]. In the following counterexample, we rigorously show that samples are not necessarily independent.

Counterexample. We use the example MDP and abstraction in Figure 3.2, where we have four states, three abstract states, and only one action. Since the example MDP has only one action, we omit the action from the notation. We examine the transition function of abstract state A, $\bar{T}_Y(\cdot|A)$ and consider the first two times we transition from A. These two transition samples, $\bar{s}'_1$ and $\bar{s}'_2$, are the first two entries in $\bar{Y}_A$. We show that the samples are not independent for at least one combination of $\bar{s}'_1$ and $\bar{s}'_2$.

Let $\bar{s}'_1 = \bar{s}'_2 = B$, i.e., the first two times we experience a transition from the abstract state A, we end up in B. We denote the i-th experienced transition from abstract state A as $\bar{Y}_A^{(i)}$. Let state 1 be the starting state.

We start with the product of the probabilities:

$$\Pr(\bar{Y}_A^{(1)} = B) \Pr(\bar{Y}_A^{(2)} = B). \tag{3.24}$$

We have $\Pr(\bar{Y}_A^{(1)} = B) = \Pr(B|1) = 0.6$ for the first term since state 1 is the starting state. The second term is more complex since it includes the probability of starting the transition from state 1 and state 2.

We have:

$$\Pr(\bar{Y}_A^{(2)} = B) = \sum_{\bar{s}\in\bar{S}} \Pr(\bar{Y}_A^{(2)} = B | \bar{Y}_A^{(1)} = \bar{s}) \Pr(\bar{Y}_A^{(1)} = \bar{s}) \tag{3.25}$$

$$= \Pr(\bar{Y}_A^{(2)} = B | \bar{Y}_A^{(1)} = A) \Pr(\bar{Y}_A^{(1)} = A) + \Pr(\bar{Y}_A^{(2)} = B | \bar{Y}_A^{(1)} = B) \Pr(\bar{Y}_A^{(1)} = B)$$

$$+ \Pr(\bar{Y}_A^{(2)} = B | \bar{Y}_A^{(1)} = C) \Pr(\bar{Y}_A^{(1)} = C). \tag{3.26}$$

$$= \Pr(\bar{Y}_A^{(2)} = B | \bar{Y}_A^{(1)} = B) \Pr(\bar{Y}_A^{(1)} = B) + \Pr(\bar{Y}_A^{(2)} = B | \bar{Y}_A^{(1)} = C) \Pr(\bar{Y}_A^{(1)} = C). \tag{3.27}$$

$$= \Pr(Y_A^{(2)} = 3 | \bar{Y}_A^{(1)} = 3) \Pr(Y_A^{(1)} = 3) + \Pr(Y_A^{(2)} = 3 | Y_A^{(1)} = 4) \Pr(Y_A^{(1)} = 4) \tag{3.28}$$

$$= 0.6 \cdot 0.6 + 0.4 \cdot 0.4 = 0.52. \tag{3.29}$$

For the step from (3.26) to (3.27), $\Pr(\bar{Y}_A^{(1)} = A)$ is 0 because there is no transition from a state in A to a state in A. Then, from (3.27) to (3.28), we use that both abstract states B and C consist of exactly 1 state. So, e.g., $\Pr(\bar{Y}_A^{(2)} = B | \bar{Y}_A^{(1)} = B) = \Pr(Y_A^{(2)} = 3 | \bar{Y}_A^{(1)} = 3)$. So, for the product of the probabilities, we end up with: $\Pr(\bar{Y}_A^{(1)} = B) \Pr(\bar{Y}_A^{(2)} = B) = 0.6 \cdot 0.52 = 0.321$.

For the joint probability, we have:

$$Pr(\bar{Y}_A^{(1)} = B, \bar{Y}_A^{(2)} = B) = \Pr(\bar{Y}_A^{(1)} = B) \Pr(\bar{Y}_A^{(2)} = B | \bar{Y}_A^{(1)} = B) \tag{3.30}$$

$$= \Pr(B|1)(\Pr(B|1) \Pr(1|B)) \tag{3.31}$$

$$= 0.6 \cdot (0.6 \cdot 1) \tag{3.32}$$

$$= 0.6 \cdot 0.6 \tag{3.33}$$

$$= 0.36. \tag{3.34}$$

Here, $\Pr(\bar{Y}_A^{(2)} = B | \bar{Y}_A^{(1)} = B) = \Pr(B|1) \Pr(1|B)$ because the first transition ends in state B and we always transition to state 1 from state B. Hence, $\Pr(\bar{Y}_A^{(2)} = B | \bar{Y}_A^{(1)} = B) = \Pr(B|1) \Pr(1|B) = 0.6 \cdot 1$.

Combining the joint probability and the product of probabilities, we end up with:

$$0.36 = \Pr(\bar{Y}_A^{(1)} = B, \bar{Y}_A^{(2)} = B) \neq \Pr(\bar{Y}_A^{(1)} = B) \Pr(\bar{Y}_A^{(2)} = B) = 0.6 \cdot 0.52. \tag{3.35}$$

Thus, the samples are not independent. Leading us to the second observation.

Observation 3.2. As independence cannot be guaranteed, Lemmas 3.1 and 3.2 cannot be readily applied to show that the abstract L1 inequality holds.

This claim follows from the fact that Lemmas 3.1 and 3.2 both use the assumption of independence in their proofs. It would still be possible to obtain independent samples if we could, for example, have access to a simulator of the problem. In that case, it is still possible to give guarantees on the accuracy of the model, which we show in Appendix 3.8.6. However, we consider the setting where a simulator is not available.

Why the Approach by Strehl and Littman [14] Fails

While the counterexample above is informative as to why Lemmas 3.1 and 3.2 cannot be applied, the failure to apply these lemmas may not come as a surprise:

in the end, as shown by Strehl and Littman [14], more work is needed. They are able to use these concentration inequalities due to an additional proof that shows that even though the samples are drawn from a Markov chain, and thus not fully independent, the inequality still serves as an upper bound. This raises the question whether we could not follow the same approach, and show that Lemma 3.1 (or 3.2) is still an upper bound in the RLAO setting.

It turns out that this is not possible, as that result uses the Markov property and requires each sample to be identically distributed. Without abstraction, only (s,a) and the next states $s' \sim P(\cdot|s,a)$ are considered, which indeed have the same distribution. In RLAO, the outcomes of multiple states are grouped together and for a pair $(\bar{s},a)$ both the state $s \in \bar{s}$ that we reach and the resulting next state $\bar{s}'$ need to be considered. Since the distributions $s' \sim P(\cdot|s_1,a)$ and $s' \sim P(\cdot|s_2,a)$ of two states $s_1, s_2 \in \bar{s}$ do not have to be the same, these samples are not guaranteed to be identically distributed.

Summary: Why Previous Strategies Fail

Summarizing, we have seen that previous strategies fail due to the combination of samples neither being independent, nor being identically distributed. We showed that if the samples would only be non-identically distributed (but independent) we could modify the proof of Lemma 3.1, leading to Lemma 3.2, that could be directly used. On the other hand, if the samples were only dependent (but still identically distributed), it would be possible to follow the strategy of Strehl and Littman [14]. However, given that we are dealing with the dependent non-identically distributed setting, neither of these previous strategies work, and a new approach is needed, as we present next.

3.3.4. Guarantees for Abstract Model Learning Using Martingales

Now we want to give a guarantee in the form of the abstract L1 inequality from (3.21).[6] In Section 3.3.2, we found this was not possible with concentration inequalities such as Hoeffding's inequality because the samples are not guaranteed to be independent. Here we consider a related bound for weakly dependent samples, the Azuma-Hoeffding inequality. This inequality makes use of the properties of a martingale difference sequence, which are slightly weaker than independence:

Definition 3.3 (Martingale difference sequence [54])**.** The sequence $Z_1, Z_2, \cdots$ is a martingale difference sequence if, $\forall i$, it satisfies the following conditions:

$$E[Z_i|Z_1, Z_2, \cdots, Z_{i-1}] = 0,$$
$$|Z_i| < \infty.$$

The properties of the martingale difference sequence can be used to obtain the following concentration inequality:

[6] We focus on the transition function, for the reward function we make some simplifying assumptions in Section 3.4. We discuss in Section 3.6 how these assumptions can be relaxed and the result extended for the reward function.

Lemma 3.3 (Azuma-Hoeffding Inequality [53, 54])**.** If the random variables $Z_1, Z_2, \cdots$ form a martingale difference sequence (Def. 3.3), with $|Z_i| \leq b$, then

$$\Pr(\sum_{i=1}^{n} Z_i > \epsilon) \leq e^{-\frac{\epsilon^2}{2b^2 n}}. \tag{3.36}$$

Our main result, Theorem 3.2, shows that we can use Lemma 3.3 to obtain a concentration inequality for the abstract transition function in RLAO (as in (3.21)). Specifically, we show that, with high probability, the empirical abstract transition function $\bar{T}_Y$ will be close to the abstract transition function $\bar{T}_{\omega_X}$:

Theorem 3.2 (Abstract L1 inequality)**.** If an agent has access to a state abstraction function ϕ and uses this to collect data for any abstract state-action pair $(\bar{s}, a)$ by acting in an MDP M according to a policy $\bar{\pi}$, we have that the following holds with a probability of at least $1-\delta$ for a fixed value of $N(\bar{s}, a)$:

$$||\bar{T}_Y(\cdot|\bar{s}, a) - \bar{T}_{\omega_X}(\cdot|\bar{s}, a)||_1 \leq \epsilon, \tag{3.37}$$

where we use the definitions of $\bar{T}_Y(\cdot|\bar{s}, a)$ and $\bar{T}_{\omega_X}(\cdot|\bar{s}, a)$ in (3.16) and (3.19), respectively, and where $\delta = 2^{|\bar{S}|} e^{-\frac{1}{8} N(\bar{s},a)\epsilon^2}$.

This theorem shows that the empirical model constructed by MBRLAO is close to an abstract MDP $\bar{M}_{\omega_X}$, and here Theorem 3.1 gives performance loss guarantees. By assuming that $\bar{M}_{\omega_X}$ is the results of an approximate model irrelevance abstraction, we can give end to end guarantees. In simpler words: our result just shows that whatever $\bar{T}_Y$ you might end up with (indeed, regardless of changing policies, etc.), it was generated by some underlying states X, and the implied $\bar{T}_{\omega_X}$ will concentrate on $\bar{T}_Y$.

Note that, unlike in planning with abstract MDPs (Section 3.2.4), there is no fixed set of weights $\bar{T}_{\omega_X}$ that can be used as ground truth that needs to be estimated. As illustrated in Section 3.3.3, the RLAO setting corresponds to a POMDP, which means that *depending on the history* there would be a different distribution over the states (and thus different weights) in each abstract state (called 'the belief' in a POMDP). Instead, both $\bar{T}_{\omega_X}$ and $\bar{T}_Y$ change over time. We show in the proof of Theorem 3.2 (in Appendix 3.8.3) that $\bar{T}_Y$ will concentrate on $\bar{T}_{\omega_X}$ as they are intimately connected. This is possible because Lemma 3.3 can be applied as long as the Z_i form a martingale difference sequence, with $|Z_i| \leq b$. In the proof, we define a suitable Z_i and show that $\bar{T}_Y$ will thus concentrate on $\bar{T}_{\omega_X}$, with high probability.

We demonstrated that performance guarantees of MBRL methods can be extended to the setting with abstracted observations using an approximate model-similarity abstraction. We also claim that similar results can be obtained for at least one other type of abstraction:

Claim 3.1. The analysis and result of Theorem 3.4 can be extended to hold for approximate Q^* abstractions.

A proof sketch can be found in Appendix 3.1. This claim demonstrates that approximate state abstractions can generally be used to find a good policy, as long as the Q^* values are similar and an optimal policy for an abstract model performs well in the original problem.

3.4. AN ILLUSTRATION: R-MAX FROM ABSTRACTED OBSERVATIONS

Here we give an illustration of how we can use Theorem 3.2 to provide guarantees for MBRL methods in RLAO with an *approximate model similarity abstraction*. We illustrate this using the R-MAX algorithm [13]. We start with a short description of R-MAX and how it operates with abstraction.

The R-MAX algorithm maintains a model of the environment. It uses this model to compute a policy periodically and then follows this policy for several steps. Initially, all the state-action pairs are *unknown* and the algorithm optimistically initializes their reward and transition functions: $R(s,a) = R_{\max}$ (the maximum reward), $T(s|s,a) = 1$, and $\forall s' \neq s : T(s'|s,a) = 0$. This initialization means that, in the model the algorithm maintains, these unknown (s,a) lead to the maximum reward, hence the name R-MAX. A state-action pair's transition and reward function are only updated once they have been visited sufficiently often, at which point the state-action pair is considered *known*. Together, this ensures that the algorithm explores sufficiently. During execution, the algorithm operates in episodes of n-steps. At the start of every episode it calculates an optimal n-step policy and follows this for n timesteps, or until a state-action pair becomes known. Once all the state-action pairs are known it calculates the optimal policy for the final model and then runs this indefinitely. The algorithm has the following guarantee:

Theorem 3.3 (R-MAX in MDPs without abstraction [13])**.** Given an MDP M, with $|S|$ states and $|A|$ actions, and inputs ϵ and δ. With probability of at least $1-\delta$ the R-MAX algorithm will attain an expected average return of $\text{Opt}(\prod_M(\epsilon, T_\epsilon)) - 2\epsilon$ within a number of steps polynomial in $|S|, |A|, \frac{1}{\epsilon}\frac{1}{\delta}, T_\epsilon$. Where T_ϵ is the ϵ-return mixing time of the optimal policy, the policies for M whose ϵ-return mixing time is T_ϵ are denoted by $\prod_M(\epsilon, T_\epsilon)$, the optimal expected T_ϵ-step undiscounted average return achievable by such policies are denoted by $\text{Opt}(\prod_M(\epsilon, T_\epsilon))$.

Here T_ϵ is the ϵ-return mixing time of a policy π, it is the minimum number of steps needed to guarantee that the expected average return is within ϵ of the optimal expected average return [136].

For R-MAX from Abstracted Observations, we make the following assumptions that stem from the original analysis: we assume that the MDP is ergodic [74], [7] that we know S and A, that the reward function is deterministic, and that we know the minimum and maximum reward. W.l.o.g., we assume the rewards are between 0 and $R_{\max}$, with $0 < R_{\max} < \infty$. We add the assumption that the agent has access to an approximate model similarity abstraction function ϕ and that each state in an abstract state has the same reward function. [8]

Algorithm 1 shows the procedure for R-MAX from Abstracted Observations. It follows the same steps as the original algorithm, except that it makes use of an abstraction function ϕ and maintains an abstract model. As in the original, the

[7] An ergodic, or recurrent, MDP is an MDP where every state is recurrent under every stationary policy, i.e., asymptotically, every state will be visited infinitely often [74].

[8] Note that this is just a slight simplification as any empirical estimate $\bar{R}$ is guaranteed to be within η_R of any $\bar{R}_\omega$, under the assumption that the rewards are deterministic.

Algorithm 1 Procedure: R-MAX from Abstracted Observations

Input: $\phi, \delta, \epsilon, T_\epsilon$
for all $(\bar{s}, a) \in \bar{S} \times A$ **do**
  $\bar{T}_Y(\bar{s}|\bar{s}, a) = 1$
  $\bar{R}_Y(\bar{s}, a) = R_{\max}$
  $\bar{Y}_{\bar{s},a} = [\,]$
end for
$\bar{M}_Y = \langle \bar{S}, A, \bar{T}_Y, \bar{R}_Y \rangle$
Select m, the number of samples required per (abstract) state-action pair to make them known.
// While there is still an unknown state-action pair.
while $\min_{(\bar{s},a)} |Y_{\bar{s},a}| < m$ **do**
  Compute optimal T_ϵ-step policy $\bar{\pi}$ in $\bar{M}_Y$ for the current abstract state.
  for T_ϵ timesteps **do**
    $\bar{s} = \phi(s)$
    $a = \bar{\pi}(\bar{s})$
    $s', r =$ Step(s, a)
    $s = s'$
    if $|\bar{Y}_{\bar{s},a}| < m$ **then**
      $\bar{Y}_{\bar{s},a}$.append$(\phi(s'))$
      if $|\bar{Y}_{\bar{s},a}| = m$ **then**
        // State-action pair has become known.
        for all $\bar{s}' \in \bar{S}$ **do**
          $\bar{T}_Y(\bar{s}'|\bar{s}, a) = \frac{1}{m} \sum_{i=1}^{m} \mathbb{1}\{\bar{Y}^{(i)}_{\bar{s},a} = \bar{s}'\}$
        end for
        $\bar{R}_Y(\bar{s}, a) = r$
        break
      end if
    end if
  end for
end while
Compute optimal policy $\bar{\pi}^*$ for $\bar{M}$ and run indefinitely.

input to the algorithm is the allowed failure probability δ, the error bound ϵ, and the ϵ-return mixing time T_ϵ of an optimal policy. We add the abstraction function ϕ as a new input. The algorithm uses this function to observe $\phi(s)$, as in Figure 3.1, and it builds an empirical (abstract) model from the observations it obtains.

Because the algorithm uses an abstraction function ϕ, we cannot guarantee the ϵ error bound. However, with Theorem 3.4 we can still guarantee an error bound that is a function of ϵ and the error η of the abstraction, thus providing the first finite-sample guarantees for RLAO:

Theorem 3.4. Given an MDP M, an approximate model similarity abstraction ϕ, with η_R and η_T, and inputs $|\bar{S}|, |A|, \epsilon, \delta, T_\epsilon$. With probability of at least $1-\delta$ the R-MAX algorithm adapted to abstraction (Algorithm 1) will attain an expected average

return of $\text{Opt}(\prod_M(\epsilon, T_\epsilon)) - 3g(\eta_T, \eta_R) - 2\epsilon$ within a number of steps polynomial in $|\bar{S}|, |A|, \frac{1}{\epsilon}\frac{1}{\delta}, T_\epsilon$. Where T_ϵ is the ϵ-return mixing time of the optimal policy, the policies for M whose ϵ-return mixing time is T_ϵ are denoted by $\prod_M(\epsilon, T_\epsilon)$, the optimal expected T_ϵ-step undiscounted average return achievable by such policies are denoted by $\text{Opt}(\prod_M(\epsilon, T_\epsilon))$, and

$$g(\eta_T, \eta_R) = T_\epsilon \eta_R + \frac{(T_\epsilon - 1)T_\epsilon}{2}\eta_T|\bar{S}|.$$

The proof can be found in Appendix 3.8.5 and follows the line of the original R-MAX proof, using the assumptions mentioned at the start of Section 3.3. To translate the results to the RLAO setting, we first use the Abstract L1 inequality (Theorem 3.2) to show that the empirical abstract model is accurate with high probability. Then the performance bounds from Theorem 3.1 can be used to bound the loss in performance by using an abstract policy based on the empirical abstract model in the MDP M instead of the optimal (ground) policy π^*. These bounds hold for *any* ω_X as long as ω_X is a valid weighting function. That ω_X will be a valid weighting function follows from its definition in (3.18). Because Theorem 3.1 allows us to directly bound the loss in performance for using an abstract policy, based on an abstract empirical model, in the original problem M, the amount of steps is polynomial only in $|\bar{S}|$ instead of $|S|$.

As is typical with abstraction, there is a trade-off between the performance and the required number of steps: a coarser abstract model can potentially learn much faster but could sacrifice optimality, while a non-abstract model might have the best performance in the limit of infinite experience. We can see this trade-off in the results of Theorems 3.3 and 3.4. When we directly model M without abstraction, Theorem 3.3 shows that the algorithm will attain an expected return of $\text{Opt}(\prod_M(\epsilon, T_\epsilon)) - 2\epsilon$ within a number of steps polynomial in $|S|, |A|, \frac{1}{\epsilon}\frac{1}{\delta}, T_\epsilon$. Theorem 3.4 shows that, when we use an approximate model similarity abstraction to learn an abstract model, this leads to an additional performance loss of $3g(\eta_T, \eta_R)$ due to the approximation. However, the advantage of using the abstraction is that the number of steps within which this is achieved is polynomial only in the size of the abstract space $|\bar{S}|$ rather than the (larger) original state space $|S|$. Thus, these results show that the performance is not arbitrarily bad with approximate model similarity abstraction. Moreover, when the abstraction errors (η_T and η_R) are small and the reduction in state space is large, abstraction helps to reach near-optimal performance significantly faster.

3.5. RELATED WORK

The problem we resolved in this paper may seem intuitive, but as we will make clear here, it is a fundamental problem in rich literature. Many studies have considered the combination of abstraction with either planning or RL. Some studies avoid, or ignore, the issue of dependency by simply assuming that samples are independent [37, 40, 42, 137]. Others avoid it by looking at convergence in the limit [34, 50, 98] or by assuming access to an MDP model [48, 49, 51].

RL With Abstraction. A negative result has been provided in the RLAO setting, showing that R-MAX [13] no longer maintains its guarantees when paired with any state abstraction function [41]. For this negative result, they give an example that uses approximate Q^* similarity abstractions [16]. Our counterexample is more powerful: indicating problems with the analysis even for approximate model similarity abstractions (an approximate model similarity abstraction is also an approximate Q^* abstraction, but not the other way around). Nevertheless, our second result shows that it is still possible to give guarantees in RLAO for R-MAX-like algorithms when we use an approximate model similarity abstraction and take the η_R and η_T inaccuracies into account.

Another study considered a setting related to abstraction, where the transition and reward functions can change over time, either abruptly or gradually [138]. The reward and transition probabilities depend on the timestep t, so $T(s'|s,a,t)$ instead of $T(s'|s,a)$. They bound the variation in the reward and transition functions over time. By taking the variation over time into account they are able to give results. In their setting, the MDP is fixed given the timestep. However, in RLAO this is not fixed. Each time the transition function at a timestep t could be different.

Recently regret guarantees have been found for the episodic (continuous) RL setting [139]. Their abstraction method learns an abstraction adaptively. The model-based version of their algorithm requires that the state and actions spaces are embedded in compact metric spaces. In this way, they can define a measure of the difference between state-action pairs in these metric spaces. However, they require oracle access to this metric. In our setting this would require knowing the transition and reward functions, which we assume we do not.

Abstraction Selection. There are quite a few studies in the area of *abstraction selection*, where the agent has access to a set of abstraction functions (state representations)[39, 48, 49, 51, 104]. Several studies assume that the given set of state representations contains at least one Markov model [48, 49, 51]. One study gives asymptotic guarantees for selecting the correct model and building an exact MDP model [48]. The assumption that an MDP model exists in the given set of representations is crucial in their analysis since the samples are i.i.d. for this MDP model. Similarly, other studies also assume that the given set of state representations contains a Markov model [49, 51]. They create an algorithm for which they obtain regret bounds, and their analysis also uses the Markov representation.

Some studies in abstraction selection do not assume the given abstraction functions contain a Markov model [39, 104]. Lattimore, Hutter, and Sunehag [104] deal with a more general setting where the problem may be non-Markovian. Instead of assuming access to a set of abstractions, they assume access to a set of models, including a model of the true environment. Since they are given models, they do not focus on learning them, making it very different from our setting. By observing the rewards obtained while executing a policy they are able to exclude unlikely models, and eventually find the true model of the environment. The other study [39] uses Theorem 2.1 from Weissman *et al.* [133], which requires i.i.d. samples. We have shown that independent samples *cannot* be guaranteed in RLAO.

MDPs With Rich Observations. Other related work is in MDPs with rich observations or block structure [35, 52, 109]. In that setting, each observation can only be generated from a *single* hidden state, which means that the issue of non-i.i.d. data due to abstraction does not arise. We can view the rich observation setting as an aggregation problem, where the observations can be aggregated to form a small (latent) MDP [109]. Their setting is related to exact model similarity (or bisimulation) [52]. In contrast, in RLAO, each observation can be generated from multiple hidden states, and we do not try to learn the MDP, as it is not small. Furthermore, we focus on approximate model similarity, which introduces the problems as described in Section 3.3.2.

I.I.D. Samples. One way to avoid the issue of dependent samples is by assuming that samples are obtained independently [37, 40, 42, 137]. One study considers the setting with a continuous domain where we are given a data set with i.i.d. samples [37]. They use discretization to aggregate states into abstract states and give a guarantee that, with a high probability, the model will be ϵ-accurate given a fixed data set. While they assume that the data has been gathered i.i.d., our results show that martingale concentration inequalities could be used to extend their results to the online data collection in the RLAO setting. Discretization has been used in another study in a continuous space [42]. They search for a solution for a linear dynamical system, where the transitions are deterministic, except for an additive noise component. They assume this noise is distributed i.i.d. and try to learn the resulting abstract transition functions by iteratively sampling N samples per abstract-state action pair until a threshold is reached. They assume that samples can be obtained cheaply, e.g., through a simulator, whereas we focus on the exploration problem where data has to be collected online and is expensive. Another study operates in the abstraction selection setting [40]. While they do not assume that a Markov model exists in the given set of abstraction functions, they assume a given data set, with i.i.d. data. They give a bound on how accurate the Q-values based on the (implicitly) learned model will be rather than on the accuracy of the model itself. As we showed, the assumption that the data is i.i.d. is not trivial since it means the data cannot just be collected online. Another study's primary focus is on bandits but also gives results for MDPs with a coloring function [137]. We can view state aggregation as a special case of this coloring. They extend the results from UCRL2 [20] to the setting with a coloring function. They use the Azuma-Hoeffding inequality for the transition function, which holds for weakly dependent samples. However, they assume the samples are independent and do not show the martingale difference sequence property for the (actually dependent) samples.

Asymptotic Results. Another way to deal with dependence between samples is by looking at convergence in the limit [34, 50, 98]. One study gives an asymptotic result for convergence of Q-learning and TD(0) in MDPs with soft state aggregation [34]. In soft state aggregation, a state s belongs to a cluster x with some probability $P(x|s)$, which means a state s can belong to several clusters. Their result requires an ergodic MDP and a stationary policy that assigns a non-zero probability to every

action. Together these imply a limiting state distribution, and they use this to show convergence asymptotically. Another study gives multiple results focusing on approximate and exact abstractions in environments without MDP assumptions [98]. Several of these results are in the planning setting, similar to other planning results for approximate abstractions [16]. Most relevant for us is their Theorem 12, which for online RL shows convergence in the limit of the empirical transition function under weak conditions, e.g., when the abstract process is an MDP. Under this condition, however, the problem reduces to RL in an (abstract) MDP rather than RLAO. Follow-up work builds on some of these results and focuses on the combination of model-free RL and exact abstraction, also without MDP assumptions [50]. They define and operate in a Q-Value Uniform Decision Process, with a mapping from histories to (non-Markovian) states and a "state-uniformity condition". The state-uniformity means that if two histories map to the same state s, their optimal Q-values are also the same. They show that, under state-uniformity, Q-learning converges in the limit to the optimal solution. In contrast to our setting, they used an exact abstraction and left extending the results to approximate abstraction as an open question.

Planning and Abstraction. For planning in abstract MDPs, there are results for exact state abstractions [15] and approximate state abstractions [16]. The results for approximate state abstractions allow for quantifying an upper bound on performance for the optimal policy of an abstract MDP, e.g., as in Theorem 3.1 for approximate model similarity in Section 3.2.3. A study built on these results by giving a result for performing RL interacting with an explicitly constructed abstract MDP [80]; since the abstract MDP is still an MDP, this is different from RLAO.

MBRL Using I.I.D. Bounds. Concentration inequalities for i.i.d. samples, such as the result from Weissman *et al.* [133], are often directly applied to the empirical transition function [13, 20, 127, 130], without mentioning that these samples in a simple RL trajectory may not be independent as shown for instance by Strehl and Littman [14] in a non-communicating MDP. [9] Strehl and Littman [14] show that there dependence is not a problem because it is still possible to use a concentration inequality for independent samples, e.g., Hoeffding's inequality, as an upper bound, which implies that derived performance loss bounds are valid. However, their proof uses that transitions and rewards are identically distributed, which is not guaranteed in RLAO.

RL Using Martingale Bounds. Martingale concentration inequalities have been used regularly in online RL analysis [14, 20, 49, 51, 82, 104, 109, 137, 138]. Our novelty is in using it in RLAO, where we use it to show that we can learn an accurate model and provide performance guarantees in this setting. Several works that employ martingale concentration inequalities are not in the RLAO setting or

[9] An MDP is communicating if, for all $s_1, s_2 \in S$, a deterministic policy exists that eventually leads from s_1 to s_2 [74].

do not use them for the transition model, and instead apply them to other parts of the analysis such as bounding the difference between the actual and expected returns [14, 20, 39, 49, 51, 104]. Other works do use martingales for a transition model [82, 109, 137, 138]. However, these either (implicitly or explicitly) assume samples to be independent [137, 138] or identically distributed [82, 109], unlike our analysis. Since, as we detailed in Section 3.3, independent nor identically distributed samples cannot be guaranteed in RLAO, their analyses do not extend to this setting.

3.6. Discussion and Future Work

Some assumptions we made, i.e., that the reward function is deterministic and each state in an abstract state has the same reward function, can be relaxed. To accurately learn an abstract reward function, one should define a suitable martingale difference sequence, after which Lemma 3.3 can be used. We considered approximate model similarity abstraction and used the properties of this abstraction to establish an upper bound on the difference in value between the original MDP and an abstract MDP under any abstract policy. This bound was imperative for our results. We established this bound by proof of induction on the difference in value for a horizon n. This technique could be used to establish similar bounds and extend our results for other abstractions, e.g., approximate Q^* similarity abstractions [16], see Claim 3.1.

Our analysis showed how to extend the results of R-MAX [13] to RLAO. Extending results of other algorithms, e.g., MBIE [14] and UCRL2 [20], requires adapting to slightly different assumptions. For instance, R-MAX assumes ergodicity, while UCRL2 and MBIE assume the problem is communicating and non-communicating, respectively. Other algorithms sometimes use concentration inequalities other than Hoeffding's Inequality, e.g., the empirical Bernstein inequality [140, 141] or the Chernoff bound. To adapt these, we could, for instance, use Bernstein-type inequalities for martingales [142].

Theorem 3.4 shows that, despite problems with dependence, we can give finite-sample guarantees when combining approximate model similarity abstractions with MBRL. For good abstraction functions, i.e., when η_R and η_T are small and $|S| \gg |\bar{S}|$, this leads to near-optimal solutions while needing fewer samples, compared to learning without abstraction. Practically, for tabular methods, these results mostly mean that concentration inequalities for independent samples have to be replaced in RLAO, for example by concentration inequalities based on martingales, as we have shown here. In deep model-based RL, several recent empirical works have shown promising results by focusing on learning exact abstractions [45, 143]. An interesting direction is adapting these methods to learn approximate abstractions instead of exact abstractions. Since, compared to exact model similarity abstractions, approximate model similarity abstraction generally results in a smaller (abstract) state space; this could lead to faster learning.

Our results shed further light on the observation from Abel *et al.* [41] that RLAO is different from performing RL in an MDP constructed with abstraction. As our observations show, in RLAO the transition functions are not static, the samples are

not identically distributed, and cannot be guaranteed to be independent. This could mean that in situations where we want to learn an abstraction, the behavior is also not quite as expected. In such situations, similar approaches that we applied here may prove useful, as many situations in RL have already been shown to not be independent processes. While our results hold for approximate model similarity models, there could be even more compact representations for which our techniques could lead to similar results. One clear example would be abstractions that focus not on state abstraction, but rather on state-action abstraction, of which state abstraction is simply a special case.

People have been applying MDPs and RL to all kinds of problems, even though we know that the Markov property very rarely holds. Given that almost all theory of RL critically depends on this property, one could wonder why these things even work? Intuitively, we expect that the states in these successful applications are somehow "Markovian enough". In this work, we provide an understanding of this vague concept. Specifically, we show that an existing criterion of state representations (approximate model similarity) in fact is a formal notion of "Markovian enough" in MBRL. Thus, it provides critical insight into under what circumstances (and therefore in what applications) MBRL methods are expected to work.

3.7. CONCLUSION

We analyzed RLAO: online MBRL combined with state abstraction when the model of the MDP is unavailable. Via a counterexample, we showed that it cannot be guaranteed that samples obtained online in RLAO are independent. Many current guarantees from MBRL methods use concentration results that assume i.i.d. samples, e.g., Theorem 2.1 from Weissman *et al.* [133], the empirical Bernstein inequality [140, 141], or the Chernoff bound. Because they use these concentration inequalities, their guarantees do not hold in RLAO. In fact, none of the existing analyses of MBRL apply to RLAO. We showed that samples in RLAO are only weakly dependent and that concentration inequalities for (weakly) dependent variables, such as Lemma 3.3, are a viable alternative through which we can come to guarantees on the empirical model. We used this result to present the first sample efficient learning results for RLAO, thus showing it is possible to combine the benefits of abstraction and MBRL. These results showcase under what circumstances performance guarantees in MBRL can be transferred to settings with abstraction.

3.8. APPENDIX

3.8.1. WELL KNOWN RESULTS

We restate some well-known results that we use in the proofs in the other sections.

HOEFFDING'S INEQUALITY

Hoeffding's inequality can tell us the probability that the average of m random independent (but not necessarily identically distributed) samples deviates more than ϵ from its expectation.

Let $Z^{(1)}, Z^{(2)}, \cdots, Z^{(m)}$ be bounded independent random variables, and let $\bar{Z}$ and μ be defined as

$$\bar{Z} \triangleq \frac{Z^{(1)} + \cdots + Z^{(m)}}{m}, \tag{3.38}$$

$$\mu \triangleq E[\bar{Z}] = \frac{E[Z^{(1)} + \cdots + Z^{(m)}]}{m}. \tag{3.39}$$

Then Hoeffding's inequality states:

Lemma 3.4 (Hoeffding's inequality [53])**.** If $Z^{(1)}, Z^{(2)}, \cdots, Z^{(m)}$ are independent and $0 \leq Z^{(i)} \leq 1$ for $i = 1, \cdots, m$, then for $0 < \epsilon < 1 - \mu$ we have the following inequalities

$$\Pr(\bar{Z} - \mu \geq \epsilon) \leq e^{-2m\epsilon^2}, \tag{3.40}$$

$$\Pr(|\bar{Z} - \mu| \geq \epsilon) \leq 2e^{-2m\epsilon^2}, \tag{3.41}$$

$$\Pr(\sum_{i=1}^{m} (Z^{(i)} - \mu) \geq \epsilon) \leq e^{-2\frac{\epsilon^2}{m}}, \tag{3.42}$$

$$\Pr(|\sum_{i=1}^{m} (Z^{(i)} - \mu)| \geq \epsilon) \leq 2e^{-2\frac{\epsilon^2}{m}}. \tag{3.43}$$

UNION BOUND

Given that we have a set of events, the union bound allows us to upper bound the probability that at least one of the events happens, even when these events are not independent.

Lemma 3.5 (Union Bound [144])**.** For a countable set of events $A_1, A_2, A_3, \cdots$, we have

$$\Pr(\cup_i A_i) \leq \sum_i \Pr(A_i). \tag{3.44}$$

I.e., the probability that at least one of the events happens is, at most, the sum of the probabilities of the individual events.

VALUE BOUNDS FOR ABSTRACT AND TRUE MODELS

Here we give upper bounds on the difference in value between the real MDP and an abstract MDP under various policies. We will use these bounds in Appendix 3.8.5 to

adapt the results of R-MAX [13] to RLAO. These bounds and proofs are very similar to existing bounds [13, 14, 16, 80]. Here we repeat these for abstract models in the undiscounted finite horizon and in the discounted infinite horizon.

We define the finite horizon value function $\forall s \in S$:

$$V^{\pi,n}(s) = R(s,\pi(s)) + \sum_{s' \in S} T(s'|s,\pi(s)) V^{\pi,n-1}(s'), \tag{3.45}$$

$$V^{\pi,1}(s) = R(s,\pi(s)). \tag{3.46}$$

We use $V^{\bar{\pi},n}$ to denote the value in M under policy $\bar{\pi}$ and $\bar{V}^{\bar{\pi},n}$ to denote the value in $\bar{M}$ under policy $\bar{\pi}$.

Theorem 3.1. *Let $M = \langle S, A, T, R\rangle$ be an MDP and $\bar{M} = \langle \bar{S}, A, \bar{T}, \bar{R}\rangle$ an abstract MDP, for some defined abstract transitions and rewards. We assume that*

$$\begin{aligned} \forall \bar{s}, \bar{s}' \in \bar{S},\ \ s \in \bar{s},\ \ a \in A : |\bar{T}(\bar{s}'|\bar{s},a) - \Pr(\bar{s}'|s,a)| \le \eta_T \\ \text{and } |\bar{R}(\bar{s},a) - R(s,a)| \le \eta_R. \end{aligned} \tag{3.47}$$

Then, for a finite horizon problem with horizon h we have:

$$V^*(s) - V^{\bar{\pi}^*}(s) \le 2h\eta_R + (h+1)h\eta_T |\bar{S}| R_{max}. \tag{3.48}$$

And for a discounted infinite horizon problem with discount γ we have:

$$V^*(s) - V^{\bar{\pi}^*}(s) \le \frac{2\eta_R}{1-\gamma} + \frac{2\gamma\eta_T |\bar{S}| R_{max}}{(1-\gamma)^2}. \tag{3.49}$$

We will use the following two Lemmas to proof the Theorem.

Lemma 3.6. Under the assumption of (3.47) and for every abstract policy $\bar{\pi}$ and for every state $s \in \bar{s}$, we have: for a finite horizon problem with horizon h:

$$|V^{\bar{\pi},h}(s) - \bar{V}^{\bar{\pi},h}(s)| \le h\eta_R + \frac{(h-1)h}{2}\eta_T |\bar{S}| R_{\max}, \tag{3.50}$$

and for a discounted infinite horizon problem with discount γ:

$$|V^{\bar{\pi}}(s) - \bar{V}^{\bar{\pi}}(s)| \le \frac{\eta_R}{1-\gamma} + \frac{\gamma\eta_T |\bar{S}| R_{\max}}{(1-\gamma)^2}. \tag{3.51}$$

Proof. The proof follows the same steps for both the discounted infinite horizon and the undiscounted finite horizon. For completeness, we show them both here.

First, for the undiscounted finite horizon. By induction, we will show that for $n \ge 1$

$$\forall \bar{s} \in \bar{S}, s \in \bar{s} : \quad |V^{\bar{\pi},n}(s) - \bar{V}^{\bar{\pi},n}(s)| \le n\eta_R + \frac{(n-1)n}{2}\eta_T |\bar{S}| R_{\max}. \tag{3.52}$$

For $n = 1$, we have

$$|V^{\bar{\pi},1}(s) - \bar{V}^{\bar{\pi},1}(s)| = |R(s,\pi(\bar{s})) - \bar{R}(\bar{s},\bar{\pi}(\bar{s}))| \le \eta_R. \tag{3.53}$$

Now assume that the induction hypothesis, (3.52), holds for $n-1$, then

$$
\begin{aligned}
&|V^{\bar{\pi},n}(s) - \bar{V}^{\bar{\pi},n}(s)| = |R(s,\bar{\pi}(\bar{s})) - \bar{R}(\bar{s},\bar{\pi}(\bar{s})) \\
&+ \sum_{s'\in S} T(s'|s,\bar{\pi}(s)) V^{\bar{\pi},n-1}(s') - \sum_{\bar{s}'\in\bar{S}} \bar{T}(\bar{s}'|\bar{s},\bar{\pi}(\bar{s}))\bar{V}^{\bar{\pi},n-1}(\bar{s}')| && (3.54)\\
&\leq |R(s,\bar{\pi}(\bar{s})) - \bar{R}(\bar{s},\bar{\pi}(\bar{s}))| + |\sum_{\bar{s}'\in\bar{S}}\sum_{s'\in\bar{s}'} T(s'|s,\bar{\pi}(s)) V^{\bar{\pi},n-1}(s') - \sum_{\bar{s}'\in\bar{S}} \bar{T}(\bar{s}'|\bar{s},\bar{\pi}(\bar{s}))\bar{V}^{\bar{\pi},n-1}(\bar{s}')| && (3.55)\\
&\leq \eta_R + |\sum_{\bar{s}'\in\bar{S}}\sum_{s'\in\bar{s}'} T(s'|s,\bar{\pi}(s)) V^{\bar{\pi},n-1}(s') - \sum_{\bar{s}'\in\bar{S}} \bar{V}^{\bar{\pi},n-1}(\bar{s}')\sum_{s'\in\bar{s}'} T(s'|s,\bar{\pi}(\bar{s}))| \\
&+ |\sum_{\bar{s}'\in\bar{S}} \bar{V}^{\bar{\pi},n-1}(\bar{s}')\sum_{s'\in\bar{s}'} T(s'|s,\bar{\pi}(\bar{s})) - \sum_{\bar{s}'\in\bar{S}} \bar{T}(\bar{s}'|\bar{s},\bar{\pi}(\bar{s}))\bar{V}^{\bar{\pi},n-1}(\bar{s}')| && (3.56)\\
&\leq \eta_R + |\sum_{\bar{s}'\in\bar{S}}\sum_{s'\in\bar{s}'} T(s'|s,\bar{\pi}(s))\left[V^{\bar{\pi},n-1}(s') - \bar{V}^{\bar{\pi},n-1}(\bar{s}')\right]| \\
&+ |\sum_{\bar{s}'\in\bar{S}} \left[\bar{T}(\bar{s}'|\bar{s},\bar{\pi}(\bar{s})) - \sum_{s'\in\bar{s}'} T(s'|s,\bar{\pi}(\bar{s}))\right]\bar{V}^{\bar{\pi},n-1}(\bar{s}')| && (3.57)\\
&\leq \eta_R + (n-1)\eta_R + \frac{(n-1-1)(n-1)}{2}\eta_T|\bar{S}|R_{\max} + \eta_T|\bar{S}|(n-1)R_{\max} && (3.58)\\
&= n\eta_R + \frac{(n-2)(n-1)}{2}\eta_T|\bar{S}|R_{\max} + \eta_T|\bar{S}|(n-1)R_{\max} && (3.59)\\
&= n\eta_R + \frac{(n-1)n}{2}\eta_T|\bar{S}|R_{\max}. && (3.60)
\end{aligned}
$$

For the step from (3.55) to (3.56), we subtract and add $\sum_{\bar{s}'\in\bar{S}} \bar{V}^{\bar{\pi},n-1}(\bar{s}')\sum_{s'\in\bar{s}'} T(s'|s,\bar{\pi}(\bar{s}))$, and from (3.57) to (3.58), we use the inductive hypothesis and the fact that $(n-1)R_{\max}$ is an upper bound on $\bar{V}^{\bar{\pi},n-1}(\bar{s}')$ since the maximum reward per timestep is $R_{\max}$.

Now, for the discounted infinite horizon. By induction, we will show that for $n \geq 1$

$$
\forall \bar{s}\in\bar{S}, s\in\bar{s}: \quad |V^{\bar{\pi},n}(s) - \bar{V}^{\bar{\pi},n}(s)| \leq \eta_R\gamma^{n-1} + \sum_{i=0}^{n-2}\gamma^i(\eta_R + \frac{\gamma\eta_T|\bar{S}|R_{\max}}{1-\gamma}). \tag{3.61}
$$

For $n=1$, we have

$$
|V^{\bar{\pi},1}(s) - \bar{V}^{\bar{\pi},1}(s)| = |R(s,\bar{\pi}(\bar{s})) - \bar{R}(\bar{s},\bar{\pi}(\bar{s}))| \leq \eta_R. \tag{3.62}
$$

Now assume that the induction hypothesis, (3.61), holds for $n-1$, then

$$|V^{\bar{\pi},n}(s) - \bar{V}^{\bar{\pi},n}(s)| = |R(s,\bar{\pi}(\bar{s})) - \bar{R}(\bar{s},\bar{\pi}(\bar{s}))$$
$$+\gamma(\sum_{s'\in S} T(s'|s,\bar{\pi}(s))V^{\bar{\pi},n-1}(s') - \sum_{\bar{s}'\in\bar{S}} \bar{T}(\bar{s}'|\bar{s},\bar{\pi}(\bar{s}))\bar{V}^{\bar{\pi},n-1}(\bar{s}'))| \tag{3.63}$$
$$\leq |R(s,\bar{\pi}(\bar{s})) - \bar{R}(\bar{s},\bar{\pi}(\bar{s}))| + \gamma|\sum_{\bar{s}'\in\bar{S}}\sum_{s'\in\bar{s}'} T(s'|s,\bar{\pi}(s))V^{\bar{\pi},n-1}(s') - \sum_{\bar{s}'\in\bar{S}} \bar{T}(\bar{s}'|\bar{s},\bar{\pi}(\bar{s}))\bar{V}^{\bar{\pi},n-1}(\bar{s}')| \tag{3.64}$$
$$\leq \eta_R + \gamma|\sum_{\bar{s}'\in\bar{S}}\sum_{s'\in\bar{s}'} T(s'|s,\bar{\pi}(s))V^{\bar{\pi},n-1}(s') - \sum_{\bar{s}'\in\bar{S}} \bar{V}^{\bar{\pi},n-1}(\bar{s}')\sum_{s'\in\bar{s}'} T(s'|s,\bar{\pi}(\bar{s}))|$$
$$+\gamma|\sum_{\bar{s}'\in\bar{S}} \bar{V}^{\bar{\pi},n-1}(\bar{s}')\sum_{s'\in\bar{s}'} T(s'|s,\bar{\pi}(\bar{s})) - \sum_{\bar{s}'\in\bar{S}} \bar{T}(\bar{s}'|\bar{s},\bar{\pi}(\bar{s}))\bar{V}^{\bar{\pi},n-1}(\bar{s}')| \tag{3.65}$$
$$\leq \eta_R + \gamma|\sum_{\bar{s}'\in\bar{S}}\sum_{s'\in\bar{s}'} T(s'|s,\bar{\pi}(s))\left[V^{\bar{\pi},n-1}(s') - \bar{V}^{\bar{\pi},n-1}(\bar{s}')\right]|$$
$$+\gamma|\sum_{\bar{s}'\in\bar{S}}\left[\bar{T}(\bar{s}'|\bar{s},\bar{\pi}(\bar{s})) - \sum_{s'\in\bar{s}'} T(s'|s,\bar{\pi}(\bar{s}))\right]\bar{V}^{\bar{\pi},n-1}(\bar{s}')| \tag{3.66}$$
$$\leq \eta_R + \gamma(\eta_R\gamma^{n-2} + \sum_{i=0}^{n-3}\gamma^i(\eta_R + \frac{\gamma\eta_T|\bar{S}|R_{\max}}{1-\gamma})) + \gamma\eta_T|\bar{S}|\frac{R_{\max}}{1-\gamma} \tag{3.67}$$
$$= \eta_R + \eta_R\gamma^{n-1} + \sum_{i=1}^{n-2}\gamma^i(\eta_R + \frac{\gamma\eta_T|\bar{S}|R_{\max}}{1-\gamma}) + \gamma\eta_T|\bar{S}|\frac{R_{\max}}{1-\gamma} \tag{3.68}$$
$$= \eta_R\gamma^{n-1} + \sum_{i=0}^{n-2}\gamma^i(\eta_R + \frac{\gamma\eta_T|\bar{S}|R_{\max}}{1-\gamma}). \tag{3.69}$$

For the step from (3.64) to (3.65), we subtract and add $\sum_{\bar{s}'\in\bar{S}} \bar{V}^{\bar{\pi},n-1}(\bar{s}')\sum_{s'\in\bar{s}'} T(s'|s,\bar{\pi}(\bar{s}))$, and from (3.66) to (3.67), we use the inductive hypothesis and the fact that $\frac{R_{\max}}{1-\gamma}$ is an upper bound on $\bar{V}^{\bar{\pi},n-1}(\bar{s}')$.

Finally, taking the limit for $n\to\infty$, we get:

$$|V^{\bar{\pi}}(s) - \bar{V}^{\bar{\pi}}(s)| \leq \eta_R \times 0 + \frac{1}{1-\gamma}(\eta_R + \frac{\gamma\eta_T|\bar{S}|R_{\max}}{1-\gamma})$$
$$= \frac{\eta_R}{1-\gamma} + \frac{\gamma\eta_T|\bar{S}|R_{\max}}{(1-\gamma)^2}.$$

□

Lemma 3.7. Under the assumption of (3.47) and for every state $s\in\bar{s}$, we have: for a finite horizon problem with horizon h:

$$|V^{*,h}(s) - \bar{V}^{*,h}(s)| \leq h\eta_R + \frac{(h-1)h}{2}\eta_T|\bar{S}|R_{\max}, \tag{3.70}$$

and for a discounted infinite horizon problem with discount γ:

$$|V^*(s) - \bar{V}^*(s)| \leq \frac{\eta_R}{1-\gamma} + \frac{\gamma\eta_T|\bar{S}|R_{\max}}{(1-\gamma)^2}. \tag{3.71}$$

Proof. First, we define

$$\forall \bar{s} \in \bar{S}, s \in S: \quad V^{*,n}(s) = \max_{a \in A} \big[R(s,a) + \gamma \sum_{s' \in S} T(s'|s,a) V^{*,n-1}(s') \big], \tag{3.72}$$

$$\bar{V}^{*,n}(\bar{s}) = \max_{a \in A} \big[\bar{R}(\bar{s},a) + \gamma \sum_{\bar{s}' \in \bar{S}} \bar{T}(\bar{s}'|s,a) \bar{V}^{*,n-1}(\bar{s}') \big]. \tag{3.73}$$

For the undiscounted case $\gamma = 1$, so we can drop γ from the notation.

The proof follows the same steps as the proof of Lemma 3.6. We start again with the undiscounted finite horizon.

By induction, we will show that for $n \geq 1$

$$\forall \bar{s} \in \bar{S}, s \in \bar{s}: \quad |V^{*,n}(s) - \bar{V}^{*,n}(\bar{s})| \leq n\eta_R + \frac{(n-1)n}{2} \eta_T |\bar{S}| R_{\max}. \tag{3.74}$$

Making use of the fact that $|\max f - \max g| \leq \max |f - g|$, we have for $n = 1$

$$|V^{*,1}(s) - \bar{V}^{*,1}(s)| = |\max_{a \in A} R(s,a) - \max_{a \in A} \bar{R}(\bar{s},a)| \leq \max_{a \in A} |R(s,a) - \bar{R}(\bar{s},a)| \leq \eta_R. \tag{3.75}$$

Now assume that the induction hypothesis, (3.74), holds for $n-1$, then

$$|V^{*,n}(s) - \bar{V}^{*,n}(\bar{s})| = \max_{a \in A} |R(s,a) - \bar{R}(\bar{s},a)$$

$$+ \sum_{s' \in S} T(s'|s,a) V^{*,n-1}(s') - \sum_{\bar{s}' \in \bar{S}} \bar{T}(\bar{s}'|\bar{s},a) \bar{V}^{*,n-1}(\bar{s}')| \tag{3.76}$$

$$\leq \max_{a \in A} |R(s,a) - \bar{R}(\bar{s},a)| + \max_{a \in A} |\sum_{\bar{s}' \in \bar{S}} \sum_{s' \in \bar{s}'} T(s'|s,a) V^{*,n-1}(s') - \sum_{\bar{s}' \in \bar{S}} \bar{T}(\bar{s}'|\bar{s},a) \bar{V}^{*,n-1}(\bar{s}')| \tag{3.77}$$

$$\leq \eta_R + \max_{a \in A} |\sum_{\bar{s}' \in \bar{S}} \sum_{s' \in \bar{s}'} T(s'|s,a) V^{*,n-1}(s') - \sum_{\bar{s}' \in \bar{S}} \bar{V}^{*,n-1}(\bar{s}') \sum_{s' \in \bar{s}'} T(s'|s,a)|$$

$$+ \max_{a \in A} |\sum_{\bar{s}' \in \bar{S}} \bar{V}^{*,n-1}(\bar{s}') \sum_{s' \in \bar{s}'} T(s'|s,a) - \sum_{\bar{s}' \in \bar{S}} \bar{T}(\bar{s}'|\bar{s},a) \bar{V}^{*,n-1}(\bar{s}')| \tag{3.78}$$

$$\leq \eta_R + \max_{a \in A} |\sum_{\bar{s}' \in \bar{S}} \sum_{s' \in \bar{s}'} T(s'|s,a) \big[V^{*,n-1}(s') - \bar{V}^{*,n-1}(\bar{s}') \big]|$$

$$+ \max_{a \in A} |\sum_{\bar{s}' \in \bar{S}} \big[\bar{T}(\bar{s}'|\bar{s},a) - \sum_{s' \in \bar{s}'} T(s'|s,a) \big] \bar{V}^{*,n-1}(\bar{s}')| \tag{3.79}$$

$$\leq \eta_R + (n-1)\eta_R + \frac{(n-1-1)(n-1)}{2} \eta_T |\bar{S}| R_{\max} + \eta_T (n-1) |\bar{S}| R_{\max} \tag{3.80}$$

$$= n\eta_R + \frac{(n-1)n}{2} \eta_T |\bar{S}| R_{\max}. \tag{3.81}$$

For the step from (3.77) to (3.78), we subtract and add $\sum_{\bar{s}' \in \bar{S}} \bar{V}^{*,n-1}(\bar{s}') \sum_{s' \in \bar{s}'} T(s'|s,a)$, and from (3.79) to (3.80), we use the inductive hypothesis and again the fact that $(n-1)R_{\max}$ is an upper bound on $\bar{V}^{*,n-1}(\bar{s}')$ since the maximum reward per timestep is $R_{\max}$.

Now, for the discounted infinite horizon. By induction, we will show that for $n \geq 1$

$$\forall \bar{s} \in \bar{S}, s \in \bar{s}: \quad |V^{*,n}(s) - \bar{V}^{*,n}(s)| \leq \eta_R \gamma^{n-1} + \sum_{i=0}^{n-2} \gamma^i (\eta_R + \frac{\gamma \eta_T |\bar{S}| R_{\max}}{1-\gamma}). \tag{3.82}$$

For $n=1$, we have

$$|V^{*,1}(s)-\bar{V}^{*,1}(s)| = |\max_{a\in A} R(s,a) - \max_{a\in A}\bar{R}(\bar{s},a)| \leq \max_{a\in A}|R(s,a)-\bar{R}(\bar{s},a)| \leq \eta_R. \tag{3.83}$$

Now assume that the induction hypothesis, (3.82), holds for $n-1$, then

$$\begin{aligned}
&|V^{*,n}(s)-\bar{V}^{*,n}(\bar{s})| = \max_{a\in A}|R(s,a)-\bar{R}(\bar{s},a) \\
&+\gamma\sum_{s'\in S}T(s'|s,a)V^{*,n-1}(s') - \gamma\sum_{\bar{s}'\in\bar{S}}\bar{T}(\bar{s}'|\bar{s},a)\bar{V}^{*,n-1}(\bar{s}')| && (3.84)\\
&\leq \max_{a\in A}|R(s,a)-\bar{R}(\bar{s},a)| + \max_{a\in A}\gamma|\sum_{\bar{s}'\in\bar{S}}\sum_{s'\in\bar{s}'}T(s'|s,a)V^{*,n-1}(s') - \sum_{\bar{s}'\in\bar{S}}\bar{T}(\bar{s}'|\bar{s},a)\bar{V}^{*,n-1}(\bar{s}')| && (3.85)\\
&\leq \eta_R + \max_{a\in A}\gamma|\sum_{\bar{s}'\in\bar{S}}\sum_{s'\in\bar{s}'}T(s'|s,a)V^{*,n-1}(s') - \sum_{\bar{s}'\in\bar{S}}\bar{V}^{*,n-1}(\bar{s}')\sum_{s'\in\bar{s}'}T(s'|s,a)| \\
&+\max_{a\in A}\gamma|\sum_{\bar{s}'\in\bar{S}}\bar{V}^{*,n-1}(\bar{s}')\sum_{s'\in\bar{s}'}T(s'|s,a) - \sum_{\bar{s}'\in\bar{S}}\bar{T}(\bar{s}'|\bar{s},a)\bar{V}^{*,n-1}(\bar{s}')| && (3.86)\\
&\leq \eta_R + \max_{a\in A}\gamma|\sum_{\bar{s}'\in\bar{S}}\sum_{s'\in\bar{s}'}T(s'|s,a)\big[V^{*,n-1}(s') - \bar{V}^{*,n-1}(\bar{s}')\big]| + \\
&\max_{a\in A}\gamma|\sum_{\bar{s}'\in\bar{S}}\big[\bar{T}(\bar{s}'|\bar{s},a) - \sum_{s'\in\bar{s}'}T(s'|s,a)\big]\bar{V}^{*,n-1}(\bar{s}')| && (3.87)\\
&\leq \eta_R + \gamma(\eta_R\gamma^{n-2} + \sum_{i=0}^{n-3}\gamma^i(\eta_R + \frac{\gamma\eta_T|\bar{S}|R_{\max}}{1-\gamma})) + \gamma\eta_T|\bar{S}|\frac{R_{\max}}{1-\gamma} && (3.88)\\
&= \eta_R + \eta_R\gamma^{n-1} + \sum_{i=1}^{n-2}\gamma^i(\eta_R + \frac{\gamma\eta_T|\bar{S}|R_{\max}}{1-\gamma}) + \gamma\eta_T|\bar{S}|\frac{R_{\max}}{1-\gamma} && (3.89)\\
&= \eta_R\gamma^{n-1} + \sum_{i=0}^{n-2}\gamma^i(\eta_R + \frac{\gamma\eta_T|\bar{S}|R_{\max}}{1-\gamma}). && (3.90)
\end{aligned}$$

For the step from (3.85) to (3.86), we subtract and add $\sum_{\bar{s}'\in\bar{S}}\bar{V}^{\bar{\pi},n-1}(\bar{s}')\sum_{s'\in\bar{s}'}T(s'|s,\bar{\pi}(\bar{s}))$, and from (3.87) to (3.88), we use the inductive hypothesis and the fact that $\frac{R_{\max}}{1-\gamma}$ is an upper bound on $\bar{V}^{*,n-1}(\bar{s}')$.

Finally, taking the limit for $n\to\infty$, we get:

$$\begin{aligned}
|V^*(s)-\bar{V}^*(s)| &\leq \eta_R\times 0 + \frac{1}{1-\gamma}(\eta_R + \frac{\gamma\eta_T|\bar{S}|R_{\max}}{1-\gamma}) \\
&= \frac{\eta_R}{1-\gamma} + \frac{\gamma\eta_T|\bar{S}|R_{\max}}{(1-\gamma)^2}.
\end{aligned}$$

□

Proof of Theorem 3.1. We can now proof Theorem 3.1, by using the triangle inequality

and the results of Lemmas 3.6 and 3.7. For the undiscounted finite horizon:

$$\begin{aligned}|V^{*,h}(s) - V^{\bar{\pi}^*,h}(s)| &\leq |V^{*,h}(s) - \bar{V}^{*,h}(s)| + |\bar{V}^{*,h}(s) - V^{\bar{\pi}^*,h}(s)| \\ &= |V^{*,h}(s) - \bar{V}^{*,h}(s)| + |\bar{V}^{\bar{\pi}^*,h}(s) - V^{\bar{\pi}^*,h}(s)| \\ &\leq h\eta_R + \frac{(h-1)h}{2}\eta_T|\bar{S}|R_{\max} + h\eta_R + \frac{(h-1)h}{2}\eta_T|\bar{S}|R_{\max} \\ &= 2h\eta_R + (h-1)h\eta_T|\bar{S}|R_{\max}.\end{aligned}$$

For the discounted infinite horizon:

$$\begin{aligned}|V^*(s) - V^{\bar{\pi}^*}(s)| &\leq |V^*(s) - \bar{V}^*(s)| + |\bar{V}^*(s) - V^{\bar{\pi}^*}(s)| \\ &= |V^*(s) - \bar{V}^*(s)| + |\bar{V}^{\bar{\pi}^*}(s) - V^{\bar{\pi}^*}(s)| \\ &\leq \frac{\eta_R}{1-\gamma} + \frac{\gamma\eta_T|\bar{S}|R_{\max}}{(1-\gamma)^2} + \frac{\eta_R}{1-\gamma} + \frac{\gamma\eta_T|\bar{S}|R_{\max}}{(1-\gamma)^2} \\ &= \frac{2\eta_R}{1-\gamma} + \frac{2\gamma\eta_T|\bar{S}|R_{\max}}{(1-\gamma)^2}.\end{aligned}$$

□

VALUE DIFFERENCE FOR SIMILAR MDPS

Finally, we give a simulation lemma for two MDPs on the same state-action space.

Lemma 3.8. Let M and M' be two MDPs on the same state-action space, with

$$\forall s, a \in S \times A: \quad |R_M(s,a) - R_{M'}(s,a)| \leq \eta_R, \tag{3.91}$$

$$\forall s, a, s' \in S \times A \times S: \quad |T_M(s'|s,a) - T_{M'}(s'|s,a)| \leq \eta_T. \tag{3.92}$$

Then, for every policy π and for every state $s \in S$ we have:

$$|V_M^{\pi,n}(s) - V_{M'}^{\pi,n}(s)| \leq n\eta_R + \frac{(n-1)n}{2}\eta_T|S|R_{\max}. \tag{3.93}$$

Proof. By induction, we will show that for $n \geq 1$

$$\forall s \in S: \quad |V_M^{\pi,n}(s) - V_{M'}^{\pi,n}(s)| \leq n\eta_R + \frac{(n-1)n}{2}\eta_T|S|R_{\max}. \tag{3.94}$$

For $n = 1$, we have

$$|V_M^{\pi,1}(s) - V_{M'}^{\pi,1}(s)| = |R_M(s,\pi(s)) - R_{M'}(s,\pi(s))| \leq \eta_R. \tag{3.95}$$

Now assume that the induction hypothesis, (3.94), holds for $n-1$, then

$$|V_M^{\pi,n}(s) - V_{M'}^{\pi,n}(s)| = |R_M(s,\pi(\bar{s})) - R_{M'}(s,\pi(s))$$
$$+ \sum_{s' \in S} T_M(s'|s,\pi(s)) V_M^{\pi,n-1}(s') - \sum_{s' \in S} T_{M'}(s'|s,\pi(s))) V_{M'}^{\pi,n-1}(s')| \tag{3.96}$$
$$\leq |R_M(s,\pi(s)) - R_{M'}(s,\pi(s))| + |\sum_{s' \in S} T_M(s'|s,\pi(s)) V_M^{\pi,n-1}(s') - \sum_{s' \in S} T_{M'}(s'|s,\pi(s)) V_{M'}^{\pi,n-1}(s')| \tag{3.97}$$
$$\leq \eta_R + |\sum_{s' \in S} T_M(s'|s,\pi(s)) V_M^{\pi,n-1}(s') - \sum_{s' \in S} T_M(s'|s,\pi(s)) V_{M'}^{\pi,n-1}(s')|$$
$$+ |\sum_{s' \in S} T_M(s'|s,\pi(s)) V_{M'}^{\pi,n-1}(s') - \sum_{s' \in S} T_{M'}(s'|s,\pi(s)) V_{M'}^{\pi,n-1}(s')| \tag{3.98}$$
$$\leq \eta_R + |\sum_{s' \in S} T_M(s'|s,\pi(s)) \left[V_M^{\pi,n-1}(s') - V_{M'}^{\pi,n-1}(s') \right]|$$
$$+ |\sum_{s' \in S} \left[T_M(s'|s,\pi(s)) - T_{M'}(s'|s,\pi(s)) \right] V_{M'}^{\pi,n-1}(s')| \tag{3.99}$$
$$\leq \eta_R + (n-1)\eta_R + \frac{(n-1-1)(n-1)}{2} \eta_T |S| R_{\max} + \eta_T (n-1)|S| R_{\max} \tag{3.100}$$
$$= n\eta_R + \frac{(n-2)(n-1)}{2} \eta_T |S| R_{\max} + \eta_T (n-1)|S| R_{\max} \tag{3.101}$$
$$= n\eta_R + \frac{(n-1)n}{2} \eta_T |S| R_{\max}. \tag{3.102}$$

For the step from (3.97) to (3.98), we add and subtract $\sum_{\bar{s}' \in S} T_M(s'|s,\pi(s)) V_{M'}^{\pi,n-1}(s')$, and from (3.99) to (3.100), we use the inductive hypothesis and the fact that $(n-1)R_{\max}$ is an upper bound on $V_{M'}^{\pi,n-1}(s')$ since the maximum reward per timestep is $R_{\max}$. □

This shows that the values under any policy are similar for similar MDPs.

3.8.2. L1 INEQUALITY FOR INDEPENDENT BUT NOT IDENTICALLY DISTRIBUTED VARIABLES

We show that we can adapt the proof of Weissman *et al.* [133] for independent, but not identically distributed, samples to obtain the following result:

Lemma 3.2. *Let $X_{\bar{s},a} = s_1, \cdots, s_m$ be a sequence of states $s \in \bar{s}$ and let $\bar{\boldsymbol{Y}}_{\bar{s},a} = \bar{Y}^{(1)}, \bar{Y}^{(2)}, \cdots, \bar{Y}^{(m)}$ be independent random variables distributed according to $\Pr(\cdot|s_1,a), \cdots, \Pr(\cdot|s_m,a)$. Then, $\forall \epsilon > 0$,*

$$\Pr(||\bar{T}_Y(\cdot|\bar{s},a) - \bar{T}_{\omega_X}(\cdot|\bar{s},a)||_1 \geq \epsilon) \leq (2^{|\bar{S}|} - 2) e^{-\frac{1}{2} m \epsilon^2}. \tag{3.103}$$

Proof of Lemma 3.2. The proof mostly follows the steps by Weissman *et al.* [133].

To shorten the notation, we define $P_Y \triangleq \bar{T}_Y(\cdot|\bar{s},a)$ and $P_{\omega_X} \triangleq \bar{T}_{\omega_X}(\cdot|\bar{s},a)$.

We will make use of the following result (Proposition 4.2 by Levin and Peres [145]), that for any distribution Q on $\bar{S}$

$$||Q - P_{\omega_X}||_1 = 2 \max_{\bar{\mathscr{S}} \subseteq \bar{S}} (Q(\bar{\mathscr{S}}) - P_{\omega_X}(\bar{\mathscr{S}})),$$

where $\bar{\mathscr{S}}$ is a subset of $\bar{S}$, and $P_{\omega_X}(\bar{\mathscr{S}}) = \sum_{\bar{s}' \in \bar{\mathscr{S}}} P_{\omega_X}(\bar{s}')$. Thus, we have that

$$||P_Y - P_{\omega_X}||_1 = 2 \max_{\bar{\mathscr{S}} \subseteq \bar{S}} (P_Y(\bar{\mathscr{S}}) - P_{\omega_X}(\bar{\mathscr{S}})). \tag{3.104}$$

Using this, we can write

$$\begin{aligned}
\Pr(||P_Y - P_{\omega_X}||_1 \geq \epsilon) &= \Pr\left[2 \max_{\bar{\mathscr{S}} \subseteq \bar{S}} \left[P_Y(\bar{\mathscr{S}}) - P_{\omega_X}(\bar{\mathscr{S}})\right] \geq \epsilon\right] && (3.105)\\
&= \Pr\left[\max_{\bar{\mathscr{S}} \subseteq \bar{S}} \left[P_Y(\bar{\mathscr{S}}) - P_{\omega_X}(\bar{\mathscr{S}})\right] \geq \frac{\epsilon}{2}\right] && (3.106)\\
&= \Pr\left[\cup_{\bar{\mathscr{S}} \subseteq \bar{S}} \left[P_Y(\bar{\mathscr{S}}) - P_{\omega_X}(\bar{\mathscr{S}}) \geq \frac{\epsilon}{2}\right]\right] && (3.107)\\
&\leq \sum_{\bar{\mathscr{S}} \subseteq \bar{S}} \Pr\left[P_Y(\bar{\mathscr{S}}) - P_{\omega_X}(\bar{\mathscr{S}}) \geq \frac{\epsilon}{2}\right], && (3.108)
\end{aligned}$$

where the last step follows from the union bound (Lemma 3.5).

Assuming $\epsilon > 0$, we have that $\Pr(P_Y(\bar{\mathscr{S}}) - P_{\omega_X}(\bar{\mathscr{S}}) \geq \frac{\epsilon}{2}) = 0$ when $\bar{\mathscr{S}} = \bar{S}$ or $\bar{\mathscr{S}} = \emptyset$. For every other subset $\bar{\mathscr{S}}$, we can define a random binary variable that is 1 when $Y^{(i)} \in \bar{\mathscr{S}}$ and 0 otherwise. Here $P_{\omega_X}(\bar{\mathscr{S}})$ acts as μ (3.39) from Lemma 3.4 and $P_Y(\bar{\mathscr{S}})$ as $\bar{Z}$ (3.38). Thus, by applying Lemma 3.4 to this random variable, we have

$$\Pr(P_Y(\bar{\mathscr{S}}) - P_{\omega_X}(\bar{\mathscr{S}}) \geq \frac{\epsilon}{2}) \leq e^{-2m\frac{\epsilon}{2}^2} = e^{-\frac{1}{2}m\epsilon^2}. \tag{3.109}$$

Then it follows that

$$\begin{aligned}
\Pr(||P_Y - P_{\omega_X}||_1 \geq \epsilon) &\leq \sum_{\bar{\mathscr{S}} \subseteq \bar{S}} \Pr(P_Y(\bar{\mathscr{S}}) - P_{\omega_X}(\bar{\mathscr{S}}) \geq \frac{\epsilon}{2}) && (3.110)\\
&\leq \sum_{\bar{\mathscr{S}} \subset \bar{S} : \bar{\mathscr{S}} \neq \bar{S}, \emptyset} \Pr(P_Y(\bar{\mathscr{S}}) - P_{\omega_X}(\bar{\mathscr{S}}) \geq \frac{\epsilon}{2}) && (3.111)\\
&\leq (2^{|\bar{S}|} - 2) e^{-\frac{1}{2}m\epsilon^2}, && (3.112)
\end{aligned}$$

where $\bar{S} \subset \bar{\mathscr{S}} : \bar{S} \neq \bar{\mathscr{S}}, \emptyset$ denotes that the empty set $\emptyset$ and the complete set $\bar{\mathscr{S}}$ are excluded. □

3.8.3. PROOF OF MAIN RESULT

Here we show how we can use a concentration inequality for martingales to learn an accurate transition model in RLAO. Specifically, the following result shows that, with a high probability, the empirical abstract transition function $\bar{T}_Y$ will be close to the abstract transition function $\bar{T}_{\omega_X}$. In the proof, which follows the general approach of Ortner, Gajane, and Auer [138], we define a suitable martingale difference sequence for the abstract transition function and use this to obtain the following result for learning a transition function in RLAO:

Theorem 3.2 (Abstract L1 inequality)**.** *If an agent has access to a state abstraction function ϕ and uses this to collect data for any abstract state-action pair $(\bar{s}, a)$ by*

acting in an MDP M according to a policy $\bar{\pi}$, we have that the following holds with a probability of at least $1-\delta$ for a fixed value of $N(\bar{s},a)$:

$$||\bar{T}_Y(\cdot|\bar{s},a) - \bar{T}_{\omega_X}(\cdot|\bar{s},a)||_1 \leq \epsilon, \tag{3.113}$$

where $\delta = 2^{|\bar{S}|} e^{-\frac{1}{8}N(\bar{s},a)\epsilon^2}$.

Proof of Theorem 3.2. We first define an abstract transition function based on $X_{\bar{s},a}$ as

$$\forall(\bar{s},a),\bar{s}': \quad \bar{T}_{\omega_X}(\bar{s}'|\bar{s},a) \triangleq \frac{1}{N(\bar{s},a)} \sum_{i=1}^{N(\bar{s},a)} T(\bar{s}'|X_{\bar{s},a}^{(i)},a), \tag{3.114}$$

where $T(\bar{s}'|X_{\bar{s},a}^{(i)},a) \triangleq \sum_{s'\in\bar{s}'} T(s'|X_{\bar{s},a}^{(i)},a)$. We write $\bar{T}_{\omega_X}$ because this definition is equivalent to using a weighting function as in (3.19):

$$\begin{aligned}
\forall(\bar{s},a),\bar{s}': \quad \bar{T}_{\omega_X}(\bar{s}'|\bar{s},a) &\triangleq \sum_{s\in\bar{s}} \omega_X(s,a) \sum_{s'\in\bar{s}'} T(s'|s,a) && \text{(Eq. 3.19)} && (3.115)\\
&= \sum_{s\in\bar{s}} \frac{1}{N(\bar{s},a)} \sum_{i=1}^{N(\bar{s},a)} \mathbb{1}\{X_{\bar{s},a}^{(i)} = s\} \sum_{s'\in\bar{s}'} T(s'|s,a) && && (3.116)\\
&= \frac{1}{N(\bar{s},a)} \sum_{i=1}^{N(\bar{s},a)} \sum_{s\in\bar{s}} \mathbb{1}\{X_{\bar{s},a}^{(i)} = s\} \sum_{s'\in\bar{s}'} T(s'|s,a) && && (3.117)\\
&= \frac{1}{N(\bar{s},a)} \sum_{i=1}^{N(\bar{s},a)} \sum_{s'\in\bar{s}'} T(s'|X_{\bar{s},a}^{(i)},a) && && (3.118)\\
&= \frac{1}{N(\bar{s},a)} \sum_{i=1}^{N(\bar{s},a)} T(\bar{s}'|X_{\bar{s},a}^{(i)},a). && \text{(Eq. 3.114)} && (3.119)
\end{aligned}$$

Now we use $\boldsymbol{z}$ to denote a vector of size $|\bar{S}|$ with entries ± 1, and we write $z(\bar{s})$ for the entry in $\boldsymbol{z}$ with index $\bar{s}$. Then we have

$$\begin{aligned}
&||\bar{T}_Y(\cdot|\bar{s},a) - \bar{T}_{\omega_X}(\cdot|\bar{s},a)||_1 = \sum_{\bar{s}'} |\bar{T}_Y(\bar{s}'|\bar{s},a) - \bar{T}_{\omega_X}(\bar{s}'|\bar{s},a)| && (3.120)\\
&= \max_{z\in\{-1,1\}^{\bar{S}}} \sum_{\bar{s}'} \Big(\bar{T}_Y(\bar{s}'|\bar{s},a) - \bar{T}_{\omega_X}(\bar{s}'|\bar{s},a)\Big) z(\bar{s}') && (3.121)\\
&= \max_{z\in\{-1,1\}^{\bar{S}}} \sum_{\bar{s}'} \Big(\frac{1}{N(\bar{s},a)} \sum_{i=1}^{N(\bar{s},a)} \mathbb{1}\{\bar{Y}_{\bar{s},a}^{(i)} = \bar{s}'\} - \frac{1}{N(\bar{s},a)} \sum_{i=1}^{N(\bar{s},a)} T(\bar{s}'|X_{\bar{s},a}^{(i)},a)\Big) z(\bar{s}') && (3.122)\\
&= \max_{z\in\{-1,1\}^{\bar{S}}} \sum_{\bar{s}'} \frac{1}{N(\bar{s},a)} \sum_{i=1}^{N(\bar{s},a)} \mathbb{1}\{\bar{Y}_{\bar{s},a}^{(i)} = \bar{s}'\} z(\bar{s}') - \sum_{\bar{s}'} \frac{1}{N(\bar{s},a)} \sum_{i=1}^{N(\bar{s},a)} T(\bar{s}'|X_{\bar{s},a}^{(i)},a) z(\bar{s}') && (3.123)\\
&= \max_{z\in\{-1,1\}^{\bar{S}}} \frac{1}{N(\bar{s},a)} \sum_{i=1}^{N(\bar{s},a)} z(\bar{Y}_{\bar{s},a}^{(i)}) - \frac{1}{N(\bar{s},a)} \sum_{i=1}^{N(\bar{s},a)} \sum_{\bar{s}'} T(\bar{s}'|X_{\bar{s},a}^{(i)},a) z(\bar{s}') && (3.124)\\
&= \max_{z\in\{-1,1\}^{\bar{S}}} \frac{1}{N(\bar{s},a)} \sum_{i=1}^{N(\bar{s},a)} \Big(z(\bar{Y}_{\bar{s},a}^{(i)}) - \sum_{\bar{s}'} T(\bar{s}'|X_{\bar{s},a}^{(i)},a) z(\bar{s}')\Big) && (3.125)\\
&= \max_{z\in\{-1,1\}^{\bar{S}}} \frac{1}{N(\bar{s},a)} \sum_{i=1}^{N(\bar{s},a)} Z_{\tau_i}(\boldsymbol{z}, X_{\bar{s},a}^{(i)}, a, \bar{Y}_{\bar{s},a}^{(i)}), && (3.126)
\end{aligned}$$

where we set

$$Z_{\tau_i}(\boldsymbol{z}, X^{(i)}_{\bar{s},a}, a_{\tau_i}, \bar{Y}^{(i)}_{\bar{s},a}) \triangleq z(\bar{Y}^{(i)}_{\bar{s},a}) - \sum_{\bar{s}'} T(\bar{s}'|X^{(i)}_{\bar{s},a}, a_{\tau_i}) z(\bar{s}').$$

To show that $\sum_i^{N(\bar{s},a)} Z_{\tau_i}(\boldsymbol{z}, X^{(i)}_{\bar{s},a}, a_{\tau_i}, \bar{Y}^{(i)}_{\bar{s},a})$ is a martingale difference sequence, we should follow Definition 3 and show that $\forall i : E[Z_{\tau_i}(\boldsymbol{z}, X^{(i)}_{\bar{s},a}, a_{\tau_i}, \bar{Y}^{(i)}_{\bar{s},a})|Z_{\tau_1}, Z_{\tau_2}, \cdots, Z_{\tau_{i-1}}] = 0$ and $|Z_i| < \infty$. For the second part, we have that $\forall i : |Z_{\tau_i}(\boldsymbol{z}, X^{(i)}_{\bar{s},a}, a_{\tau_i}, \bar{Y}^{(i)}_{\bar{s},a})| \leq 2$, since $|z(\bar{Y}^{(i)}_{\bar{s},a})| \leq 1$ and $|\sum_{\bar{s}'} T(\bar{s}'|X^{(i)}_{\bar{s},a}, a_{\tau_i}) z(\bar{s}')| \leq 1$. For the first part, we use the following Lemma, the proof of which follows after the current proof.

Lemma 3.9. Let π be a policy, and suppose the sequence $s_1, a_1 \cdots, s_{t-1}, a_{t-1}, s_t$ is to be generated by π. If $1 \leq \tau_1 < \tau_2 < \cdots < \tau_{i-1} < \tau_i \leq t$, then

$$E[Z_{\tau_i}(\boldsymbol{z}, X^{(i)}_{\bar{s},a}, a_{\tau_i}, \bar{Y}^{(i)}_{\bar{s},a})|Z_{\tau_1}, Z_{\tau_2}, \cdots, Z_{\tau_{i-1}}] = 0.$$

Lemma 3.9 shows that $\sum_i^{N(\bar{s},a)} Z_{\tau_i}(\boldsymbol{z}, X^{(i)}_{\bar{s},a}, a_{\tau_i}, \bar{Y}^{(i)}_{\bar{s},a})$ is a martingale difference sequence with $\forall i : |Z_{\tau_i}(\boldsymbol{z}, X^{(i)}_{\bar{s},a}, a_{\tau_i}, \bar{Y}^{(i)}_{\bar{s},a})| \leq 2$ for any fixed $\boldsymbol{z}$ and fixed $N(\bar{s},a) = n$ so that by the Azuma-Hoeffding inequality (Lemma 3.3):

$$\Pr(\sum_{i=1}^{N(\bar{s},a)} Z_{\tau_i} > \epsilon) \leq e^{-\frac{\epsilon^2}{8N(\bar{s},a)}}. \tag{3.127}$$

Similarly, $\sum_i^{N(\bar{s},a)} \frac{1}{N(\bar{s},a)} Z_{\tau_i}(\boldsymbol{z}, X^{(i)}_{\bar{s},a}, a_{\tau_i}, \bar{Y}^{(i)}_{\bar{s},a})$ is a martingale difference sequence with $\forall i : |\frac{1}{N(\bar{s},a)} Z_{\tau_i}(\boldsymbol{z}, X^{(i)}_{\bar{s},a}, a_{\tau_i}, \bar{Y}^{(i)}_{\bar{s},a})| \leq \frac{2}{N(\bar{s},a)}$ for any fixed $\boldsymbol{z}$ and $N(\bar{s},a) = n$ so that, by the Azuma-Hoeffding inequality (Lemma 3.3), the following holds:

$$\Pr(\frac{1}{N(\bar{s},a)} \sum_{i=1}^{N(\bar{s},a)} Z_{\tau_i} > \epsilon) \leq e^{-\frac{\epsilon^2}{2\frac{4}{N(\bar{s},a)^2} N(\bar{s},a)}} \tag{3.128}$$

$$= e^{-\frac{\epsilon^2}{\frac{8}{N(\bar{s},a)}}} \tag{3.129}$$

$$= e^{-\frac{1}{8} N(\bar{s},a)\epsilon^2}. \tag{3.130}$$

From (3.120) and (3.126), we then obtain

$$\Pr(||\bar{T}_Y(\cdot|\bar{s},a) - \bar{T}_{\omega_X}(\cdot|\bar{s},a)||_1 > \epsilon) = \Pr(\max_{\boldsymbol{z}\in\{-1,1\}^S} \frac{1}{N(\bar{s},a)} \sum_{i=1}^{N(\bar{s},a)} Z_{\tau_i} > \epsilon). \tag{3.131}$$

A union bound (Lemma 3.5) over all $2^{|\bar{S}|}$ vectors $\boldsymbol{z}$ for a fixed value of $N(s,a)$ shows

$$\Pr(\max_{\boldsymbol{z}\in\{-1,1\}^S} \frac{1}{N(\bar{s},a)} \sum_{i=1}^{N(\bar{s},a)} Z_{\tau_i} > \epsilon) \leq \sum_{\boldsymbol{z}\in\{-1,1\}^S} \Pr(\frac{1}{N(\bar{s},a)} \sum_{i=1}^{N(\bar{s},a)} Z_{\tau_i} > \epsilon). \tag{3.132}$$

So, using (3.130), we have that the following holds with probability $1 - 2^{|\bar{S}|} e^{-\frac{1}{8} N(\bar{s},a)\epsilon^2}$:

$$||\bar{T}_Y(\cdot|\bar{s},a) - \bar{T}_{\omega_X}(\cdot|\bar{s},a)||_1 \leq \epsilon. \tag{3.133}$$

□

Now we give the proof of Lemma 3.9:

Proof of Lemma 3.9. We follow the general structure of the proof of Lemma 8 by Strehl and Littman [14]. We have

$$E[Z_{\tau_i}(\boldsymbol{z}, X^{(i)}_{\bar{s},a}, a_{\tau_i}, \bar{Y}^{(i)}_{\bar{s},a})] = \sum_{c_{\tau_i+1}} \Pr(c_{\tau_i+1}) Z_{\tau_i}(\boldsymbol{z}, X^{(i)}_{\bar{s},a}, a_{\tau_i}, \bar{Y}^{(i)}_{\bar{s},a}) \tag{3.134}$$

$$= \sum_{c_{\tau_i}} \Pr(c_{\tau_i}) \sum_{\bar{Y}^{(i)}_{\bar{s},a}} \Pr(\bar{Y}^{(i)}_{\bar{s},a} | c_{\tau_i}, a_{\tau_i}) Z_{\tau_i}(\boldsymbol{z}, X^{(i)}_{\bar{s},a}, a_{\tau_i}, \bar{Y}^{(i)}_{\bar{s},a}) \tag{3.135}$$

$$= \sum_{c_{\tau_i}} \Pr(c_{\tau_i}) \sum_{\bar{Y}^{(i)}_{\bar{s},a}} \Pr(\bar{Y}^{(i)}_{\bar{s},a} | X^{(i)}_{\bar{s},a}, a_{\tau_i}) Z_{\tau_i}(\boldsymbol{z}, X^{(i)}_{\bar{s},a}, a_{\tau_i}, \bar{Y}^{(i)}_{\bar{s},a}). \tag{3.136}$$

The sum $\sum_{c_{\tau_i+1}}$ is over all possible sequences c_{τ_i+1} that end in a state $\bar{s}_{\tau_i+1}$, resulting from τ_i actions chosen by an agent following policy π. Conditioning on the sequence of random variables $Z_{\tau_1}, Z_{\tau_2}, \cdots, Z_{\tau_{i-1}}$ can make some sequences c_{τ_i} more likely and others less likely, that is

$$E[Z_{\tau_i}(\boldsymbol{z}, X^{(i)}_{\bar{s},a}, a_{\tau_i}, \bar{Y}^{(i)}_{\bar{s},a}) | Z_{\tau_1}, Z_{\tau_2}, \cdots, Z_{\tau_{i-1}}] \tag{3.137}$$

$$= \sum_{c_{\tau_i}} \Pr(c_{\tau_i} | Z_{\tau_1}, Z_{\tau_2}, \cdots, Z_{\tau_{i-1}}) \sum_{\bar{Y}^{(i)}_{\bar{s},a}} \Pr(\bar{Y}^{(i)}_{\bar{s},a} | X^{(i)}_{\bar{s},a}, a_{\tau_i}) Z_{\tau_i}(\boldsymbol{z}, X^{(i)}_{\bar{s},a}, a_{\tau_i}, \bar{Y}^{(i)}_{\bar{s},a}). \tag{3.138}$$

Significantly, since $P(\bar{Y}^{(i)}_{\bar{s},a} | \bar{s}_{\tau_i}, a_{\tau_i}, Z_{\tau_1}, \cdots, Z_{\tau_i-1}) = P(\bar{Y}^{(i)}_{\bar{s},a} | \bar{s}_{\tau_i}, a_{\tau_i})$, fixed values of $Z_{\tau_1}, Z_{\tau_2}, \cdots, Z_{\tau_{i-1}}$ do not influence the innermost sum of (3.138). For this innermost sum, we have

$$\sum_{\bar{Y}^{(i)}_{\bar{s},a}} \Pr(\bar{Y}^{(i)}_{\bar{s},a} | X^{(i)}_{\bar{s},a}, a_{\tau_i}) Z_{\tau_i}(\boldsymbol{z}, X^{(i)}_{\bar{s},a}, a_{\tau_i}, \bar{Y}^{(i)}_{\bar{s},a}) \tag{3.139}$$

$$= \sum_{\bar{Y}^{(i)}_{\bar{s},a}} \Pr(\bar{Y}^{(i)}_{\bar{s},a} | X^{(i)}_{\bar{s},a}, a_{\tau_i}) \left[z(\bar{Y}^{(i)}_{\bar{s},a}) - \sum_{\bar{s}'} T(\bar{s}' | X^{(i)}_{\bar{s},a}, a_{\tau_i}) z(\bar{s}') \right] \tag{3.140}$$

$$= \sum_{\bar{Y}^{(i)}_{\bar{s},a}} \Pr(\bar{Y}^{(i)}_{\bar{s},a} | X^{(i)}_{\bar{s},a}, a_{\tau_i}) z(\bar{Y}^{(i)}_{\bar{s},a}) - \sum_{\bar{Y}^{(i)}_{\bar{s},a}} \Pr(\bar{Y}^{(i)}_{\bar{s},a} | X^{(i)}_{\bar{s},a}, a_{\tau_i}) \sum_{\bar{s}'} T(\bar{s}' | X^{(i)}_{\bar{s},a}, a_{\tau_i}) z(\bar{s}') \tag{3.141}$$

$$= \sum_{\bar{Y}^{(i)}_{\bar{s},a}} \Pr(\bar{Y}^{(i)}_{\bar{s},a} | X^{(i)}_{\bar{s},a}, a_{\tau_i}) z(\bar{Y}^{(i)}_{\bar{s},a}) - \sum_{\bar{s}'} T(\bar{s}' | X^{(i)}_{\bar{s},a}, a_{\tau_i}) z(\bar{s}') \tag{3.142}$$

$$= 0. \tag{3.143}$$

So we conclude

$$E[Z_{\tau_i}(\boldsymbol{z}, X^{(i)}_{\bar{s},a}, a_{\tau_i}, \bar{Y}^{(i)}_{\bar{s},a}) | Z_{\tau_1}, Z_{\tau_2}, \cdots, Z_{\tau_{i-1}}] \tag{3.144}$$

$$= \sum_{c_{\tau_i}} \Pr(c_{\tau_i} | Z_{\tau_1}, Z_{\tau_2}, \cdots, Z_{\tau_{i-1}}) \sum_{\bar{s}_{\tau_i+1}} \Pr(\bar{s}_{\tau_i+1} | X^{(i)}_{\bar{s},a}, a_{\tau_i}) Z_{\tau_i}(\boldsymbol{z}, X^{(i)}_{\bar{s},a}, a_{\tau_i}, \bar{Y}^{(i)}_{\bar{s},a}) \tag{3.145}$$

$$= \sum_{c_{\tau_i}} \Pr(c_{\tau_i} | Z_{\tau_1}, Z_{\tau_2}, \cdots, Z_{\tau_{i-1}}) \times 0 \tag{3.146}$$

$$= 0. \tag{3.147}$$

□

3.8.4. PROOF SKETCH OF CLAIM 3.1

Proof sketch for Claim 3.1, which we repeat here:

Claim 3.1. The analysis and result of Theorem 3.4 can be extended to hold for approximate Q^* abstractions.

Proof sketch. To prove this claim, we will use a result from Abel, Hershkowitz, and Littman [16]. In their proof of Lemma 1, they show that the optimal policy for an abstract model $\bar{M}$, constructed using an approximate Q^* abstraction, is also near-optimal for the original problem M. Crucially, their proof uses that the Q^* values of the abstract model are close to those of the original problem.

Let $\hat{\bar{M}}$ denote the model learned through MBRLAO, constructed from $\bar{T}_Y(\cdot|\bar{s},a)$ and $\bar{T}_{\omega_X}(\cdot|\bar{s},a)$ in (3.16) and (3.19) (on pages 33 and 34, respectively). We can use Theorem 3.2 (on page 39) to show that learned model $\hat{\bar{M}}$ will be close to the abstract model $\bar{M}$ since the theorem is agnostic to the specific type of state abstraction. Because the models are close, we can show that their Q^* values are also close together. This can be shown through a proof similar to that of Lemma 3.8 (on page 54), but using Q-values instead of V-values. By extension, the Q^* values of the learned model will be close to the Q^* values of the original problem M. This result could replace the one of Lemma 3.11 (on page 61), which is used in the proof of Theorem 3.4 (on page 63).

In the proof of Theorem 3.4, the step from (3.169) to (3.170) can be skipped since we can directly compare (3.169) with (3.171) using the result described here. Then the rest of the proof follows, as it is agnostic to the type of abstraction used. □

3.8.5. R-MAX FROM ABSTRACTED OBSERVATIONS

Here we use the result of Theorem 3.2 to show that we can provide efficient learning guarantees for R-MAX [13] in RLAO. In Appendix 3.8.5, we use Theorem 3.2 and the value bounds in Appendix 3.8.1 to establish two supporting Lemmas. Then, in Appendix 3.8.5, we adapt one lemma and the guarantees of R-MAX to RLAO.

SUPPORTING LEMMAS

We can use Theorem 3.2 to determine the number of samples required to guarantee that the distance $||\bar{T}_Y(\cdot|\bar{s},a) - \bar{T}_{\omega_X}(\cdot|\bar{s},a)||_1$ will be smaller than ϵ with probability $1-\delta$:

Lemma 3.10. For inputs κ and ϵ $(0<\kappa<1, 0<\epsilon<2)$, the following holds for a number of samples $m \geq \frac{2[\ln(2^{|\bar{S}|}-2)-\ln(\kappa)]}{\epsilon^2}$:

$$\Pr(||\bar{T}_Y(\cdot|\bar{s},a) - \bar{T}_{\omega_X}(\cdot|\bar{s},a)||_1 \geq \epsilon) \leq \kappa. \tag{3.148}$$

Proof. To shorten the notation, we use the definitions $P_Y \triangleq \bar{T}_Y(\cdot|\bar{s},a)$ and $P_{\omega_X} \triangleq \bar{T}_{\omega_X}(\cdot|\bar{s},a)$. It follows from Theorem 3.2 that

$$\Pr(||P_Y - P_{\omega_X}||_1 \geq \epsilon) \leq 2^{|\bar{S}|} e^{-\frac{1}{8}m\epsilon^2}. \tag{3.149}$$

We need to select m such that $\kappa \geq 2^{|\bar{S}|} e^{-\frac{1}{8} m\epsilon^2}$:

$$\kappa \geq 2^{|\bar{S}|} e^{-\frac{1}{8} m\epsilon^2} \tag{3.150}$$

$$\frac{\kappa}{2^{|\bar{S}|}} \geq e^{-\frac{1}{8} m\epsilon^2} \tag{3.151}$$

$$\ln(\kappa) - \ln(2^{|\bar{S}|}) \geq -\frac{m\epsilon^2}{8} \tag{3.152}$$

$$\frac{m\epsilon^2}{8} \geq \ln(2^{|\bar{S}|}) - \ln(\kappa) \tag{3.153}$$

$$m \geq \frac{8[\ln(2^{|\bar{S}|}) - \ln(\kappa)]}{\epsilon^2}. \tag{3.154}$$

Thus if $m \geq \frac{8[\ln(2^{|\bar{S}|}) - \ln(\kappa)]}{\epsilon^2}$, we have

$$\Pr(||P_Y - P_{\omega_X}||_1 \geq \epsilon) \leq \kappa. \qquad \square$$

We want to give results for an empirical abstract model $\hat{\bar{M}}$ in the abstract space from ϕ, whose transition probabilities and rewards are within η_T and η_R, respectively, from those of an abstract MDP $\bar{M}$. We use $V^{*,n}$ to denote the value in M under the n-step optimal policy and $V^{\hat{\bar{\pi}}^*,n}$ to denote the value in M under the n-step optimal policy $\hat{\bar{\pi}}^*$ for $\hat{\bar{M}}$. The following lemma shows that we can upper bound the loss in value when applying $\hat{\bar{\pi}}^*$ to M:

Lemma 3.11. Let M be an MDP, $\bar{M}$ an abstract MDP constructed using an approximate model similarity abstraction ϕ, with η_R and η_T, and $\hat{\bar{M}}$ an MDP in the abstract space from ϕ with

$$|\bar{T}(\bar{s}'|\bar{s},a) - \hat{\bar{T}}(\bar{s}'|\bar{s},a)| \leq \epsilon, |\bar{R}(\bar{s},a) - \hat{\bar{R}}(\bar{s},a)| = 0. \tag{3.155}$$

Then

$$V^{*,n}(s) - V^{\hat{\bar{\pi}}^*,n}(s) \leq 2n\eta_R + (n-1)n(\eta_T + \epsilon)|\bar{S}|R_{\max}. \tag{3.156}$$

Proof. Note that we assume that $|\bar{R}(\bar{s},a) - \hat{\bar{R}}(\bar{s},a)| = 0$ because we assume a deterministic reward. Then, we have

$$\forall \bar{s}, a \in \bar{S} \times A, s \in \bar{s}: \quad |R(s,a) - \hat{\bar{R}}(\bar{s},a)| \leq \eta_R, \tag{3.157}$$

$$\forall \bar{s}, a, \bar{s}' \in \bar{S} \times A \times \bar{S}, s \in \bar{s}: \quad |\sum_{s' \in \bar{s}'} T(s'|s,a) - \hat{\bar{T}}(\bar{s}'|\bar{s},a)| \leq \eta_T + \epsilon. \tag{3.158}$$

We use $\hat{\bar{V}}^{\hat{\bar{\pi}}^*,n}(\bar{s})$ to denote the n-step value under the n-step optimal policy $\hat{\bar{\pi}}^{*,n}$ for the empirical abstract MDP $\hat{\bar{M}}$. Then, by Theorem 3.1, we have $\forall s \in \bar{s}, \bar{s} \in \bar{S}$:

$$V^{*,n}(s) - V^{\hat{\bar{\pi}}^*,n}(s) = 2n\eta_R + (n-1)n(\eta_T + \epsilon)|\bar{S}|R_{\max}. \tag{3.159}$$

$\square$

PROOF OF THEOREM 3.4

First, we restate an Implicit Explore or Exploit Lemma that is used in the proof of R-MAX. We are interested in the event A_M, the event that we encounter an unknown state-action pair during an n-step trail in M. For two MDPs with different dynamics only in the unknown state-action pairs, the probability that we encounter an unknown state-action pair in an n-step trial is tiny if the difference in the n-step value between the two MDPs is slight. The proof follows the steps the proof of Lemma 3 from Strehl and Littman [14].

Lemma 3.12 (Implicit Explore or Exploit)**.** Let M be an MDP. Let L be the set of known abstract state-action pairs, and let M_L be an MDP that is the same as M on the known pairs $(\bar{s}, a) \in L$, but different on the unknown pairs $(\bar{s}, a) \notin L$. Let s be some state, and A_M the event that an unknown abstract state-action pair is encountered in a trial generated by starting from state s_1 and following π for n steps in M. Then,

$$V_M^{\pi,n}(s_1) \geq V_{M_L}^{\pi,n}(s_1) - nR_{\max}\Pr(A_M). \tag{3.160}$$

Proof. For a fixed path $p_t = s_1, a_1, r_1, \cdots, s_t, a_t, r_t$, we define $\Pr_M(p_t)$ as the probability that p_t occurs when running policy π in M starting from state s_1. We let L_t be the set of paths p_t such that there is at least one unknown state s_i in p_t $(\phi(s_i), a) \notin L)$. We further let $r_M(t)$ be the reward received at time t and $r_M(p_t, t)$ the reward at time t in the path p_t. We have the following:

$$\begin{aligned}
E\big[r_{M_L}(t)\big] - E\big[r_M(t)\big] &= \sum_{p_t \in L_t} \Big(\Pr_{M_L}(p_t) r_{M_L}(p_t, t) - \Pr_M(p_t) r_M(p_t, t)\Big) && (3.161)\\
&+ \sum_{p_t \notin L_t} \Big(\Pr_{M_L}(p_t) r_{M_L}(p_t, t) - \Pr_M(p_t) r_M(p_t, t)\Big) && (3.162)\\
&= \sum_{p_t \notin L_t} \Big(\Pr_{M_L}(p_t) r_{M_L}(p_t, t) - \Pr_M(p_t) r_M(p_t, t)\Big) && (3.163)\\
&\leq \sum_{p_t \notin L_t} \Pr_{M_L}(p_t) r_{M_L}(p_t, t) \leq R_{\max}\Pr(A_M). && (3.164)
\end{aligned}$$

Here $\sum_{p_t \in L_t} \big(\Pr_{M_L}(p_t) r_{M_L}(p_t, t) - \Pr_M(p_t) r_M(p_t, t)\big) = 0$ because, by definition, M and M_L behave identically on the known state-action pairs, and $\sum_{p_t \notin L_t} \Pr_{M_L}(p_t) r_{M_L}(p_t, t) \leq R_{\max}\Pr(A_M)$ is true because $r_{M_L}(p_t, t)$ is at most $R_{\max}$. Finally we can write

$$\begin{aligned}
V_{M_L}^{\pi,n}(s_1) - V_M^{\pi,n}(s_1) &= \sum_{t=0}^{n} \big(E\big[r_{M_L}(t)\big] - E\big[r_M(t)\big]\big) && (3.165)\\
&\leq nR_{\max}\Pr(A_M). && (3.166)
\end{aligned}$$

Thus, $V_M^{\pi,n}(s_1) \geq V_{M_L}^{\pi,n}(s_1) - nR_{\max}\Pr(A_M)$. □

Now we are ready to prove the theorem.

Theorem 3.4. Given an MDP M, an approximate model similarity abstraction ϕ, with η_R and η_T, and inputs $|\bar{S}|, |A|, \epsilon, \delta, T_\epsilon$. With probability of at least $1-\delta$ the R-MAX algorithm adapted to abstraction (Algorithm 1) will attain an expected return of $\text{Opt}(\prod_M(\epsilon, T_\epsilon)) - 3\frac{g(\eta_T, \eta_R)}{T_\epsilon} - 2\epsilon$ within a number of steps polynomial in $|\bar{S}|, |A|, \frac{1}{\epsilon}\frac{1}{\delta}, T_\epsilon$.

Here T_ϵ is the ϵ-return mixing time of the optimal policy, the policies for M whose ϵ-return mixing time is T_ϵ are denoted by $\prod_M(\epsilon, T_\epsilon)$, the optimal expected T_ϵ-step undiscounted average return achievable by such policies are denoted by $\text{Opt}(\prod_M(\epsilon, T_\epsilon))$, and

$$g(\eta_T, \eta_R) = T_\epsilon \eta_R + \frac{(T_\epsilon - 1)\, T_\epsilon}{2} \eta_T |\bar{S}| R_{\max}.$$

Proof of Theorem 3.4. The proof uses elements of the Theorem from Brafman and Tennenholtz [13]. The proof follows the following steps:

1. We show that the expected average reward of the algorithm is at least as stated if the algorithm does not fail.
2. The probability of failing is at most δ. We can decompose this probability into three elements.
   a) Probability that the transition function estimates are not within the desired bounds.
   b) The probability that we do not attain the number of required visits in polynomial time.
   c) The probability that the actual return is lower than the expected return.

Now we first assume the algorithm does not fail. We define an abstract MDP $\bar{M}_{\omega_X}$ constructed from ϕ with η_T and η_R. Similar to M_L, $\bar{M}_{\omega_X,L}$ is the same as $\bar{M}_{\omega_X}$ on the known abstract state-action pairs, but with a self-loop and the maximum reward ($R_{\max}$) on the unknown abstract state-action pairs, i.e., $\forall (\bar{s}, a) \notin L : \bar{T}_{\omega_X,L}(\bar{s}|\bar{s}, a) = 1, \bar{R}_{\omega_X,L}(\bar{s}, a) = R_{\max}$. We also define an empirical abstract MDP $\bar{M}_Y$, of which the transition probabilities $\bar{T}_Y(\bar{s}'|\bar{s}, a)$ are within some ϵ_2 (defined later) of those in $\bar{M}_{\omega_X}$ and with $\bar{R}_{\omega_X}(\bar{s}, a) = \bar{R}_Y(\bar{s}, a)$ because of the assumption that the rewards are deterministic. Then, $\bar{M}_{Y,L}$ is the abstract MDP that is the same as $\bar{M}_Y$ on the known abstract state-action pairs and the same as $\bar{M}_{\omega_X,L}$ on the unknown abstract state-action pairs. We denote the R-MAX policy with $\bar{\pi}$.

Let A_M be the event that, following $\bar{\pi}$, we encounter an unknown abstract state-action pair $(\phi(s), a) \notin L$ in T_ϵ steps. From Lemma 3.12, we have that:

$$\forall s \in S : \quad V_M^{\bar{\pi},n}(s) \geq V_{M_L}^{\bar{\pi},n}(s) - T_\epsilon R_{\max} \Pr(A_M). \tag{3.167}$$

Now suppose that $R_{\max} \Pr(A_M) < \epsilon_1$, for some ϵ_1 (defined later), then we have

$$\begin{aligned}
V_M^{\bar{\pi},T_\epsilon}(s) &\geq V_{M_L}^{\bar{\pi},T_\epsilon}(s) - T_\epsilon R_{\max} \Pr(A_M) && (3.168)\\
&\geq V_{M_L}^{\bar{\pi},T_\epsilon}(s) - T_\epsilon \epsilon_1 && (3.169)\\
&\geq V_{\bar{M}_{\omega_X,L}}^{\bar{\pi},T_\epsilon}(s) - T_\epsilon \epsilon_1 - g(\eta_T, \eta_R) && (3.170)\\
&\geq V_{\bar{M}_{Y,L}}^{\bar{\pi},T_\epsilon}(s) - T_\epsilon \epsilon_1 - g(\epsilon_2) - g(\eta_T, \eta_R) && (3.171)\\
&\geq V_{\bar{M}_Y}^{*,T_\epsilon}(s) - T_\epsilon \epsilon_1 - g(\epsilon_2) - g(\eta_T, \eta_R) && (3.172)\\
&\geq V_M^{*,T_\epsilon}(s) - T_\epsilon \epsilon_1 - g(\epsilon_2) - g(\eta_T, \eta_R) - 2g(\eta_T + \epsilon_2, \eta_R). && (3.173)
\end{aligned}$$

Here the step from (3.168) to (3.169) follows because of the assumption that $R_{\max}\Pr(A_M) < \epsilon_1$. The step from (3.169) to (3.170) follows from Lemma 3.6, where $g(\eta_T, \eta_R) = T_\epsilon \eta_R + \frac{(T_\epsilon - 1)T_\epsilon}{2}\eta_T |\bar{S}| R_{\max}$. The step from (3.170) to (3.171) follows from Lemma 3.8, where $g(\epsilon_2) = \frac{(T_\epsilon - 1)T_\epsilon}{2}\epsilon_2 |\bar{S}| R_{\max}$. The step from (3.171) to (3.172) follows because the R-MAX policy $\bar{\pi}$ is the optimal policy for $\bar{M}_{Y,L}$, and $\bar{M}_{Y,L}$ is the same as $\bar{M}_Y$ on the known state-action pairs and overestimates the value of the unknown state-action pairs (to the maximum value). Finally, the step from (3.172) to (3.173) follows from Lemma 3.11.

In (3.173) the results are for the undiscounted T_ϵ-step sum of rewards, so to obtain the result for the average reward per step, we have to divide (3.173) by T_ϵ. We get

$$\text{Opt}(\prod_M(\epsilon, T_\epsilon)) - T_\epsilon\epsilon_1/T_\epsilon - g(\epsilon_2)/T_\epsilon - g(\eta_T, \eta_R)/T_\epsilon - 2g(\eta_T + \epsilon_2, \eta_R)/T_\epsilon \tag{3.174}$$

$$= \text{Opt}(\prod_M(\epsilon, T_\epsilon)) - \epsilon_1 - \frac{(T_\epsilon - 1)T_\epsilon}{2}\epsilon_2 |\bar{S}| R_{\max}/T_\epsilon$$

$$- (T_\epsilon \eta_R + \frac{(T_\epsilon - 1)T_\epsilon}{2}\eta_T |\bar{S}| R_{\max})/T_\epsilon - 2(T_\epsilon \eta_R + \frac{(T_\epsilon - 1)T_\epsilon}{2}(\eta_T + \epsilon_2)|\bar{S}| R_{\max})/T_\epsilon \tag{3.175}$$

$$= \text{Opt}(\prod_M(\epsilon, T_\epsilon)) - \epsilon_1 - \frac{(T_\epsilon - 1)}{2}\epsilon_2 |\bar{S}| R_{\max}$$

$$- \eta_R - \frac{(T_\epsilon - 1)}{2}\eta_T |\bar{S}| R_{\max} - 2\eta_R - (T_\epsilon - 1)(\eta_T + \epsilon_2)|\bar{S}| R_{\max} \tag{3.176}$$

$$= \text{Opt}(\prod_M(\epsilon, T_\epsilon)) - \epsilon_1 - \frac{(T_\epsilon - 1)}{2}\epsilon_2 |\bar{S}| R_{\max}$$

$$- 3\eta_R - \frac{(T_\epsilon - 1)}{2}\eta_T |\bar{S}| R_{\max} - (T_\epsilon - 1)\epsilon_2 |\bar{S}| R_{\max} - (T_\epsilon - 1)\eta_T |\bar{S}| R_{\max} \tag{3.177}$$

$$= \text{Opt}(\prod_M(\epsilon, T_\epsilon)) - \epsilon_1 - 3\frac{(T_\epsilon - 1)}{2}\epsilon_2 |\bar{S}| R_{\max} - 3\eta_R - 3\frac{(T_\epsilon - 1)}{2}\eta_T |\bar{S}| R_{\max} \tag{3.178}$$

$$= \text{Opt}(\prod_M(\epsilon, T_\epsilon)) - \epsilon_1 - 3\frac{(T_\epsilon - 1)}{2}\epsilon_2 |\bar{S}| R_{\max} - 3\frac{g(\eta_T, \eta_R)}{T_\epsilon} \tag{3.179}$$

$$= \text{Opt}(\prod_M(\epsilon, T_\epsilon)) - \frac{3}{8}\epsilon - 3\frac{(T_\epsilon - 1)}{2}\epsilon_2 |\bar{S}| R_{\max} - 3\frac{g(\eta_T, \eta_R)}{T_\epsilon} \tag{3.180}$$

$$= \text{Opt}(\prod_M(\epsilon, T_\epsilon)) - \frac{3}{8}\epsilon - 3\frac{(T_\epsilon - 1)}{2}\frac{3\epsilon}{4|\bar{S}| R_{\max}(T_\epsilon - 1)}|\bar{S}| R_{\max} - 3\frac{g(\eta_T, \eta_R)}{T_\epsilon}. \tag{3.181}$$

$$= \text{Opt}(\prod_M(\epsilon, T_\epsilon)) - \frac{3}{8}\epsilon - \frac{9}{8}\epsilon - 3\frac{g(\eta_T, \eta_R)}{T_\epsilon}. \tag{3.182}$$

$$= \text{Opt}(\prod_M(\epsilon, T_\epsilon)) - \frac{3}{2}\epsilon - 3\frac{g(\eta_T, \eta_R)}{T_\epsilon}. \tag{3.183}$$

In the step from (3.178) to (3.179) we use that $g(\eta_T, \eta_R) = T_\epsilon \eta_R + \frac{(T_\epsilon - 1)T_\epsilon}{2}\eta_T |\bar{S}| R_{\max}$. Then, in the last steps, we define ϵ_1 and ϵ_2. In the step from (3.179) to (3.180) we set $\epsilon_1 = \frac{3}{8}\epsilon$. And in the step from (3.180) to (3.181) we set $\epsilon_2 = 3\epsilon/(4|\bar{S}| R_{\max}(T_\epsilon - 1))$.

The above assumed that the algorithm did not fail, but we cannot guarantee this with probability 1 within a number of steps that is polynomial in the input. We

will show that we can upper bound the probability of failure by δ. There are three reasons why the algorithm could fail.

1. First, we need to show that the transition functions of $\bar{M}_Y$ are within $\eta_T + \epsilon_2$ of the transition functions of $\bar{M}_{\omega_X}$, with high probability. This is to ensure that, once all the abstract state-action pairs are known, the loss of value because of an inaccurate transition model, $V^{*,T_\epsilon}_{\bar{M}_Y} - V^{*,T_\epsilon}_{M}$ is within $2g(\eta_T + \epsilon_2, \eta_R) = 2T_\epsilon\eta_R + (T_\epsilon - 1)T_\epsilon(\eta_T + \epsilon_2)|\bar{S}|R_{\max}$ by Lemma 3.11. We can use the martingale concentration inequality to choose a number of samples K_1 so that the probability that our transition estimate is outside the desired bound is less than $\frac{\delta}{3|\bar{S}||A|}$ for every abstract state-action pair if we sample each pair K_1 times. By Lemma 3.10, we can guarantee this by using $K_1 \geq \frac{2[\ln(2^{|\bar{S}|}-2)-\ln(\delta/(3|\bar{S}||A|))]}{(\frac{3\epsilon}{4|\bar{S}|R_{\max}(T_\epsilon-1)})^2} = \frac{32|\bar{S}|^2R^2_{\max}(T_\epsilon-1)^2[\ln(2^{|\bar{S}|}-2)-\ln(\delta/(3|\bar{S}||A|))]}{9\epsilon^2}$. Then, by applying the Union Bound on all $|\bar{S}|A$ pairs, we have that the total probability that any transition function is outside the desired bound is less than $\delta/3$.

2. Before we assumed that $R_{\max}\Pr(A_M) < \epsilon_1 (= \frac{3\epsilon}{8})$. Here we can show that after K_2 T_ϵ-step trials where $R_{\max}\Pr(A_M) \geq \frac{3\epsilon}{8}$, all the abstract state-action pairs are visited at least K_1 times (become known) with a probability of at least $1-\delta/3$ by using Hoeffding's Inequality. Let X_i be the indicator variable that is 1 if we visit an unknown abstract state-action pair in a trial, and 0 otherwise. For the trials where

$$R_{\max}\Pr(X_i = 1) \geq \frac{3\epsilon}{8}$$
$$\Pr(X_i = 1) \geq \frac{3\epsilon}{8}/R_{\max},$$

we can use Hoeffding's Inequality to establish an upper bound, we have:

$$\Pr(\sum_{i=1}^{K_2}((\frac{3\epsilon}{8}/R_{\max}) - X_i) \geq K_2^{2/3}) = \tag{3.184}$$
$$\Pr(\frac{3\epsilon}{8}\frac{K_2}{R_{\max}} - \sum_{i=1}^{K_2} X_i \geq K_2^{2/3}) \leq e^{-\frac{2(K_2^{2/3})^2}{K_2}} \leq e^{-\frac{2(K_2^{2/3})^2}{K_2}} = e^{-2K_2^{1/3}}, \tag{3.185}$$
$$\Pr(\frac{3\epsilon}{8}\frac{K_2}{R_{\max}} - K_2^{2/3} \geq \sum_{i=1}^{K_2} X_i) \leq e^{-2K_2^{1/3}}. \tag{3.186}$$

After K_2 exploring episodes we want $\sum_{i=1}^{K_2} X_i$, the number of visits to unknown state-action pairs, to be $K_1|\bar{S}||A|$. So we can choose K_2 such that $\frac{3\epsilon}{8}\frac{K_2}{R_{\max}} - K_2^{2/3} \geq K_1|\bar{S}||A|$, and $e^{-2K_2^{1/3}} \leq \delta/3$ to guarantee that the probability of failing to explore enough is at most $\delta/3$.

3. Finally, the actual return may be lower than the expected return when we perform a T_ϵ-step trial where we do not explore. We use Hoeffding's Inequality to determine the number of steps K_3 needed to ensure that the actual average

return is within $\epsilon/2$ of $\text{Opt}(\prod_M(\epsilon, T_\epsilon)) - \frac{3}{2}\epsilon - 3\frac{g(\eta_T,\eta_R)}{T_\epsilon}$. We need to choose K_3 so that the probability of obtaining an actual return that is smaller than the desired $\text{Opt}(\prod_M(\epsilon, T_\epsilon)) - 2\epsilon - 3\frac{g(\eta_T,\eta_R)}{T_\epsilon}$ is at most $\delta/3$ within $K_3 = Z|\bar{S}|T_\epsilon$ exploitation steps, with some number $Z > 0$. Let X_i denote the average return in the i-th exploitation step and μ the average expected return in an exploitation step so that μ is at least $\text{Opt}(\prod_M(\epsilon, T_\epsilon)) - \frac{3}{2}\epsilon - 3\frac{g(\eta_T,\eta_R)}{T_\epsilon}$. Then

$$\Pr(\sum_{i=1}^{K_3}(\frac{\mu - X_i}{R_{\max}}) \geq K_3^{2/3}) \leq e^{-2\frac{(K_3^{2/3})^2}{K_3}} = e^{-2K_3^{1/3}}. \tag{3.187}$$

This means that, with a probability of at most $e^{-2K_3^{1/3}}$, the average return for K_3 exploitation steps is more than $\frac{R_{\max}}{K_3^{1/3}}$ lower than μ:

$$\Pr(\sum_{i=1}^{K_3}(\frac{\mu - X_i}{R_{\max}}) \geq K_3^{2/3}) \leq e^{-2K_3^{1/3}}, \tag{3.188}$$

$$\Pr(K_3\frac{\mu}{R_{\max}} - \sum_{i=1}^{K_3}\frac{X_i}{R_{\max}} \geq K_3^{2/3}) \leq e^{-2K_3^{1/3}}, \tag{3.189}$$

$$\Pr(K_3\mu - \sum_{i=1}^{K_3} X_i \geq R_{\max}K_3^{2/3}) \leq e^{-2K_3^{1/3}}, \tag{3.190}$$

$$\Pr(\mu - \sum_{i=1}^{K_3}\frac{X_i}{K_3} \geq \frac{R_{\max}}{K_3^{1/3}}) \leq e^{-2K_3^{1/3}}. \tag{3.191}$$

We can now choose Z, so that $\epsilon/2 \leq \frac{R_{\max}}{(Z|\bar{S}|T_\epsilon)^{\frac{1}{3}}}$ and $e^{-2(Z|\bar{S}|T_\epsilon)^{1/3}} \leq \delta/3$, to get the desired result: with probability at most $\delta/3$ the obtained value will be more than $\epsilon/2$ lower than the expected value.

The probability of failure is thus at most $3 * \delta/3 = \delta$, and an average return at most $2\epsilon + 3\frac{g(\eta_T,\eta_R)}{T_\epsilon}$ lower than $\text{Opt}(\prod_M(\epsilon, T_\epsilon))$ will be obtained with a probability of at least $1 - \delta$. □

3.8.6. SIMULATOR DATA COLLECTION

Here we assume that we have access to a simulator and use this in our procedure to give a guarantee in the form of the abstract L1 inequality from (3.21). To some extent, this is not surprising, but to the best of our knowledge, this is the first work that explicitly shows how to combine MBRL and abstraction using a simulator. We assume that the simulator allows us to select (or move to) any state and draw a sample from its transition function, which we call the independent samples assumption:

Assumption 3.1 (Independent samples)**.** We assume we can obtain independent samples, e.g., for any state-action pair (s, a), we can draw samples directly from its transition function $T(\cdot|s, a)$.

Algorithm 2 Procedure: MBRLAO

Input: $M, \phi, \delta, \epsilon, \pi$
$\bar{Y}$ = COLLECTSAMPLES$(M, \phi, \delta, \epsilon, \pi)$
The sampling results in sequences $\bar{Y}_{\bar{s},a}$, one for every pair $(\bar{s}, a)$:
$\bar{Y}_{\bar{s},a} = \phi(s'^{(1)}), \cdots, \phi(s'^{(m)})$
$= \bar{s}'^{(1)}, \cdots, \bar{s}'^{(m)}$
for all $(\bar{s}, a, \bar{s}') \in \bar{S} \times A \times \bar{S}$ **do**
  $\bar{T}_Y(\bar{s}'|\bar{s}, a) = \frac{1}{m}\sum_{i=1}^{m} \mathbb{1}\{\bar{Y}_{\bar{s},a}^{(i)} = \bar{s}'\}$
end for
$\bar{M}_Y \triangleq \langle \bar{S}, A, \bar{T}_Y, \bar{R}, \gamma \rangle$
$\bar{\pi}_Y^*$ = Value Iteration$(\bar{M}_Y)$
Apply $\bar{\pi}_Y^*$ to M

Algorithm 3 COLLECTSAMPLES with Simulator

Input: $M, \phi, \delta, \epsilon$
$\kappa = \frac{\delta}{|\bar{S}||A|}$
$m = \lceil \frac{2[\ln(2^{|\bar{S}|}-2)-\ln(\kappa)]}{\epsilon^2} \rceil$
for all $(\bar{s}, a) \in \bar{S} \times A$ **do**
  $\bar{Y}_{\bar{s},a} = [\ ]$
  $x_{\bar{s},a}$ = select a prototype state $s \in \bar{s}$
  for $i = 1 : m$ **do**
    s' = Sample$(T(\cdot|x_{\bar{s},a}, a))$
    $\bar{Y}_{\bar{s},a}$.append$(\phi(s'))$
  end for
end for
Return: all $\bar{Y}_{\bar{s},a}$

If a simulator of the MDP is available, this is a reasonable assumption. For every pair $(\bar{s}, a)$, the simulator sampling procedure (Algorithm 3) selects a prototype $x_{\bar{s},a} \in \bar{s}$ from which to sample. We define a weighting function $\omega_X(s, a)$ that has a weight of 1 if s is the prototype $x_{\bar{s},a}$ and 0 otherwise:

$$\forall_{(\bar{s},a),s\in\bar{s}}\ \omega_X(s, a) \triangleq \mathbb{1}\{s = x_{\bar{s},a}\}. \tag{3.192}$$

Then we use this ω_X to define the abstract transition function $\bar{T}_{\omega_X}$ according to (3.10). $\bar{T}_{\omega_X}(\bar{s}'|\bar{s}, a) = \sum_{s'\in\bar{s}'} T(s'|s = x_{\bar{s},a}, a)$. This way, the samples we collect for one pair $(\bar{s}, a)$ are i.i.d. They are independent because of Assumption 3.1 and identically distributed because we sample from the prototype. Because the samples are i.i.d., we can use Lemma 3.1. We show that, with the simulator we can combine MBRL with abstraction and still learn an accurate model. We can guarantee that $\bar{T}_Y$ will be close to $\bar{T}_{\omega_X}$, with a high probability:

Theorem 3.5. Under assumption 3.1, following the procedure in Algorithm 1, with the data collection from Algorithm 3 and inputs $|\bar{S}|, A, \epsilon$, and δ. For $\bar{T}_Y$ constructed

by the algorithm, we have that with probability $1-\delta$, the following holds:

$$\forall_{(\bar{s},a)} \; ||\bar{T}_Y(\cdot|\bar{s},a) - \bar{T}_{\omega_X}(\cdot|\bar{s},a)||_1 \leq \epsilon. \tag{3.193}$$

PROOF OF THEOREM 3.5

Before starting with the actual proof, we first go over Algorithm 3 and give two lemmas the proof uses.

The agent will draw samples using the simulator as described in Algorithm 3. Since we assume we can sample directly from the transition functions $T(\cdot|s,a)$, this algorithm loops over all pairs $(\bar{s},a)$ and samples m times[10] from each transition function. More formally, for every pair $(\bar{s},a)$, the algorithm selects one prototype state $x_{\bar{s},a} = s \in \bar{s}$. Then, it loops over every pair $(\bar{s},a)$ and samples m transitions from $T(\cdot|x_{\bar{s},a},a)$. The set of collected experiences for each abstract state-action pair $(\bar{s},a)$ is represented by $\bar{Y}_{\bar{s},a}$, as defined by (3.15).

Given $\bar{Y}_{\bar{s},a}$, we define the learned model $\bar{T}_Y(\cdot|\bar{s},a)$ according to (3.16), $\bar{T}_{\omega_X}$ according to (3.19), and ω_X according to (3.192). It follows from Lemma 3.1 that we can derive a number of samples that we require to guarantee that $\Pr(||\bar{T}_Y(\cdot|\bar{s},a) - \bar{T}_{\omega_X}(\cdot|\bar{s},a)||_1 \geq \epsilon) \leq \kappa$ is true for inputs κ and ϵ:

Lemma 3.13. For inputs κ and ϵ $(0 < \kappa < 1, 0 < \epsilon < 2)$, we have that the following holds for $m \geq \frac{2[\ln(2^{|\bar{S}|}-2)-\ln(\kappa)]}{\epsilon^2}$:

$$\Pr(||\bar{T}_Y(\cdot|\bar{s},a) - \bar{T}_{\omega_X}(\cdot|\bar{s},a)||_1 \geq \epsilon) \leq \kappa. \tag{3.194}$$

Proof. To shorten the notation, we use the definitions $P_Y \triangleq \bar{T}_Y(\cdot|\bar{s},a)$ and $P_{\omega_X} \triangleq \bar{T}_{\omega_X}(\cdot|\bar{s},a)$. From Lemma 3.1, we have that

$$\Pr(||P_Y - P_{\omega_X}||_1 \geq \epsilon) \leq (2^{|\bar{S}|} - 2)e^{-\frac{1}{2}m\epsilon^2}. \tag{3.195}$$

We need to select m such that $\kappa \geq (2^{|\bar{S}|} - 2)e^{-\frac{1}{2}m\epsilon^2}$:

$$\kappa \geq (2^{|\bar{S}|} - 2)e^{-\frac{1}{2}m\epsilon^2} \tag{3.196}$$

$$\frac{\kappa}{2^{|\bar{S}|} - 2} \geq e^{-\frac{1}{2}m\epsilon^2} \tag{3.197}$$

$$\ln(\kappa) - \ln(2^{|\bar{S}|} - 2) \geq -\frac{m\epsilon^2}{2} \tag{3.198}$$

$$\frac{m\epsilon^2}{2} \geq \ln(2^{|\bar{S}|} - 2) - \ln(\kappa) \tag{3.199}$$

$$m \geq \frac{2[\ln(2^{|\bar{S}|} - 2) - \ln(\kappa)]}{\epsilon^2} \tag{3.200}$$

Thus, if $m \geq \frac{2[\ln(2^{|\bar{S}|}-2)-\ln(\kappa)]}{\epsilon^2}$ we have

$$\Pr(||P_Y - P_{\omega_X}||_1 \geq \epsilon) \leq \kappa. \qquad \square$$

[10] The value of m in Algorithm 3 is chosen based on the results further along in this section.

Using the Union bound, we can give a lower bound on the probability that $\bar{T}_Y(\cdot|\bar{s},a)$ and $\bar{T}_{\omega_X}(\cdot|\bar{s},a)$ are ϵ close for every $(\bar{s},a)$:

Lemma 3.14. If

$$\forall_{(\bar{s},a)} \ \left[\Pr(||\bar{T}_Y(\cdot|\bar{s},a) - \bar{T}_{\omega_X}(\cdot|\bar{s},a)||_1 \geq \epsilon)\right] \leq \frac{\delta}{|\bar{S}||A|} \tag{3.201}$$

then the following holds with a probability of at least $1-\delta$:

$$\max_{(\bar{s},a)} \left[||\bar{T}_Y(\cdot|\bar{s},a) - \bar{T}_{\omega_X}(\cdot|\bar{s},a)||_1\right] \leq \epsilon. \tag{3.202}$$

Proof. We define

$$\Delta_{\bar{s},a} \triangleq ||\bar{T}_Y(\cdot|\bar{s},a) - \bar{T}_{\omega_X}(\cdot|\bar{s},a)||_1. \tag{3.203}$$

Then $\Pr(\max_{(\bar{s},a)}\{\Delta_{\bar{s},a} \geq \epsilon\})$ is the probability that $\Delta_{\bar{s},a} \geq \epsilon$ for at least one abstract state-action pair. From the union bound, it follows that $\Pr(\max_{(\bar{s},a)}\{\Delta_{\bar{s},a} \geq \epsilon\}) \leq \delta$:

$$\Pr(\max_{(\bar{s},a)}\{\Delta_{\bar{s},a} \geq \epsilon\}) \leq \sum_{\bar{s},a} \Pr(\Delta_{\bar{s},a} \geq \epsilon) \tag{3.204}$$

$$\leq \sum_{\bar{s},a} \frac{\delta}{|\bar{S}||A|} \tag{3.205}$$

$$= \delta. \tag{3.206}$$

It follows that $\Pr(\max_{(\bar{s},a)}\{\Delta_{\bar{s},a} \leq \epsilon\}) \geq 1-\delta$ since $\Pr(\max_{(\bar{s},a)}\{\Delta_{\bar{s},a} \leq \epsilon\}) = 1 - \Pr(\max_{(\bar{s},a)}\{\Delta_{\bar{s},a} \geq \epsilon\})$. Thus the probability that (3.202) holds is at least $1-\delta$. □

Now we are ready to proof Theorem 3.5:

Proof of Theorem 3.5. By Assumption 3.1, and the earlier assumption that $|S|$ and $|A|$ are finite, we have that we can obtain m samples in finite time for every abstract state-action pair and any $m > 0$. Given the inputs $|\bar{S}|, A, \epsilon$, and δ, Algorithm 3 sets $m = \lceil \frac{2[\ln(2^{|\bar{S}|}-2)-\ln(\kappa)]}{\epsilon^2} \rceil$, where $\kappa = \frac{\delta}{|\bar{S}||A|}$. Then, for every $(\bar{s},a)$, a prototype state $x_{\bar{s},a} = s \in \bar{s}$ is selected. We use (3.192) to define ω_X and (3.19) to define $\bar{T}_{\omega_X}$.

For all $(\bar{s},a)$, Algorithm 3 obtains a sequence $\bar{Y}_{\bar{s},a}$ by sampling from the transition function from the prototype state $x_{\bar{s},a}$ and Algorithm 2 constructs the empirical transition functions as in (3.16).

Given our choice of m and the inputs $\kappa = \frac{\delta}{|\bar{S}||A|}$ and ϵ, it follows from Lemma 3.10 that

$$\forall_{(\bar{s},a)} \ \Pr(||\bar{T}_Y(\cdot|\bar{s},a) - \bar{T}_{\omega_X}(\cdot|\bar{s},a)||_1 \geq \epsilon) \leq \frac{\delta}{|\bar{S}||A|}. \tag{3.207}$$

By the union bound, we have that the following holds with a probability of at least $1-\delta$:

$$\forall_{(\bar{s},a)} \ ||\bar{T}_Y(\cdot|\bar{s},a) - \bar{T}_{\omega_X}(\cdot|\bar{s},a)||_1 \leq \epsilon. \tag{3.208}$$

□

4

ABSTRACTION FOR BAYESIAN RL IN FACTORED POMDPS

Bayesian reinforcement learning provides an elegant solution to addressing the exploration–exploitation trade-off in Partially Observable Markov Decision Processes (POMDPs) when the environment's dynamics and reward function are initially unknown. By maintaining a belief over these unknown components and the state, the agent can effectively learn the environment's dynamics and optimize their policy. However, scaling Bayesian reinforcement learning methods to large problems remains to be a significant challenge. While prior work has leveraged factored models and online sample-based planning to address this issue, these approaches often retain unnecessarily complex models and factors within the belief space that have minimal impact on the optimal policy. While this complexity might be necessary for accurate model learning, in reinforcement learning, the primary objective is not to recover the ground truth model but to optimize the policy for maximizing the expected sum of rewards. Abstraction offers a way to reduce model complexity by removing factors that are less relevant to achieving high rewards. In this work, we propose and analyze the integration of abstraction with online planning in factored POMDPs. Our empirical results demonstrate two key benefits. First, abstraction reduces model size, enabling faster simulations and thus more planning simulations within a fixed runtime. Second, abstraction enhances performance even with a fixed number of simulations due to greater statistical strength. These results underscore the potential of abstraction to improve both the scalability and effectiveness of Bayesian reinforcement learning in factored POMDPs.

Parts of this chapter have been published in Transactions on Machine Learning Research [55].

4.1. INTRODUCTION

Deep reinforcement learning methods have achieved significant milestones, such as attaining superhuman performance on Atari games with only 100k frames [79], solving highly complex games such as Go [146], and achieving high performance in simulated control tasks [147]. Most of these achievements rely on recent function approximation advances made with the work on deep neural networks. However, despite these advances, Reinforcement Learning (RL) still faces critical hurdles that must be addressed to enable its application in diverse real-world scenarios. One of the most pressing challenges is the high sample complexity of deep RL methods, which remains problematic in real-world applications where data collection is expensive, difficult, or dangerous. Fortunately, many such applications offer prior knowledge that can be leveraged to reduce sample complexity. To effectively utilize this knowledge, it is crucial to move away from the tabula rasa approaches of neural networks and incorporate domain-specific prior knowledge into the learning process [56–58].

Another critical challenge is exploration, which is strongly tied to sample complexity. Effective exploration of unknown and interesting parts of the environment is essential in almost every application. A better exploration strategy means faster learning and improved sample efficiency. Importantly, in RL, an agent must balance exploration (i.e., learning) with exploitation (i.e., maximizing reward). Most deep RL methods rely on heuristics for exploration, which can perform poorly in complex domains [66]. These challenges are particularly pronounced in partially observable environments, where agents must make decisions based on limited information.

Model-based Bayesian RL (BRL) [59] offers a principled approach to addressing the exploration-exploitation trade-off by maintaining a belief over the environment's state and dynamics. This belief enables the agent to balance the exploration–exploitation trade-off effectively. In this work, we build on the Factored Bayes-Adaptive POMDP (FBA-POMDP) framework [60, 61], a model-based BRL approach that combines partial observability and structured factored models. FBA-POMDPs incorporate factorized representations of the environment's dynamics, allowing agents to exploit problem structure for improved scalability. Additionally, thanks to its Bayesian nature, prior knowledge can be incorporated into FBA-POMDPs in a principled way via Bayesian priors, further improving sample efficiency.

Despite its advantages, the FBA-POMDP framework faces significant challenges. While factorization enables better generalization, the inclusion of irrelevant state factors in the model can lead to unnecessarily large model spaces. This increases computational demands during planning and reduces statistical strength by hypothesizing dependencies that are irrelevant to maximizing rewards. For instance, in a cluttered environment, an agent may only need to consider the positions of objects to navigate effectively, while features such as their colors or shapes are irrelevant for the reward. Abstracting away such unnecessary details can simplify the model space, improve computational efficiency, and enhance learning performance. Previous studies have demonstrated that even lossy abstractions can improve performance in planning [148, 149]. This is because simplified models

can generate more simulations within a fixed runtime, potentially leading to better results in sampling-based online planning. Motivated by this insight, we propose incorporating abstraction into the FBA-POMDP framework to improve scalability and learning efficiency.

We focus on discrete Factored POMDPs (F-POMDPs) and explore the application of abstraction within the FBA-POMDPs framework to enhance planning efficiency, scalability, and learning performance. To achieve this, we augment Factored BA-POMCP (FBA-POMCP) [60, 61], an established online planning and learning algorithm for FBA-POMDPs, to incorporate multiple levels of abstraction. This contribution addresses abstraction discovery in the partially observable reinforcement learning setting, where the lack of access to the true model and the complexity of dealing with partial observability make the creation and utilization of abstractions both important and challenging.

We base ourselves on a previous abstraction method for *planning* in fully observable factored Markov decision processs (MDPs) [135]. It creates abstractions automatically based on the problem's structure, enabling agents to plan and learn more effectively. Our approach extends this idea to a partially observable *learning* setting, where the underlying dynamics are not known beforehand, introducing additional complexities. In this setting, the combination of an unknown model, partial observability, and inherent inexactness of approximate abstractions makes it difficult to evaluate abstractions in advance, making the ability to dynamically update abstract models during learning crucial. To address this challenge, we propose to automatically construct abstractions based on the hypothesized connectivity of state factors to the reward function and leverage the Bayes-adaptive framework to adapt the abstract model over time. A key feature of our approach is its ability to maintain a belief over multiple candidate structures and their corresponding abstract models, ensuring consistent belief updates of the (abstract) models.

Our work represents a novel step toward combining abstraction with BRL in F-POMDPs. Empirically, we demonstrate that abstraction improves performance in two critical ways: (1) by reducing model size, allowing for more simulations within a given computation time, and (2) by simplifying the learning problem, leading to faster learning and improved performance in fewer episodes. These findings highlight the potential of abstraction to address key challenges in BRL for F-POMDPs and open a promising direction for future research into leveraging abstractions to improve scalability, exploration, and decision-making in complex, real-world environments.

4.2. BACKGROUND

In this section, we introduce the rich body of literature that our work builds upon. In particular, Section 4.2.1 introduces the POMDP as the general mathematical model for sequential decision-making. We then describe (model-based) Bayesian reinforcement learning in factored POMDPs in Section 4.2.2, which is a (belief-space) Partially-observable Markov decision process itself. Lastly, we discuss algorithmic approaches for solving these decision problems in Section 4.2.3.

4.2.1. POMDPS AND FACTORIZATION

Sequential decision-making in stochastic domains with hidden states can be formalized as a POMDP [19, 150]. The POMDP is defined by the tuple $(\mathbb{S}, \mathbb{A}, \mathbb{O}, \mathscr{D}, \mathscr{R}, \gamma, H)$, where $\mathbb{S}$, $\mathbb{A}$, and $\mathbb{O}$ are the (discrete) set of states, actions, and observations, respectively. The dynamics $\mathscr{D}$ specify the system's transition probabilities $\mathscr{D} \in \mathbb{D}$: $(\mathbb{S} \times \mathbb{A}) \to \Delta(\mathbb{S} \times \mathbb{O})$, and $\mathscr{R}$: $(\mathbb{S} \times \mathbb{A} \times \mathbb{S}) \to \mathbb{R}$ is the reward function. The (maximum) number of time steps in an episode is the horizon $H \in \mathbb{Z}$, and $\gamma \in [0, 1]$ is the discount factor.

The goal of the agent is to maximize the discounted return, $\sum_t \gamma^t r_t$. To do so, it can use the observable action ($a \in \mathbb{A}$) and observation ($o \in \mathbb{O}$) history $h_t = (a_0, o_1, \ldots, a_{t-1}, o_t)$, or it can use the belief as a sufficient statistic. The belief is the probability distribution over the current state $b \in \mathbb{B}$: $\Delta\mathbb{S}$, which can be updated with Bayes' rule: $b'(s') = \tau(b, a, o)(s') \propto \sum_s \mathscr{D}(s', o | s, a) b(s)$. However, in most problems, the computation of the belief update is intractable. Section 4.2.3 will cover how to find a solution.

Running Example As a simple intuitive example, consider the Corridor domain shown in Figure 4.1a. The agent starts at the "Start" location and its goal is to reach the "Reward" location. As depicted in the figure, there is also a "Boots" location and a "Button" location. To reach the reward the "Door" must be open. There is also always a "Person" present in the environment. There are 8 different persons, with exactly one present during each episode. The probability of the person opening the door at some point during an episode varies depending on which person is present, but this probability is generally very low, ranging from about 1.25% to 10%. The agent cannot interact with the person.

The state consists of the location of the agent, the person present and the binary statuses of the boots, button, and door. The agent has four actions: move *left* or *right*, *put on boots*, *push button*, and *lock pick* the door. Moving left or right succeeds 30% of the time without boots and 95% of the time with boots. The probability of success for both *put on boots* and *push button* is 90%, provided the agent is in the correct location. When the button is pressed, there is a 100% chance that the door opens. The *lock pick* action only has an effect when the agent is standing next to the door and has a 40% chance of success. The agent can also open the door by "bashing" into it, specifically by using the move right action to collide with the door. This method is not very effective and has only a 5% chance of opening the door. The state is not fully observable to the agent; instead, it receives a noisy observation, which will be explained later.

Factorization The dynamics of POMDPs can often be captured efficiently through factorization and graphical models, such as Dynamic Bayes networks (DBNs) [151, 152]. Throughout this work, the graphical model we rely on is specifically a two-stage DBN [151]. Such models represent random variables by their features (also referred to as factors) and provide the ability to capture independence between these features. We assume that the full set of state factors and their (discrete) set of possible values is known.

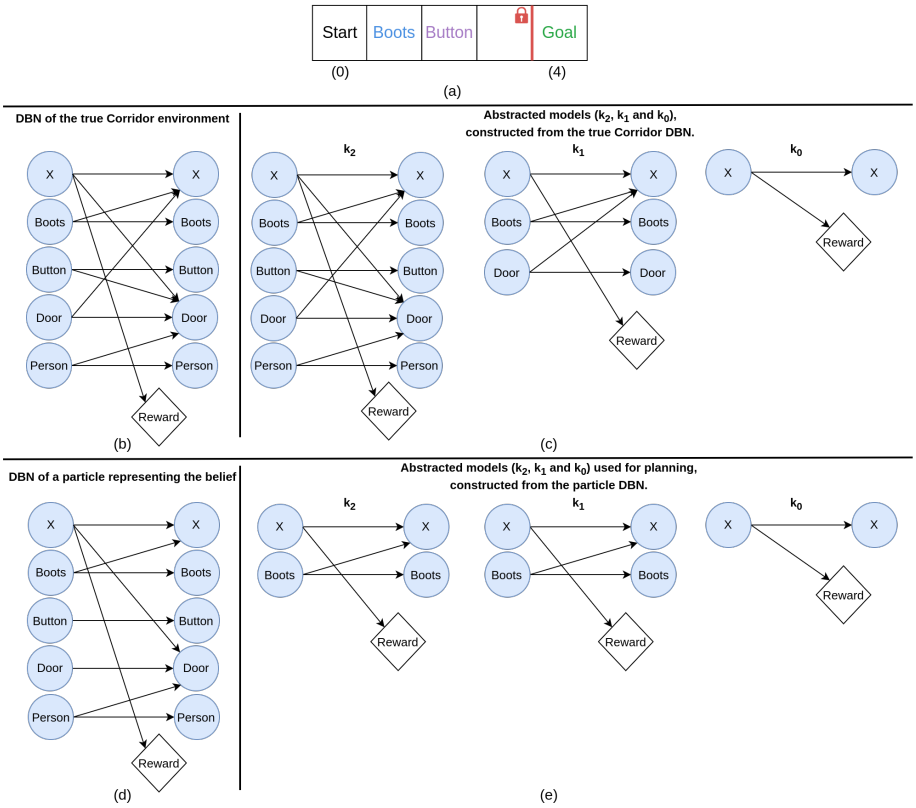


Figure 4.1: a) The Corridor domain. b) ground truth graph representing the dynamics of the Corridor domain for the action *right*. All the factors are partially observable, the agent gets an independent but noisy observation of each state factor. c) The abstract models for different levels of abstraction (denoted by k_0, k_1, k_2), constructed from the true Corridor DBN for the action *right*. d) We use particle filters to represent the belief, each particle contains a DBN of the corridor, this is an example of such a DBN, and e) the abstract models constructed from this example DBN.

A Bayes network (BN) is defined by a topology and conditional probability tables (CPTs). The topology $G \in \mathbb{G}$ describes the structure of the nodes in the graph, whether there is a dependency between pairs of nodes, where $\mathbb{G}$ is the set of all possible edge configurations. These directed edges define the *parents* $Pa(x'_i; G)$ of each node x'_i as the set of incoming nodes (where we typically drop the dependency on the topology G in the notation). The CPTs $\theta \in \Theta$ govern the conditional probability distribution of a node x'_i given the full set of nodes x : $p(x'_i|x) = p(x'_i|Pa(x'_i; G); \theta_i)$.[1] In discrete environments, for example, these typically are categorical distributions; one for each

[1]Here, x denotes the full set of nodes, and $Pa(x'_i; G) \subseteq x$ is the subset that directly influences x'_i.

parent value combination for each node. The *Dynamic* BN (DBN) restricts the space of graph topologies by allowing only directed edges from variables at one time step to variables at the next. This is convenient for (Markovian) dynamic systems, where random variables change over time. The Factored POMDP (F-POMDP) factorizes the state and observation space into nodes and describes the dynamics with a DBN for each action [150] (e.g., see Figure 4.1 as a DBN of the running example). In this work, we propose to learn abstract models that remove factors from the graph based on the reward node's dependency on them.

Bayesian inference over DBNs When the parameters of a model are not given, the Bayesian approach is to assume a prior (to compute posteriors) instead. The DBN is defined by — and thus a prior must describe a probability distribution over — its *topology* and CPTs. The prior over the topology assigns a (prior) probability to each graph structure: the probability that a next state and observation factor depends on a current state factor. The CPTs are categorical distributions and, hence, the Dirichlet distribution is a natural prior [153]. Dirichlet distributions are parameterized by a collection of conditional count tables (CCCT), with one conditional counts table (CCT) $\chi \in X$ for each unique set of parent values for each node.

Given initial counts specified by a prior and data from a categorical distribution, the posterior is again a Dirichlet (with new counts CCCT). In particular, given a topology G, prior CCCT and a new data point (x, x'), the Bayesian posterior is computed by incrementing the count $\chi_{x'_i, x[Pa(x'_i)]}$ of each node x'_i that is associated with its parent's values $x[Pa(x')]$. This incrementing operation on POMDP transitions will be used frequently and we denote updating counts $\chi \in X$ given a transition (s, a, s', o) with $\mathscr{U}$: $(X \times \mathbb{S} \times \mathbb{A} \times \mathbb{S} \times \mathbb{O}) \to X$. Note that there is no closed-form solution to the posterior over topologies.

4.2.2. FACTORED BA-POMDPS

If the state transitions were *not* hidden, one could simply maintain a set of the counts CCTs associated with each transition and over time converge to the true dynamics. This is the case under full observability (MDPs), and is called the Bayes-Adaptive MDP (BA-MDP) [21]. Unfortunately, this is not the case in partially observable environments[2], and hence, there is uncertainty over these counts χ.

The Factored Bayes-Adaptive POMDP (FBA-POMDP) [61] captures this uncertainty by using the POMDP formalism. In particular, this Bayes-adaptive model *is* a POMDP whose state space consists of both the state and the dynamics of the original POMDP [59]. Formally, the FBA-POMDP is a tuple $(\dot{\mathbb{S}}, \mathbb{A}, \mathbb{O}, \dot{\mathscr{D}}, \dot{\mathscr{R}}, \gamma, H)$, where $\dot{\mathbb{S}}$ is the augmented state space: $\dot{\mathbb{S}} = \mathbb{S} \times \mathbb{G} \times X$. I.e., each (hyper-)state $\dot{s} \in \dot{\mathbb{S}}$ contains a domain state s, a topology G, and a CCCT χ: $\dot{s} = \langle s, G, \chi \rangle$.[3] The action and observation spaces, the horizon, and the discount factor are taken directly from

[2] In an MDP we see the whole transition (s,a,s'), so we can update the count $\chi(s, a, s')$. In contrast, both s and s' are hidden in the partially observable case. So we have to update $\chi(s, a, s')$ based on our belief resulting from the action-observation history rather than the real transitions.

[3] In Section 4.3, we extend these hyperstates to incorporate an abstract model, comprising an abstract topology $\tilde{G}$ and a corresponding abstract set of counts $\tilde{\chi}$.

the original POMDP. Similarly, the reward function relies on the underlying system: $\dot{\mathscr{R}}(\dot{s}, a, \dot{s}') = \mathscr{R}(s, a, s')$. Note that, while this function is typically assumed known in BRL, we also conduct experiments in which this too is uncertain. Lastly, the dynamics $\dot{\mathscr{D}}$ dictate how augmented states transition:

$$\dot{\mathscr{D}} = p(s', o, G', \chi' | s, a, G, \chi) \tag{4.1}$$

$$= p(s', o | s, a; G, \chi) \mathbb{1}_G(G') \mathbb{1}_{\chi'}(\mathscr{U}(\chi, s, a, s', o)), \tag{4.2}$$

where $p(s', o | s, a; G, \chi)$ can be written as:

$$p(s', o | s, a; G, \chi) = p(s' | s, a; G, \chi) p(o | a, s'; G, \chi). \tag{4.3}$$

In (4.2) the term $p(s', o | s, a; G, \chi)$ shows that the model (G, χ) in state $\dot{s}$ determines the probabilities of the next state s' and observation o. The $\mathbb{1}_{\cdot}(\cdot)$ is the indicator function, and encodes that there is only one non-zero transition, namely where the next topology equals the previous $G' = G$ and the next counts are increments of the previous according to $\mathscr{U}$. Since the POMDP is fully specified, the original learning problem is cast to a planning problem *with known* dynamics, given a prior $p_{\mathscr{D}}$. Most importantly, the exact solution to this planning problem yields the optimal policy, in terms of exploration-exploitation, with respect to the prior [59]. Now we can apply our standard POMDP planning tools (e.g., particle filtering, planning) to FBA-POMDPs, as anytime solvers provide good approximations which converge to the exact solution in the limit of infinite compute [60, 61, 154].

4.2.3. SOLVING FBA-POMDPS

Unfortunately, FBA-POMDPs are very large, and naive applications of planning techniques will fail. Specifically, methods that require exact belief updates cannot be directly applied, as these updates are only feasible with a finite POMDP representation, which is impractical for large problems. This limitation makes it difficult to apply traditional POMDP planning methods without significant modifications. We give a high-level description of how solutions for FBA-POMDPs can be found, and refer to the original work [60, 61] for details. Just like in any other POMDP, a planning solution requires two components: belief tracking and action selection.[4]

Belief tracking in the FBA-POMDP The belief, the posterior over the current state, is a probability distribution over the POMDP state and its distribution $b \in \Delta(\mathbb{S} \times \mathbb{D})$ given the observed history $h_t = (a_0, o_1, \ldots, a_{t-1}, o_t)$. Unfortunately, the computation of the belief update is intractable in most problems. Thus, the belief is often approximated with particles instead [155]. A particle filter represents a distribution through *particles*, which in this case represent FBA-POMDP states, specifically each particle is a weighted FBA-POMDP state (w, s, G, χ) with (unnormalized) weight $w \in \mathbb{R}^+$. There are numerous sampling mechanisms for updating the belief given a

[4] The original POMCP [154] implementation combined these steps to some extent, but we separate them out.

new action-observation pair [155], but in this work we applied sequential importance sampling and re-invigorate the belief with a Metropolis-Hastings-within-Gibbs sampling procedure [61] when necessary (details in Appendix 4.8.1). The initial particles are sampled from a prior belief, which may encode varying degrees of domain knowledge. Across experiments, we consider different settings in which parts of the DBN structure or certain CPTs are assumed to be known, while others are learned from data. Details on the prior knowledge used in the experiments can be found in Sections 4.4 and 4.8.4.

Action selection in the FBA-POMDP Even with approximated belief updates, the belief space can be very large, especially in high-dimensional problems. Thus it is often infeasible to compute the action that maximizes the discounted return for every possible belief the agent could end up in. As a result, we extend the planner for FBA-POMDP [61] to pick actions *online* instead. Like any Monte Carlo Tree Search (MCTS) method, this method incrementally builds a look-ahead tree of simulated interactions in the (FBA-POMDP) environment. Each iterations samples a (hyper) state $(s, G, \chi) \sim b$ and simulates an interaction in the FBA-POMDP, where actions are picked according to Upper Confidence Bounds Applied to Trees (UCT) [156] to trade-off exploration and exploitation. For more details, see [157] for a survey on MCTS, [154] for MCTS in POMDPs, and [60, 61] for MCTS in Bayes-Adaptive POMDPs (BA-POMDPs).

4.2.4. STATE ABSTRACTION FOR (FACTORED) MDPS

State abstraction can be used to simplify complex problems by mapping the original state space to a smaller abstract state space [15]. This mapping is defined by an abstraction function ϕ, which maps each state s to an abstract state $\bar{s}$, where the bar notation indicates the abstract state space. Related to our abstraction approach is the notion of model-similarity abstraction. This comes in both an exact form, model-irrelevance abstraction [15], and an approximate form, approximate model-similarity abstraction [16]. These abstractions are also known as (approximate) stochastic bisimulation [75, 134].

In the exact case, states are grouped if and only if they yield identical rewards and transition functions in the abstract space under all actions. Formally, in a model-irrelevance abstraction, $\phi(s_1) = \phi(s_2)$ if and only if

$$\forall_{a \in A} \; R(s_1, a) = R(s_2, a), \tag{4.4}$$

$$\text{and } \forall_{\bar{s}' \in \bar{S}} \; T(\bar{s}' | s_1, a) = T(\bar{s}' | s_2, a), \tag{4.5}$$

where the transition probability to an abstract state $\bar{s}'$ is given by $T(\bar{s}' | s, a) := \sum_{s' \in \bar{s}'} T(s' | s, a)$. In the approximate case, these equalities are relaxed, requiring that the reward functions and transition probabilities to any abstract state $\bar{s}'$ differ by no more than a small parameter η. Exact abstractions preserve optimality, that is, a solution in the abstract MDP is an optimal solution in the original MDP. Approximate abstractions do not preserve optimality, but have a bounded loss in value based on the value of η [16].

In Factored MDPs (F-MDPs), Dearden and Boutilier [135] suggest leveraging the structure of the problem to remove factors which are less relevant or irrelevant. An abstract MDP can be constructed from the remaining factors. Such an abstraction can be viewed implicitly as a mapping function ϕ, where each configuration of the retained factors defines an abstract state, and all corresponding combinations of the removed factors are mapped to this same abstract state.

4.3. ABSTRACTION FOR FBA-POMCP

FBA-POMCP is able to learn and exploit the structure in POMDPs in a Bayesian way. However, it struggles when the number of factors grows large. On the one hand, the presence of many state factors itself slows down sampling, potentially leading to insufficient simulations to derive adequate actions. On the other hand, the prior belief over models with many state factors typically have low probability for models in which all factors have a small number of parents. As a consequence, the particle filter typically contains models that have at least a few factors with many parents. This leads to slow learning (low statistical strength) and possibly to exploration of factors with little or no effect on the rewards. These issues are problematic because our primary focus is on the task performance, rather than on learning the correct model itself.

A natural idea, therefore, is to explore in how far abstraction can address these two issues. While abstractions could lead to inaccurate models, in the regular (non-Bayes adaptive) planning case, it has been demonstrated that abstracting away factors with a weak influence can still improve the performance of online planning [148]. Further, without abstraction, the agent could waste time exploring the dynamics of factors with little impact on the performance, if they are falsely believed to be influential. By removing these factors, abstraction can reduce unnecessary exploration and focus on the factors relevant for performance. As such, we propose to explore the impact that abstraction of state factors can have when learning in partially observable settings, formalized as FBA-POMDPs.

Specifically, we propose to perform the Partially Observable Monte-Carlo Planning (POMCP) simulations with an abstracted FBA-POMDP model. We hypothesize that such abstraction can improve performance by 1) increasing the number of simulations that can be done thus improving performance in online planning, and 2) reducing unnecessary exploration of factors with little impact on the performance, and allowing to focus exploration on the relevant factors.

We cover the combination of abstraction with FBA-POMCP in four parts. First, we give a high-level overview of the abstraction method and how it is added to FBA-POMCP. Second, we define abstractions on different levels, denoted by $k_0, k_1, ...$, where k_0 represents the coarsest abstraction. We define these through *subsets of state factors* and show how to generate a subset of state factors from a graph structure for a particular level of abstraction. Third, we show how to use the subset of state factors to construct the abstract structure and counts. Finally, we provide theoretical support for the combination of FBA-POMCP with abstraction.

4.3.1. ADDING ABSTRACTION TO FBA-POMCP

We propose a method to enable the FBA-POMDP framework to benefit from abstraction. Specifically, we use abstract models for online planning with a variant of FBA-POMCP. To operationalize this, we cover the following steps:

1. We expand the representations to include abstract states.
2. While we do not exploit abstraction in the belief update, the abstracted belief state still needs to be updated. We cover the necessary modifications to the belief update process.
3. Finally, we explain how FBA-POMCP can use abstracted states.

Expanding the Belief Representation When initializing the weighted particle filter in FBA-POMCP, each particle is a hyper-state $\dot{s} = \langle s, G, \chi \rangle$ associated with a weight w. The hyper-state $\dot{s}$ contains a ground state s, a graph structure G, and a set of counts χ. For the initialization we require a probability distribution over the possible starting states, over the possible structures, and a probability distribution over the counts given a structure. For the running example, Figure 4.1d shows a possible structure of a hyper-state, in this case the factor *Door* is not believed to influence the x factor, and the factor *Button* is not believed to influence the *Door* factor. When combining FBA-POMCP with abstraction, we abstract G and χ and add the resulting abstracted structure $\bar{G}$ and counts $\bar{\chi}$ to each hyper-state $\dot{s}$. This leads to an abstract hyper-state: $\bar{s} = \langle \dot{s}, \bar{G}, \bar{\chi} \rangle$. The particle filter thus stores both the original hyper-state $\dot{s}$ and the abstracted structure $\bar{G}$ and counts $\bar{\chi}$. The construction of the abstract hyper-states is done during the initialization of the particle filter, as shown in Algorithm 4.

For our running example, Figure 4.1e illustrates the structure $\bar{G}$ for different levels of abstraction, corresponding to the original structure G in Figure 4.1d. In Algorithm 4, the function Abstract creates the abstract model from a hyper-state $\dot{s}$, as detailed in Algorithm 5. In the following sections we elaborate on the methods for selecting a subset based on the level of abstraction k and on creating the abstract $\bar{G}$ and $\bar{\chi}$.

The Belief Update Process To ensure consistency of the belief, we use the full model (G and χ) during the belief update [18, 61], as described in Sections 4.2.2 and 4.2.3. This means that the belief update process remains largely the same. The main difference is that we now track and update both the full and abstract models, as shown in Algorithm 6. As in Section 4.2.2, the graph structures of both the full and abstract models stay the same during the update. The counts χ are updated via $\mathcal{U}$. Since each state maps to exactly one abstract state, the abstract counts $\bar{\chi}$ can be updated through the update of the counts χ. Essentially, the (abstract) graphs remain unchanged, while the (abstract) counts are updated.

Using Abstracted States in FBA-POMCP The planning process exclusively uses abstract models. Specifically, in Algorithm 11, the abstract representation is used

Algorithm 4 Initialize Abstract Particle Filter

Input: $p_{\dot{s}_0}$: prior over initial state
n: number of desired particles
k: abstraction level
for $i \in 0, \ldots, n$ **do**
  $\langle s_i, G_i, \chi_i \rangle \sim p_{\dot{s}_0}$
  $\langle \bar{s}_i, \bar{G}_i, \bar{\chi}_i \rangle \leftarrow \text{Abstract}(k, \langle s_i, G_i, \chi_i \rangle)$
  $w_i \leftarrow \frac{1}{n}$
end for
return $\{\bar{s}_i, \bar{G}_i, \bar{\chi}_i, w_i\}_{i=0}^{n}$

Algorithm 5 Abstract

Input: k: abstraction level
$\dot{s} = \langle s, G, \chi \rangle$: hyper-state
$Q \leftarrow \text{GetSubsetK}(k, G)$
$\bar{G} \leftarrow G$
$\bar{\chi} \leftarrow \chi$
for $(q, a) \in Q \times A$ **do**
  $\bar{G}, \bar{\chi} \leftarrow q.\text{MarginalizeCounts}(\bar{G}, \bar{\chi}, Q, a)$
end for
for $x \in G - Q$ **do**
  $\bar{G}.\text{remove}(x)$ // Remove factor
  $\bar{\chi}.\text{remove}(x)$ // Remove factor
end for
return $\langle \dot{s}, \bar{G}, \bar{\chi} \rangle$

to perform the environment *Step* function, which is utilized during simulations and roll-outs in the look-ahead tree search. The *Step* function uses the hyper-state with the abstract model and an action to sample a next abstract state by iteratively sampling the factors. It is shown in Appendix 4.8.2.

The benefit of using an abstract model is that it speeds up planning since it contains fewer state factors, allowing for fasting sampling of the next state. When there is limited time for planning, this is one way in which abstraction can improve performance. It is important to note that the abstract models are constructed at the beginning of the agent's lifetime, when its belief is initialized. As a result, these abstract models are always available.

4.3.2. ABSTRACTION VIA SUBSETS OF STATE FACTORS

We introduce a method for performing abstraction in the Bayesian RL (BRL) context. Since we are interested in understanding how abstraction impacts the learning process, we base our approach on a relatively simple planning method for F-MDPs [135], which is easy to understand and analyze. We make three important adaptations: 1) we make it applicable to partially observable problems, 2) we extend it to the RL setting, where the abstraction is not only used for planning but also for learning, and 3) we incorporate the counts required in BRL. Our approach introduces a level of abstraction that determines the factors to include based on the graph structure.

We define different levels of abstraction based on their connection to the reward in the graph structure. Each abstraction level is defined by a set of state factors that is included in the model, observation factors are always kept in the model. After

Algorithm 6 SIS with Abstraction

Input: $\{\dot{s}, \bar{G}, \bar{\chi}, w\}_{i=0}^{n}$: current (weighted) filter
a, o: action and observation
for $i \in 0, \ldots, n$ **do**
  $s'_i \sim p(\cdot|s_i, a; G_i, \chi_i)$
  $w'_i \leftarrow w_i \times p(o|s'_i, a; G_i, \chi_i)$
  $\chi'_i \leftarrow \mathscr{U}(\chi_i, s_i, a, s'_i, o)$
  $\bar{\chi}'_i \leftarrow \bar{\mathscr{U}}(\bar{\chi}_i, s_i, a, s'_i, o)$
end for
// Normalize & re-sample
return $\{s'_i, G_i, \chi'_i, \bar{G}_i, \bar{\chi}'_i w'_i\}_{i=0}^{n}$

Algorithm 7 GetSubsetK

Input: k: abstraction level
G: Graph structure.
$Q \leftarrow$ GetMinimumSet() // k_0, the IR factors
if $k == 0$ **then**
  return Q
end if
$Q' \leftarrow Q$
for L = 1; L ≤ k; L++ **do**
  for $(q, a) \in Q \times A$ **do**
    $Q' \leftarrow Q' \cup G$.getNode(q, a).parents()
  end for
  $Q \leftarrow Q'$
end for
return Q

abstraction, this can lead to observation factors without parents, we explain how we deal with this in Section 4.3.3. We start building abstractions from the factors directly influencing the reward, the immediately relevant factors (IR). We first give a formal definition and then illustrate it with an example.

Definition 4.1. The set of immediately relevant factors (IR) contains only the factors $q \in G$ that directly influence the reward. Specifically, these are the factors that are parents of the reward, denoted as $Pa(\text{reward})$. The smallest subset of state factors k_0 is equal to IR, $k_0 = \text{IR}$. The set k_i is the smallest set such that the following holds:

1. $k_{i-1} \subseteq k_i$.
2. If $q \in k_{i-1}$ then $Pa(q) \in k_i$.

The set k_{inf} refers to the full model.

In the running example, we see an example of a structure in Figure 4.1d. In this problem, the agent only receives a reward when it is in the goal location, i.e., $x = 4$. This is reflected in the graph structure where the only parent factor of the reward is x. In this case, x is the only IR factor and the abstraction k_0 only contains x as shown in Figure 4.1e. To construct the subset of state factors for k_n, we add the parents of the factors in k_{n-1}. So to see which factors to include in k_1 in this example, we check in Figure 4.1d which factors are parents of x. In this case, that is only *Boots*. Finally, for k_2, no new factors are added since in the structure in Figure 4.1d the parents of *Boots* do not include any factors not yet in the set of k_1.

The procedure to get the subset of state factors for a given level of abstraction k and a particle (or hyper-state) $\dot{s}$ is shown in Algorithm 7. First, it initializes a set Q

with the set IR, retrieved with *GetMinimumSet*. For abstraction level k_0, this is what is returned immediately. For higher levels, it then builds the subset incrementally by adding the parents of the factors in Q. That is, to construct the subset of state factors k_n, it starts with k_{n-1} and then adds the parents of these factors.

When the reward function is known, the function GetMinimumSubset directly returns the set IR. When the reward function is unknown, the reward itself is also modeled as a state factor that takes the same value as the reward. Uncertainty about the reward function can then be incorporated in the belief, and hyper-states may end up with different graph structures for the reward. The IR can then be retrieved by finding the parents of the reward state factor. We demonstrate that our method can deal with uncertainty about the reward and IR in Section 4.4.4.

4.3.3. ABSTRACT MODEL CONSTRUCTION

After retrieving a subset of state factors, we construct the abstract model. Since the abstraction uses only a subset of the state factors, this involves removing factors, and therefore we need to decide how to treat factors that have missing parents as a result of this. To illustrate, when we abstract a full model such as the one in Figure 4.1d to create the abstract model on level k_0 (Figure 4.1e), we remove *Boots* and create a distribution for x with only x itself as the parent factor. The question then is how we can define a CCT that does not depend on 'Boots' from the original one that does.

For probability distributions in known models it is logical to resort to marginalization. However, in Section 4.3.3 we show that the problem is more deeply rooted: marginalization leaves us with a term that is difficult to specify since it depends not only on the values of its parents but also on the policy, and it can change over time. As such there is no fundamentally right approach to do this form of abstraction. Instead, novel ideas and approximate approaches are needed. We explore two initial ideas for this form of abstraction in Sections 4.3.3 and 4.3.3. In Section 4.3.3 we make an assumption on the abstraction and show that we can simply aggregate the counts in that case. In Section 4.3.3, we motivate using approximate abstraction and discuss potential issues that arise with the approximation.

MARGINALIZATION OF PROBABILITY DISTRIBUTIONS

Before considering the case of CCTs, we treat the case of probability distributions. For a probability distribution, given a factor X with a set of parents Parents(X), we can marginalize out a parent Y or a set of parents. For ease of notation, we show the marginalization for one parent, multiple parents can be removed by repeating this process:

$$P(x_i|\mathbf{z}) = \sum_{y_j} P(x_i, y_j|\mathbf{z}) \tag{4.6}$$

$$= \sum_{y_j} P(x_i \mid \mathbf{z}, y_j) P(y_j \mid \mathbf{z}), \tag{4.7}$$

where $\mathbf{z} = (z_1, z_2, \ldots, z_n)$ denotes the realization of Parents(X) $\setminus Y$, while x_i and y_j represent a specific realization of the factors X and Y. The term $P(y_j \mid \mathbf{z})$ acts as a

weight for the contribution of $P(x_i|\mathbf{z}, y_j)$ to the marginal distribution $P(x_i \mid \mathbf{z})$.

However, estimating $P(y_j \mid \mathbf{z})$ for a DBN is nontrivial. In the running example, consider the probability of moving from $x = 1$ to $x' = 2$ after taking the action *right*. The sampled model in Figure 4.1d only has the factors x and *Boots* as parents of x, and the abstract model k_0 (Figure 4.1e) does not include *Boots*. Following (4.7), we can obtain the marginal distribution for x by summing over the separate values of *Boots*:

$$P^{right}(x'|x) = \sum_{b \in Boots} P^{right}(x'|x, b)P(b|x). \tag{4.8}$$

However, while the probabilities $P^{right}(x'|x, b)$ are well defined (e.g., $P^{right}(x' = 2|1, b = \text{On}) = 0.95$), the value of $P(b|x)$ is not as clear. This probability represents the likelihood that the boots are on given a specific location x. However, it does not depend solely on x itself. For instance, we might know that the probability of the boots being on is 0 at the start of the episode, but this probability generally depends on the history of actions and observations. In general, we can make the following observation:

Observation 4.1. Accurately estimating $P(y|\mathbf{z})$ without additional information is generally not possible. This is because y can depend on other variables, including itself, and on the policy that can change over time.

The view of $P(y|\mathbf{z})$ as a weight in (4.7) is related to the concept of a weighting function in work on state abstraction [15, 135]. Theoretical work shows that, for some abstractions, a policy based on the abstract model (with any weighting function) can perform well in the real problem in planning [15, 16, 158] and in RL (see Chapter 3). For probability distributions, Dearden and Boutilier [135] use a sort of average of the probabilities but also remark this can lead to suboptimal solutions. The best way to approach estimating $P(y|\mathbf{z})$ for the optimal performance is still an open problem.

BELIEFS AND AGGREGATING COUNTS FOR EXACT ABSTRACTIONS

In the previous section we discussed the problem of dealing with conditional probability tables (CPTs) where parents are abstracted, leading to dependence on the policy and history for estimating $P(y|\mathbf{z})$ in (4.7). In this section, we begin by considering a simplified setting where the abstraction is assumed to be exact. This assumption allows us to sidestep the difficulty of estimating $P(y|\mathbf{z})$ and to introduce the CCCT more cleanly under idealized conditions. This assumption is made only in this subsection to clarify the conceptual difference between exact and inexact abstractions. The more realistic and interesting case is when the assumption does not hold, which we address in Section 4.3.3.

One situation where abstraction makes sense is when the model is overspecified; it contains links that are unnecessary. This is the case when the abstract model probabilistically behaves in the same way as the full model, which is what we assume in this subsection:

Assumption 4.1. The abstraction is exact. That is, let Z be the set of removed parents for a factor X, let $\mathbf{z} = (z_1, z_2, \ldots, z_n)$ denote a specific realization of the

remaining parents $\text{Parents}(X) \setminus Z$, and let $\mathbf{z}_1^{\text{removed}}$ and $\mathbf{z}_2^{\text{removed}}$ be two different realizations of the removed parents Z. Then, for all realizations x_i of X, we assume:

$$P(x_i|\mathbf{z}) = P(x_i|\mathbf{z}, \mathbf{z}_1^{\text{removed}}) \tag{4.9}$$
$$= P(x_i|\mathbf{z}, \mathbf{z}_2^{\text{removed}}). \tag{4.10}$$

This assumption implies that the links between the removed parents and the corresponding child node were obsolete. For instance, consider a change in the running example where the boots would have no effect on x, then this would mathematically mean that $P(x'|x, \text{Boots} = \text{On}) = P(x'|x, \text{Boots} = \text{Off})$. We observe:

Observation 4.2. Under Assumption 4.1, $P(y_j|\mathbf{z})$ has no influence. That is, since

$$P(x_i|\mathbf{z}) = \sum_{y_j} P(x_i|\mathbf{z}, y_j) P(y_j|\mathbf{z}) \quad (4.7) \tag{4.11}$$
$$= \forall_{y \in Y} : P(x_i|\mathbf{z}, y). \tag{4.12}$$

Thus, if boots had no influence, data collected with boots on and off can be used to estimate $P(x'|x)$. For example, consider the conditional counts $\chi(x'|x = 1, \text{Boots})$ in Table 4.1.

Observation 4.2 means that, to marginalize in the CCCT, we no longer have to be concerned about $P(y_j|\mathbf{z})$. Which means that in Table 4.1 we can aggregate the counts in the columns, formally:

$$\bar{\chi}(x_i|\mathbf{z}) = \sum_{y_j} \chi(x_i|\mathbf{z}, y_j). \tag{4.13}$$

For example, to determine the conditional counts in Table 4.1, we apply (4.13) to

Table 4.1: Initial conditional count table, for $x = 1$ and action *right*.

Boots	$x' = 1$	$x' = 2$
On	2	8
Off	6	4

Table 4.2: Count table after aggregation, for $x = 1$ and action *right*.

$x' = 1$	$x' = 2$
2+6 = 8	8+4 = 12

Table 4.3: Count table after aggregation and normalization, for $x = 1$ and action *right*.

$x' = 1$	$x' = 2$
$1/2 * 8 = 4$	$1/2 * 12 = 6$

construct the abstracted (or marginalized) counts $\bar{\chi}$ from the original counts χ. This is done for every action by aggregating the counts as follows:

$$\bar{\chi}^{right}(x'|x) = \sum_{b \in Boots} \chi^{right}(x'|x, b). \tag{4.14}$$

Writing out (4.14) we obtain the counts $\tilde{\chi}^{right}(x'|x=1)$ after aggregation:

$$\tilde{\chi}^{right}(x'=1|x=1) = \sum_{b \in Boots} \chi^{right}(x'=1|x=1,b) = 2+6=8, \tag{4.15}$$

$$\text{and } \tilde{\chi}^{right}(x'=2|x=1) = \sum_{b \in Boots} \chi^{right}(x'=2|x=1,b) = 8+4=12. \tag{4.16}$$

The resulting conditional counts are shown in Table 4.2. Note that now the resulting row has counts ($\{8,12\}$) which are higher than the individual previous rows for Boots *On* ($\{2,8\}$) and *Off* ($\{6,4\}$). This implies that after abstraction we are (relatively) more confident about these transitions than before. Under Assumption 4.1 this does not have a large influence when the prior is close to the true distribution, as in that case these estimates should be close together. However, this could be different when the abstraction is not exact or if the prior is not close to the true distribution.

4

AGGREGATING COUNTS FOR APPROXIMATE ABSTRACTIONS

Previously, we assumed that the abstraction was exact. Of course, this may not always be the case, or it may not be necessary to make this assumption. There exist scenarios where one could argue for the use of *approximate* abstractions, as discussed in [16, 135] and Chapter 3. For example, in cases where a parent only has a small influence, abstracting these parents away can lead to faster learning (see Chapter 3). Additionally, reducing the size of the model through abstraction can enhance performance by facilitating faster planning [148].

However, when the abstraction is not exact, Observation 4.2 no longer holds. Specifically, with an approximate abstraction, $P(x|\mathbf{z},y)$ generally varies for different instantiations of y. Consequently, $P(y|\mathbf{z})$ does influence the result. In this case, abstracting a candidate model could result in a probability distribution that deviates significantly from the behavior of the candidate model.

As an example, consider again the running example where with *Boots* = *On* we have a probability of moving of 95% and with *Boots* = *Off* only 30%. Table 4.1 shows are initial estimates where *Boots* is still included, counts of $\{2,8\}$ and $\{6,4\}$ for *Boots* = *On* and *Boots* = *Off*, respectively. These are reasonably accurate with, if we translate the counts to probabilities, an expected 80% and 40% chance of moving, respectively. However, when *Boots* is removed we see in Table 4.2 this leads to counts of $\{8,12\}$, or an expected probability of moving of 60%. With *Boots* being removed from the model the agent is highly likely to be in a state with *Boots* = *Off*, and thus this leads to an overestimation of the probability of moving for the agent.

As alluded to in the previous section, the abstraction also increases the confidence in the resulting counts. When the abstraction is not exact, we could say that the increased confidence in the resulting counts is not warranted, there is overconfidence. This overconfidence can slow down learning since it will take more experience to change the belief. For example, the change in Table 4.1 of adding an extra observation to the row with *Boots* = *Off* has a relatively larger effect than adding one extra observation after aggregation in Table 4.2.

This means that the proposed aggregation in (4.13) does not work as well when Assumption 4.1 does not hold, since it can lead to incorrect estimations with a higher

confidence. As such, we want to adapt the aggregation method. **It is still an open question what the best way to aggregate when using approximate abstraction.**

We propose a way to reduce the overconfidence in the resulting dynamics through a normalization scheme, which should lead to quicker learning in cases where the prior and abstraction are biased. Let $Z = Y_1, Y_2, \ldots, Y_n$ denote the set of removed parents. To normalize, we multiply each entry by

$$\frac{1}{\prod_{Y \in Z} |\text{dom}(Y)|}, \tag{4.17}$$

where $|\text{dom}(Y)|$ represents the number of values that the parent Y can take. For example, in the case of the position x from the running example, we have $|\text{dom}(x)| = 5$.

Using $\tilde{\chi}$ to represent normalized counts, applying the normalization factor (4.17) to (4.13) results in:

$$\tilde{\chi}(x_i|\mathbf{z}) = \frac{1}{\prod_{Y \in Z} |\text{dom}(Y)|} \sum_{(y_1, y_2, \ldots, y_n) \in Y_1 \times Y_2 \times \cdots \times Y_n} \chi(x_i|\mathbf{z}, y_1, y_2, \ldots, y_n). \tag{4.18}$$

Intuitively, the proposed normalization scheme reduces the counts proportionally to the amount of rows that is removed during aggregation. Applying (4.18) to the example where we remove *Boots* this leads to:

$$\tilde{\chi}^{right}(x' = 1|x = 1) = \frac{1}{|\text{dom}(Boots)|} \sum_{b \in Boots} \chi^{right}(x' = 1|x = 1, b) = \frac{1}{2}(2 + 6) = 4, \tag{4.19}$$

$$\text{and } \tilde{\chi}^{right}(x' = 2|x = 1) = \frac{1}{|\text{dom}(Boots)|} \sum_{b \in Boots} \chi^{right}(x' = 2|x = 1, b) = \frac{1}{2}(8 + 4) = 6, \tag{4.20}$$

also shown in Table 4.3.

By applying the normalization of (4.17) the amount of counts in the table after aggregation is equal to the average amount of counts in the initial prior. For example, in Table 4.1 the counts in the rows both sum up to 10, and the counts after the aggregation and normalization also sum up to 10 (Table reftable:testmarg).

The proposed normalization provides a robustness against mistakes in the prior and approximate abstraction, by lowering the impact that the prior has on the learning. In our experiments, we investigate this normalization scheme, showing that this can significantly speed up learning.

THE OBSERVATION SPACE FOR ABSTRACT MODELS

Since we remove state factors from the model in the abstraction, a natural question is how we deal with the observation factors. We can consider two cases, 1) where a part of the parents is removed, and 2) when all the parents are removed.

In the first case, we simply perform the aggregation in the same way as for the state factors. It is the second state that poses a problem, as it leaves us with no

parents for the observation factor. Since in this case the observation would provide no actual information about the underlying state, we enhance the observation space of the abstract model by including an observation "not observed" for all observation factors. For observation factors where all parents are removed the observation function simply returns "not observed".

4.3.4. THEORETICAL SUPPORT

Here we will show how the combination of FBA-POMCP with the abstraction method can lead to near-optimal performance when the abstraction is good. We first show that transforming the original FBA-POMDP with the abstraction method results in another FBA-POMDP. Because of this, the theoretical guarantees of FBA-POMCP apply to the abstracted problem. Then we give a definition for the quality of the abstraction that gives a guarantee on the performance in the original FBA-POMDP. Together this shows that abstractions leads to near-optimal solutions with FBA-POMCP, when a good abstraction is used.

First, we note that the abstraction results in another FBA-POMDP:

Lemma 4.1. The result of applying the abstraction method, as described in Algorithm 5 and Section 4.3.3, to the original FBA-POMDP results in another FBA-POMDP, the abstract FBA-POMDP.

In Appendix 4.8.3, we present a constructive proof. In essence, abstraction reduces the state space and marginalization produces a dynamics function for the resulting state space. Lemma 4.1 implies that we can use POMCP to find a near-optimal solution with respect to the belief in the abstract FBA-POMDP, due to the following result:

Theorem 4.1 (Katt et al., 2019)**.** Given a belief $b(s, G, \chi)$, FBA-POMCP converges to an ϵ-optimal value function of a FBA-POMDP: $V(b,a) \xrightarrow{p} V^*(b,a) - \epsilon$.

The bias of the value function, ϵ, can be made arbitrarily small by increasing the maximum search depth. If there is no limit on the search depth, the bias ϵ is $O(\frac{\log(n)}{n})$ in the limit of the number of simulations n starting from the belief b [154].

The question that remains is, how good is the solution for the abstract FBA-POMDP in the original FBA-POMDP? In general, the quality of a solution when using abstraction depends on the type and quality of the abstraction [16, 135]. We define the quality η of the abstraction as the upper bound of the difference between optimal value function of the original FBA-POMDP and the value function of the original FBA-POMDP under a ϵ-optimal policy for the abstracted FBA-POMDP:

Definition 4.2. An abstraction has a quality η, s.t. every ϵ-optimal solution π of the abstract FBA-POMDP applied to the original FBA-POMDP has suboptimality bounded by $\epsilon + \eta$:

$$\forall (b,a) \in \mathbb{B} \times \mathbb{A} : \; |V^*(b,a) - V^{\pi}(b,a)| \leq \epsilon + \eta. \tag{4.21}$$

This definition is based on existing suboptimality bounds for abstract models such as the approximate model-similarity abstraction [16, 135], discussed in Section 4.2.4. Informative bounds can be derived, for instance, by assessing how well the abstract model can approximate the original model, see [135] and Chapter 3. In the case of approximate model-similarity abstraction, the parameter η is small when the grouped states have similar transition and reward functions. In Chapter 3, we show that, in the context of learning with abstraction in MDPs, this type of abstraction can yield such bounds. Analogously, in POMDPs, similar results could be expected to hold if the dynamics of the abstract model closely resemble those of the true model. Extending the findings in the fully observable to the partially observable setting could be feasible by incorporating learning of the observation function and applying a simulation lemma for POMDPs [159].

When combined with Theorem 4.1, definition 4.2 implies that combining FBA-POMCP with abstraction leads to a near-optimal solution for the original FBA-POMDP, particularly when the abstraction effectively captures the dynamics of the original problem (i.e., when η is small):

Corollary 4.1. Given a belief $b(s, G, \chi)$ and an abstraction, assuming there exists an η for which (4.21) holds, FBA-POMCP combined with abstraction converges to an $\epsilon+\eta$-optimal value function of the original FBA-POMDP: $V^{\pi}(b,a) \xrightarrow{p} V^{*}(b,a) - (\epsilon+\eta)$.

Proof. By Lemma 4.1 the problem after abstraction is still an FBA-POMDP. By Theorem 4.1, we can use FBA-POMCP to get a policy π within ϵ of the optimal solution of this abstract FBA-POMDP. Then, by Definition 4.2 this leads to a solution within $\epsilon+\eta$ of the optimal solution of the original FBA-POMDP. □

This result shows that we can reach near-optimal performance in the original problem with good abstractions, but that better performance can be achieved in theory without abstraction when $\eta > 0$. However, in practice abstraction could perform better through faster simulations, and it could get to a good performance more quickly through greater statistical strength due to aggregation.

4.4. EXPERIMENTS

We aim to investigate three questions; 1) does abstraction lead to faster simulations and enable scaling to more complex problems, 2) does abstraction lead to faster learning by reducing unnecessary exploration, and 3) does abstraction provide these benefits when the reward function is not known? We empirically evaluated our approach on four domains to answer these questions. The first domain is the Corridor problem from the running example, a simple problem where the trade-off of using abstraction is shown. The second domain is Cracky Pavement Gridworld where we show the advantage of abstraction in a very large problem, with a state space of up to size $|\mathbb{S}| = 6 \times 10^{25}$. In addition, in this domain we show the effectiveness of the proposed normalization step for abstraction, (4.17), in the k_0 model. The third domain is an adjusted version of Collision Avoidance [61, 160], made more complex to allow for more abstraction. The final domain is Room Configuration, where the

agent must learn the reward function. For further details on the experimental setup and the domains we refer to Appendix 4.8.4.

4.4.1. CORRIDOR

The Corridor domain is the domain described in the running example in Section 4.2.1, where there are 8 different *Persons* that can be present, each with a small probability of opening the door. The agent is uncertain about the transition dynamics of the x-position, for the actions left and right, and the door. We evaluate the abstractions k_0, k_1, and the full model k_{inf}. The abstraction k_0 only includes the factor x. Abstraction k_1 includes x and, depending on the structures included in the belief, can also include the factors *Boots*, *Button* and *Door*.

Results We test the effectiveness of two levels of abstraction under a fixed number of simulations. Figure 4.2a shows the simple moving average of the return per episode, and the shaded areas show the 95% confidence interval. Figures 4.2b, 4.2c, and 4.2d illustrate the behavior of the different models across episodes. Specifically, these figures display how often the agent put on the boots, how often the door was opened, and through which means the door was opened. For example, in

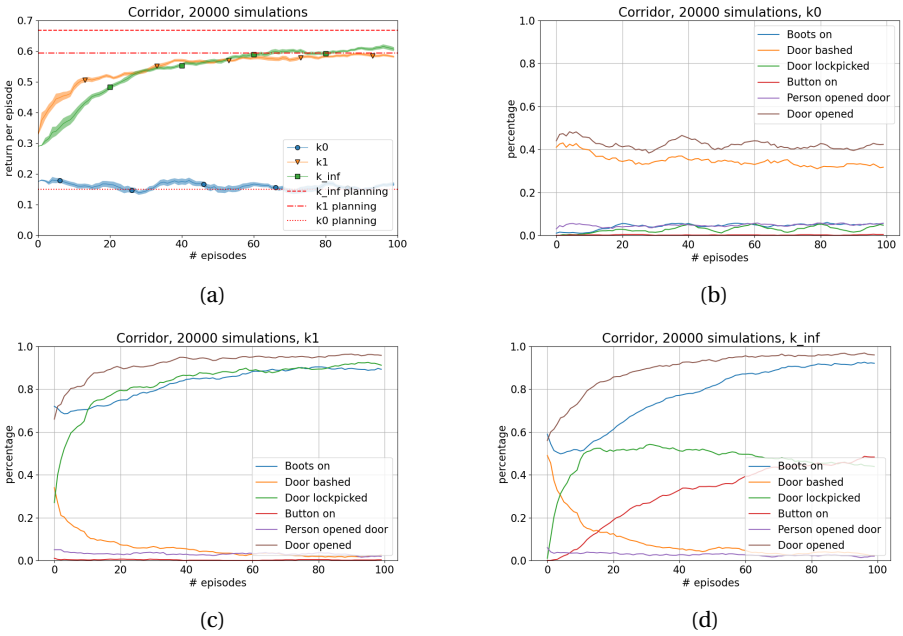


Figure 4.2: a) Performance in the Corridor domain. The learning behavior in the Corridor domain is shown for different models: b) k_0, c) k_1, and d) k_{inf}. The y-axis shows the percentage of runs in which a certain behavior or occurrence happened during each episode.

Figure 4.2d, it can be seen that in the first episode, the agent rarely lock picked the door (close to 0%), whereas after 20 episodes, the agent lock picked the door in more than 50% of the runs.

First, there is a large difference in the performances between k_0 and the other two models. This difference occurs because the full model and model k_1 can learn to open the door through more effective means than simply bashing against it, while k_0 cannot. This is due to the fact that the model k_0 only keeps the x factor and does not include *Door*, *Boots*, and *Button*. Consequently, it does not recognize that the actions to open the door, push the button, and put on the boots have any effect. In addition, since these actions have no direct effect on the x-position, k_0 is unable to learn interactions with the environment beyond moving left and right. It ultimately learns a strategy of just moving to the right. This strategy can still lead to reaching the goal state, as there is a chance that the door opens when the agent bashes into it by moving right, or when the person opens the door. This can be observed in Figure 4.2b, which shows that most of the time the agent successfully opened the door by bashing into it. While this shows the agent never learns to open the door by pushing the button, it does occasionally open the door through the *lock pick* action. This can happen near the end of the episode, with just one step remaining when the agent is at location $x = 3$ and the door is still closed. In such situations, the values for the different actions in the tree search will all be zero. Since the action is then randomly chosen among those with the same value, this can lead to selecting the *lock pick* action.

The k_1 model can effectively learn to interact with a part of the environment because it keeps not only the x factor but also the factors that influence x. This means that if the believed model is the correct model, the k_1 model also contains *Boots* and *Door*. Consequently, k_1 learns to *put on boots* and to *lock pick* the door, leading to a much better performance than k_0. Due to greater statistical strength from aggregation, k_1 learns to use lock pick more quickly than the full model.

Technically, it would be possible for k_1 to also open the door by *pushing the button* if the *Button* factor is also believed to influence x. However, as shown in Figure 4.2c, this did not occur frequently in the experiments. Instead, k_1 rapidly learns to *put on boots* and *lock pick*.

When comparing the performance of k_1 with the full model k_{inf}, we can distinguish three phases. Initially, k_1 learns more quickly than k_{inf} because the full model k_{inf} takes longer to learn to *lock pick*. Then, k_{inf} catches up as it learns to *lock pick*. Finally, k_{inf} starts to surpass k_1 by learning to open the door through pushing the button, as is visible in Figure 4.2d.

This experiment shows that when the abstraction is not exact, it can initially still lead to better performance because of greater statistical strength due to aggregation. Since in this domain the abstract models cannot learn the optimal behavior, they eventually get outperformed by the full model.

4.4.2. CRACKY PAVEMENT GRIDWORLD

The Cracky Pavement Gridworld is a grid world as shown in Figure 4.3a. The DBN of the domain is shown in Figure 4.3b. The state space is factored into the x and

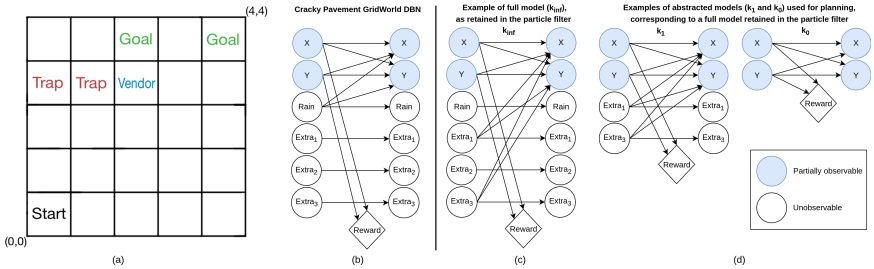


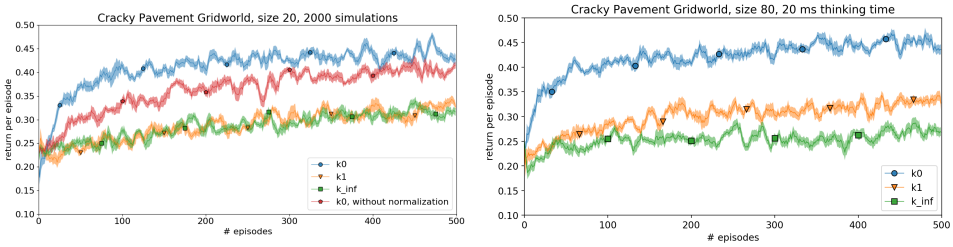
Figure 4.3: a) The Cracky Pavement Gridworld, b) ground truth graph representing the dynamics of the Cracky Pavement Gridworld problem with 3 extra binary factors, c) example of a full model in the particle filter, d) examples of two abstract models.

4

y locations, *Rain*, and 20 or 80 extra binary factors. Only x, y, and *Rain* influence the movement of the agent, though the agent must infer this. Movement success depends on tile type and rain conditions: movement is difficult on Trap tiles, and on the Vendor tile when it is dry, due to the presence of a vendor. When it rains, the vendor leaves and the tile becomes easier to traverse. The agent observes x and y noisily and does not know which factors affect transitions. We evaluate the abstractions k_0, k_1, and the full model k_{inf}. The abstraction k_0 includes only the factors x and y since they are the ones that directly influence the reward (Figure 4.3b). Abstraction k_1 also includes the parents of x and y.

Results First, we examine the results for one particular setting, with 80 extra binary factors and 2000 simulations. Figure 4.4a shows these results, where the lines represent the simple moving average of the return per episode, and the shaded regions indicate the 95% confidence interval. Both k_1 and the full model k_{inf} struggle to learn a good policy, while k_0 outperforms both.

The primary difference between k_0 and the other two models is that k_0 retains



(a) (b)

Figure 4.4: Performance in the Cracky Pavement Gridworld domain, a) with a fixed thinking time, b) with a fixed number of simulations.

Table 4.4: Average number of simulations in the Cracky Pavement Gridworld.

	Size 20			Size 80		
Time	k_0	k_1	k_{inf}	k_0	k_1	k_{inf}
5ms	634	550	272	521	429	149
10ms	1324	1263	573	1097	871	240
15ms	1978	1658	660	2044	1726	328
20ms	2571	2145	803	2433	2360	440

tb

Table 4.5: Average return over the first 500 episodes in the Cracky Pavement Gridworld.

	Size 20			Size 80		
Time	k_0	k_1	k_{inf}	k_0	k_1	k_{inf}
5ms	0.33	0.26	0.23	0.31	0.25	0.18
10ms	0.39	0.29	0.26	0.37	0.28	0.21
15ms	0.41	0.31	0.27	0.41	0.30	0.23
20ms	0.42	0.30	0.28	0.42	0.30	0.25

only the x and y factor, while k_1 includes any factor it believes influences x or y, such as *Rain* or parts of the additional binary factors, and k_{inf} retains all factors as it does not abstract. Although k_0 sacrifices the ability to account for *Rain*, it performs better because it simplifies learning about the trap states by only considering x and y. This leads to greater statistical strength as it is much easier to learn $P(x'|x,y)$ than $P(x'|x,y,\text{rain},\text{and numerous binary factors})$.

While k_0 cannot distinguish between rain and no rain and therefore does not learn when the vendor is on the tile, it can learn to navigate around this tile. Although this is generally not optimal, it is optimal when rain and the vendor are not considered. The full model k_{inf} and k_1 can eventually outperform k_0 by learning that it is better to cross the vendor tile when it is raining, as shown in Appendix 4.8.5. However, this takes a considerable amount of time, and during the earlier episodes, k_0 performs much better earlier.

These results shows that the greater statistical strength obtained by removing information, like *Rain* and extra binary factors, can result in a significant increase in performance. Additionally, Figure 4.4a compares the performance of k_0 with and without the proposed normalization step for abstraction (4.18), demonstrating that the normalization step can significantly improve learning performance.

In Figure 4.4b, where we compare the performance with a fixed amount of thinking time instead of a fixed number of simulations, we see that k_1 outperforms the full model k_{inf}. The main difference between k_1 and k_{inf} is that k_1 abstracts away all the factors that, given the graph topology, are not directly or indirectly relevant for

the reward. This means that the k_1 model is generally smaller than the k_{inf} model, leading to faster simulations. As shown in Table 4.4, k_1 performs an average of 2360 simulations with 20ms of thinking time, while the full model only reaches 440 simulations. This increase in simulation speed results in the improved performance shown in Figure 4.4b.

Table 4.5 shows that an increase in thinking time generally increases performance, most notably when increasing from 5ms to 10ms, with diminishing returns for further increases. The differences between k_1 and k_{inf} are also most pronounced at lower thinking times, especially with 80 additional binary factors, where the abstraction provides the most benefit. These findings demonstrate that augmenting FBA-POMCP with abstraction can increase performance through computational efficiency.

Overall, these results show that abstraction can be beneficial in multiple ways. Increasing the simulation speed leads to better performance, and simplifying the problem leads to faster learning due to greater statistical strength. The improvement in simulation speed is most pronounced when many factors are abstracted away, maximizing the difference in simulation speed between the models with and without abstraction.

4.4.3. COLLISION AVOIDANCE

In the Collision Avoidance domain, the agent flies from one side to the other in a 10 (width) x 5 grid. The episode ends when the agent reaches the last column, where it has to avoid colliding with a moving obstacle. This obstacle has a 20% chance to stay stationary and otherwise randomly moves either up or down. The agent can decide to move up, down, or stay level. We increased the complexity of the original Collision Avoidance [61, 160] by adding additional factors: *Speed* (slow,

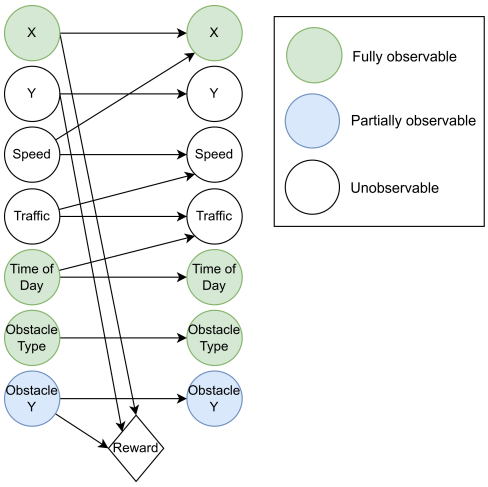


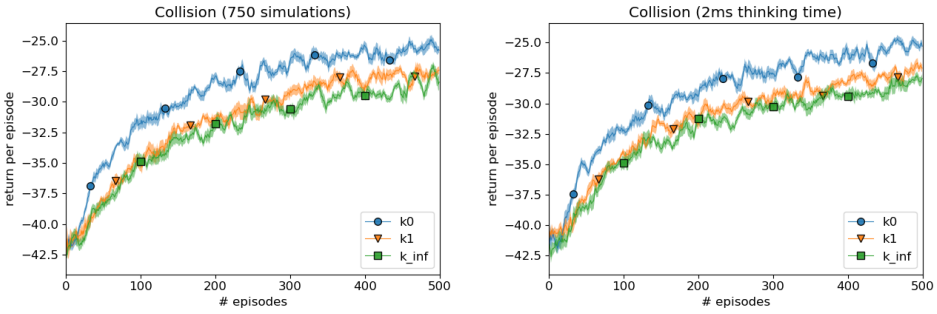
Figure 4.5: Ground truth graph representing the dynamics of the Collision avoidance problem.

fast), *Traffic* (low or high amount of traffic), *Time of Day* (day, night), and *Obstacle Type* (3 types, e.g., helicopter, plane). These influence the agent's forward movement and the obstacle's behavior, Figure 4.5 shows the resulting dynamics. Only the obstacle's transition function is uncertain and the agent receives noisy observations of the obstacle. We evaluate the abstractions k_0, k_1, and k_2, which is equivalent to the full model k_{inf}. The abstraction k_0 includes the factors x, y, and *Obstacle Y*. The abstraction k_1 additionally includes the factors *Speed* and *Obstacle Type*, if *Obstacle Type* has a connection to *Obstacle Y*.

Results We again test the effectiveness of two levels of abstraction under a fixed number of simulations. One of the benefits of abstraction is a smaller and (therefore) faster model. Another benefit is an increase in statistical strength through the removal of factors. The lines in Figure 4.6 show the simple moving average of the return per episode, and the shaded areas show the 95% confidence interval.

We see that abstraction k_0 learns significantly faster than both k_1 and k_{inf}, both with a fixed number of simulations and with a fixed thinking time. The abstraction k_0 allows for faster learning of the transition function of *Obstacle Y* since it removes the (possible) parents *Speed* and *Obstacle Type*. When *Speed* and *Obstacle Type* are parents of *Obstacle Y* in the sampled graph structure, k_1 retains these connections. As a result, k_1 learns the transition function of *Obstacle Y* at a rate comparable to k_{inf}.

In k_1, *Time of Day* and *Traffic* are removed. However, since their transition functions, along with that of *Speed*, are considered known, this does lead to greater statistical strength in learning *Speed*. The impact of *Time of Day* and *Traffic* is also less pronounced, as they only influence x indirectly through *Speed*. Nevertheless, k_1 tends to perform slightly better than k_{inf}. This is not due to faster simulations; in this domain, any speed up is minimal, and similar results are observed even with a fixed number of simulations. One potential advantage of removing *Time of Day* and *Traffic* is that it reduces the branching factor in the tree, as there will be no separate



(a) (b)

Figure 4.6: Performance in the Collision Avoidance domain, a) with a fixed thinking time, b) with a fixed number of simulations.

observations for these factors. This focuses the tree search by eliminating the need to consider these factors, albeit at a slight cost to model accuracy.

4.4.4. ROOM CONFIGURATION

In the Room Configuration domain, the task of the agent is to configure items in a 4 (width) x 3 grid to satisfy a teacher's preferences. Each tile contains a configurable item with two settings. The teacher is concerned only with the configuration of three specific items, but the agent does not initially know exactly which ones. The reward is modeled via a fully observable *Happy* factor, which takes on the same value as the reward. The agent must learn which configuration factors affect *Happy*. Small rewards or penalties are given when changing item settings, and a large reward is received once all desired configurations are set. The agent receives noisy observations of its location and has imperfect knowledge of movement success. We evaluate the abstraction k_0 and the full model k_{inf}.

Results We test learning of the reward function and the advantage of abstraction when only factors that (the agent believes) are irrelevant to the reward are abstracted away. The lines in Figure 4.8 show the moving average of the return per episode, and the shaded areas show the 95% confidence interval. We can see that the agent learns how to perform well in 10 episodes, after which the performance remains the same.

The abstraction k_0 performs more simulations (455) than the full model k_{inf} (365) within a fixed amount of time (10ms). This computational advantage allows k_0 to outperform the full model, as shown in Figure 4.8b. Figure 4.8a further demonstrates that there is no performance difference between the two models when the number of simulations is fixed. These results show that the agent can quickly learn the structure of the reward and that abstraction can increase performance through its

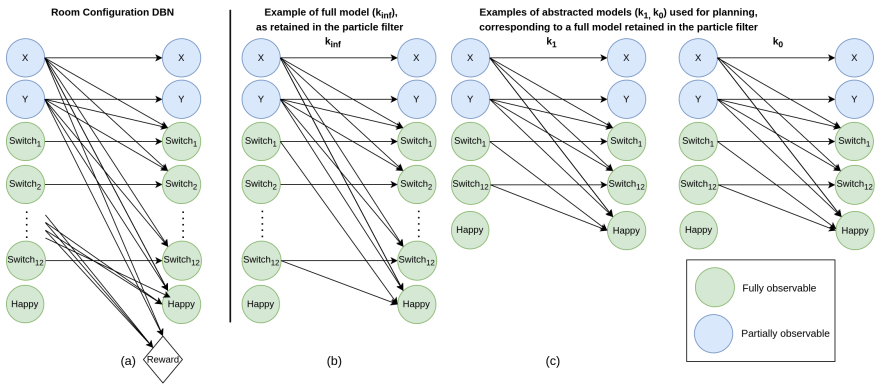
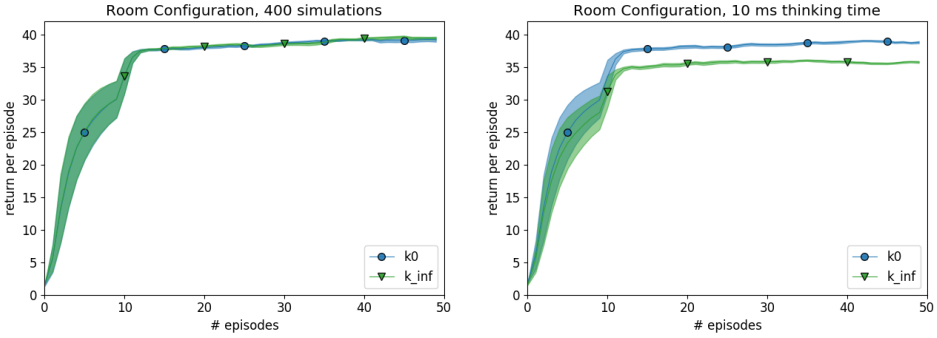


Figure 4.7: a) Ground truth graph representing the dynamics of the Room Configuration domain for the *switch* action, b) example of a full model in the particle filter, c) examples of two abstract models.



(a) (b)

Figure 4.8: Performance in the Room Configuration domain, a) with a fixed thinking time, b) with a fixed number of simulations.

increased simulation speed.

4.5. RELATED WORK

The FBA-POMDP is a factored version of the tabular BA-POMDP [59]. The infinite-POMDP approach [161] is a non-parametric Bayesian approach. This approach requires no knowledge of the state space. However, it assumes a hierarchical Dirichlet Process as prior, for which providing an informative prior can be difficult. BRL in continuous POMDPs typically makes Gaussian assumptions, such as in Dallaire *et al.* [162], where Gaussian processes are the model of choice. To extend our approach to continuous POMDPs, DBNs that work with continuous variables [163] could be investigated. Alternatively, constructing different levels of abstraction in a continuous domain could done through varying levels of discretization.

The BA-MDP [21], and the respective solution methods [164–166], are the fully observable counter-part to the (F)BA-POMDP. In this setting (BRL for MDPs), work has been done on exploring applications of Deep RL [167, 168]. This line of work solves the (easier) fully observable problem, and how to extend these methods to POMDPs is unclear.

Other approaches make use of recurrent networks for dealing with partial observability to generalize deep RL to POMDPs [169–172]. However, these networks are general-purpose, requiring many samples. This realization has motivated RL-specific architectures designed to capture history efficiently [11, 88]. Deep variational methods are another approach that can be efficient [85, 86]. However, none of these methods allow for encoding prior knowledge or tackling the exploration problem, to which the FBA-POMDP framework provides an elegant solution.

In the last two decades, there have additionally been many different approaches to (not deep or Bayes) learning in POMDPs, e.g., McCallum [81], Azizzadenesheli,

Lazaric, and Anandkumar [109], Shani, Brafman, and Shimony [173], Liu and Zheng [174], and Bennett and Kallus [175]. However, these also do not allow for encoding prior knowledge or tackling the exploration problem.

Planning with abstractions has been studied before, mostly in the context of MDPs. The method by Dearden and Boutilier [135], that our approach is based on, applied abstraction to factored problems. We extend this work to the partially observable BRL and planning setting and introduce a mechanism to automatically create multiple levels of abstraction based on the structure of the problem. Another line of work has focused on building abstractions during the tree search [176, 177]. Hostetler, Fern, and Dietterich [176] also provide results for doing the tree search using an abstraction function. However, this means the simulations themselves need to be done using the full model. Chitnis *et al.* [149] instead first learn an abstraction, which they then use for planning. Rather than being given an abstraction, their goal is to learn one. We investigate the effects of an abstraction method on learning efficiency and performance. He, Suau de Castro, and Oliehoek [148] do planning in the partially observable setting and construct an abstract model before the tree search. The key difference with our approach is that we do not assume access to a simulator of the real environment. Instead, we *learn* abstract dynamics while acting in the real-world.

Our work is closely related to posterior sampling approaches in causal RL, particularly the work by Mutti *et al.* [178]. Their method maintains a prior distribution over potential factorizations of a F-MDP and performs posterior sampling reinforcement learning accordingly. Our approach differs significantly in several aspects. First, we extend the methodology from fully observable scenarios (F-MDP) to partially observable ones (F-POMDP). Their approach explicitly notes that exact solutions for the sampled F-MDPs are required, a condition generally computationally intractable and even more prohibitive in the context of F-POMDPs. In contrast, our method circumvents this exact-solution requirement by employing an anytime planning algorithm (FBA-POMCP) within the "planning as learning" paradigm.

4.6. DISCUSSION

A limitation of the FBA-POMCP method is that it does not plan beyond the current episode. By not planning for future episodes, it does not consider the value that knowledge obtained in the current episode can have for future episodes. Approaches that quantify information gain [179–181] and those that plan for long horizons [182] could address this limitation, and future work could explore combining these strategies.

A difficulty of applying abstractions within FBA-POMCP lies in marginalizing the belief. As discussed in Section 4.3.3, when aggregating counts for approximate abstractions, it is unclear what the best solution is, since this is generally unknown beforehand and can depend on the policy. Even for a fixed policy, marginalization may lead to a belief that misrepresents the true (abstract) dynamics. For example, in the corridor problem, marginalizing away the *Button* factor leads to misrepresented

abstract dynamics because marginalizing it away combines the situations where the button is in the non-pushed state with those where it is in the pushed state, implicitly resulting in a belief that at each step there is a probability that the button is in the pushed state. This is a misrepresentation since the button starts in the non-pushed state and, since it is marginalized away, the agent will not learn to push the button, so it will always be in the non-pushed state. Such situations can make it more difficult for the agent to learn. An idea could be to use knowledge about the starting position and the abstracted model in the marginalization. For instance, if the *Button* factor is removed from the model, we could keep the counts for when it is in the non-pushed state and ignore the counts for the pushed state since it is always in the non-pushed state at the beginning of an episode.

In the Corridor experiment, we aimed to make the results more interpretable by showing how the agent's behavior changes during learning, across the different models. This helped clarify the sources of the observed differences in return per episode, as well as the adaptations the agent makes over time. The behavior of the smaller abstract models is easier to understand than that of the full model, and this improved interpretability offers an additional motivation for using abstraction and could be an interesting perspective for future research.

The Corridor experiment also shows that abstraction does not always help, and can in fact be detrimental. When crucial information is removed, it can make learning impossible. In general, the usefulness of abstraction depends on how much value (or information) is lost. If the loss in value is too great, more simulations will not help to achieve a better performance. On the other hand, if there is no loss in value or the loss in value is relatively small, abstraction can help improve performance by making learning easier and by speeding up simulations during planning.

Combining learning abstractions with guarantees is challenging. In fact, whether it was possible to bound the value loss of model-based RL using a given η-approximate model-similarity abstraction was an open question until our work in Chapter 3. What we can say is that theoretically, for any abstraction, there exists some η such that the bound in (4.21) holds, even if this bound becomes vacuous at higher values. However, it is challenging to estimate or verify the actual value of η in practice. This motivates *abstraction selection* [39, 40] as an important direction for future work, allowing agents to adaptively switch between representations over time.

Given the challenge of determining whether a specific abstraction will be beneficial, abstraction selection could be a viable approach. This can be particularly challenging in RL, where the problem is often not fully known. With abstraction selection, the algorithm would choose which abstraction to select during learning. For instance, selection could be done by deriving value loss bounds for specific abstractions [135] and using these to make a decision. Alternatively, the problem of abstraction selection could be viewed as a non-stationary multi-armed bandit problem, where at each episode we select a (abstract) model. The problem is non-stationary since the models the agent learns change each episode with the experience it obtains, and thus the policy and the expected reward can also change. Multi-armed bandit methods that deal with such non-stationarity could be used [183–185].

Regarding the quality of the abstraction, represented by η, we can consider

different quantities of a problem to determine whether η is small. For example, in the context of approximate abstractions in MDPs, we can consider approximate model similarity or approximate Q^* abstractions [16]. In an approximate model similarity abstraction, the size of η depends on how close the transition functions and rewards of the grouped states are. For approximate Q^* abstractions, it depends on the maximum difference in the Q^* values of the grouped states. Extending these notions to F-POMDPs requires accounting for the observation function and the factorization structure, making such extensions non-trivial. Nevertheless, existing work on approximate abstractions in MDPs provides valuable insights: results indicate that η remains small when little information is lost through abstraction. In our setting, this suggests that η is small when the abstraction primarily removes largely irrelevant factors.

Our abstraction approach has been developed within the framework of FBA-POMDPs. However, the proposed abstraction method is general and leverages only the structural properties of the problem, making it potentially beneficial for a broader range of F-POMDP algorithms. An interesting direction is to investigate how Thompson Sampling [186], particularly its adaptation to POMDPs [187], could be combined with factored representations and our abstraction framework. Such a combination may reduce the dimensionality of the sampling space and further improve scalability.

4.7. CONCLUSION

We proposed combining learning and online planning for BRL for FBA-POMDPs with abstraction. We empirically showed that this combination significantly improves learning, scalability, and performance, using an intuitive and straightforward form of abstraction. This happens through several effects. First, we have shown that abstraction improves performance by increasing the simulation speed. Moreover, we have shown that abstraction improves performance even with a fixed number of simulations through greater statistical strength. With an abstract model, the agent takes different actions since it does not need to explore the dynamics of less relevant factors. Finally, the abstraction method allows FBA-POMCP to learn in very large problems with a state space up to $|\mathbb{S}| = 6 \times 10^{25}$.

To the best of our knowledge, this is the first work to explore abstraction in BRL for POMDPs, representing an initial step in investigating this combination. In the future, abstraction could also be further incorporated into FBA-POMDP by directly maintaining a belief over which factors should be part of the model.

4.8. APPENDIX

4.8.1. EXTENSION BELIEF TRACKING IN THE FBA-POMDP

The belief, the posterior over the current state, is a probability distribution over the POMDP state and its distribution $b \in \Delta(\mathbb{S} \times \mathbb{D})$ given the observed history $h_t = (a_0, o_1, \ldots, a_{t-1}, o_t)$. Unfortunately, the computation of the belief update is intractable in most problems. Thus, the belief is often approximated with particles instead [155]. A particle filter represents a distribution through *particles*, which in this case represent FBA-POMDP states, specifically each particle is a weighted FBA-POMDP state (w, s, G, χ) with (unnormalized) weight $w \in \mathbb{R}^+$.

There are numerous sampling mechanisms for updating the belief given a new action-observation pair [155]. Here, we focus on sequential importance sampling (SIS). SIS consists of two operations: propagation and re-weighting (see Algorithm 8). First, the *proposal distribution* propagates a particle by sampling its next value from the FBA-POMDP transition function $p(\dot{s}'|\dot{s}, a)$. Then, the likelihood of the particle generating the received observation $p(o|\dot{s}, a, \dot{s}')$ is used to re-weight the particle (recall (4.3)). The initial belief (particle filter) is initialized by sampling from the priors over the POMDP state and the DBNs describing the dynamics (see Algorithm 9).

Specific to the FBA-POMDP, the belief over the topologies can deteriorate: the number of unique graph structures is determined (and limited) by the initial particle filter, as topologies do not get updated during the belief updates. When that happens, Katt, Oliehoek, and Amato [61] re-invigorate the belief with a Metropolis-Hastings-within-Gibbs sampling procedure [188].

4.8.2. ALGORITHMS

Algorithm 10 shows the main loop of the FBA-POMCP algorithm [61]. The *Simulate* that is uses is shown in Algorithm 11. The GreedyActionSelection selects the action that has the highest value. In case of a tie, it randomly selects one of the actions with the highest value. Algorithm 12 shows the *Step* function when abstractions are used.

Algorithm 8 Sequential Importance Sampling

Input: $\{\dot{s}_i, w_i\}_{i=0}^n$: current (weighted) filter
a, o: action and observation
for $i \in 0, \ldots, n$ **do**
  $s'_i \sim p(\cdot|s_i, a; G_i, \chi_i)$
  $w'_i \leftarrow w_i \times p(o|s'_i, a; G_i, \chi_i)$
  $\chi'_i \leftarrow \mathcal{U}(\chi_i, s_i, a, s'_i, o)$
end for
return $\{s'_i, G_i, \chi'_i, w'_i\}_{i=0}^n$ // Normalize & re-sample

Algorithm 9 Initialize Particle Filter

Input: $p_{\dot{s}_0}$: prior over initial state
n: number of desired particles
for $i \in 0, \ldots, n$ **do**
  $\langle s_i, G_i, \chi_i \rangle \sim p_{\dot{s}_0}$
  $w_i \leftarrow \frac{1}{n}$
end for
return $\{s_i, G_i, \chi_i, w_i\}_{i=0}^n$

Algorithm 10 FBA-POMCP

Input: B: particle filter with hyper-states $\dot{s}$
num_sims: number of simulations to do.
$h_0 \leftarrow ()$ // The empty history (i.e., now)
for $i \in 1, \ldots,$ num_sims **do**
  // First, we root sample a hyper-state:
  $\dot{s} \sim B$ // Sample from belief
  Simulate$(\dot{s}, 0, h_0)$
end for
$a \leftarrow$ GreedyActionSelection(h_0).
return a

Algorithm 11 Simulate

Input: $\dot{s} = \langle s, G, \chi \rangle$: hyper-state
d: search depth
h: simulated history.
if IsTerminal(h)||$d ==$ max_depth **then**
  return 0
end if
$a \leftarrow$ UCBactionSelection(h)
$R \sim R(\dot{s}, a)$
$\dot{s}', o \leftarrow$ Step$(\dot{s}, a)$
$h' \leftarrow (h, a, o)$
if $h' \in$ Tree **then**
  $r \leftarrow R + \gamma$ Simulate$(\dot{s}', h')$
else
  ConstructNode(h')
  $r \leftarrow R + \gamma$ RollOut$(\dot{s}', h')$
end if
$(\ldots)$ // Update statistics in nodes
return r

Algorithm 12 Step (with abstraction)

Input: $\bar{s}$: abstracted hyper-state
a: simulated action.
// Recall that $\bar{s} = \langle \dot{s}, \bar{G}, \bar{\chi} \rangle$, with $\dot{s}$ containing a current state s.
$s', o \sim p_{\bar{G}, \bar{\chi}}(\cdot | s, a)$ // Sample next state and observation from abstracted counts.
$\dot{s}' \leftarrow \langle s', G, \chi \rangle$
$\bar{s}' \leftarrow \langle \dot{s}, \bar{G}, \bar{\chi} \rangle$
return $\bar{s}', o$

4.8.3. PROOF

We restate the Lemma from Section 4.3.4 and give a proof sketch:

Lemma 1. *The result of applying the abstraction method, as described in Algorithm 5 and Section 4.3.3, to the original FBA-POMDP results in another FBA-POMDP, the abstract FBA-POMDP.*

Proof. First, we define the observation space and function, followed by the state space for different levels of abstraction. The reward function does not require changes since, during abstraction, the models in the particles will always keep the factors that are believed to be part of the IR set.

As detailed in Section 4.3.3, the abstract observation space $\bar{\mathbb{O}}$ is enhanced by including a "not observed" option for all observation variables. This addition addresses scenarios where the observation function depends on state factors that are abstracted away; in such cases, the observation function will return "not observed". Otherwise, the observation space and function remain unchanged.

For the state space, we distinguish abstraction level k_0 from other abstraction levels. For k_0, only the factors in the set IR are included, while higher abstraction levels can include additional factors depending on the topology.

For level k_0, the state space $\bar{\mathbb{S}}$ is derived by aggregating states based on the distinct values of the remaining factors. Transition functions are adjusted for the new state space through the marginalization procedure described in Section 4.3.3.

For higher abstraction levels, the belief specifies the factors that are relevant. For example, for abstraction level k_1, factors that directly influence the IR set are also considered relevant. Similar to k_0, factors that are never considered relevant are removed. For state factors included in only some models, the abstract state space is enhanced by adding a "not relevant" value. This value ensures transitions default to "not relevant" when the factor is not present in a model. Transitions are then adjusted for the new state space through the marginalization procedure. The observation function is also adjusted to return "not observed" for factors that return "not relevant".

Combining these transformations, we obtain a fully specified FBA-POMDP after abstraction. □

In practice, implementing the "not relevant" value is unnecessary, as only factors included in the simulated particle affect observations and actions during tree search.

4.8.4. EXTENDED EXPERIMENT DETAILS

EXPERIMENTAL SETUP

In the experiments, we investigated various levels of abstraction across different domains and considered both a fixed amount of simulations and a fixed amount of computation time. For each level of abstraction (including the full model) and each of the different settings, we ran a separate experiment. Due to the stochasticity in the runs, we conducted up to 10000 runs for each experiment. In the figures, we report the moving average of the returns over a window of 10 ($\frac{x_n+\cdots+x_{n+9}}{10}$), with the shaded areas indicating the 95% confidence interval. To avoid cluttering the figures with markers, we placed only 5 markers per line, spaced evenly along the x-axis.

Table 4.6: Fixed experiment settings.

Parameter	Corridor	Cracky Pavement	Collision	Room Conf
γ	0.95	0.95	0.95	0.95
# of particles in belief	500	500	500	10000
# of episodes	100	500	500	50
# of runs	100	100	10000	1000
Horizon (H)	20	12	20	13
UCT constant	5	1	500	10
Reinvigoration	Yes	No	No	Yes
Log-likelihood threshold	-1500	N/A	N/A	-500
State factors	5	[23, 83]	7	15
$\lvert\mathbb{S}\rvert$	320	$[5 \times 10^7, 6 \times 10^{25}]$	6000	196608

The settings of the experiments, and some specifics of the environments, are detailed in Table 4.6. In the table, γ denotes the discount factor, the UCT constant is the exploration constant, and the log-likelihood threshold is the threshold below which reinvigoration is triggered. The log-likelihood is obtained during the belief update, and a low log-likelihood can indicate the belief does not adequately represent the observed data. The parameters were chosen to maintain a reasonable total run time. Because of this, we ran the experiments in the Cracky Pavement Gridworld and the Collision avoidance domains without the invigoration step. We show in Appendix 4.8.5 that the effect of abstraction is orthogonal to the effect of reinvigoration.

We performed the experiments with a fixed number of simulations on three different machines: Intel(R) Xeon(R) Gold 6130 CPU @ 2.10GHz with 384GB RAM, Intel(R) Xeon(R) Gold 5218 CPU @ 2.30GHz with 190GB RAM, and AMD EPYC 7452 32-Core Processor CPU @ 2.0GHz with 256GB RAM. For the experiments with a fixed amount of computation time, we used (2 cores of) an AMD EPYC 7452 32-Core Processor CPU @ 1.5GHz with 512GB RAM. The software is written in C++.

DOMAIN DESCRIPTIONS

Corridor The Corridor domain is the domain described in the running example in Section 4.2.1, where there are 8 different *Persons* that can be present. Each person has a slightly different probability of opening the door, such that the probability that the person opens the door during an episode is approximately between 0.0125 and 0.1. In this domain, we assume prior knowledge of all observation functions and the locations of the start, boots, button, door, and goal. In addition, we assume the structure of the transition functions of the factors *Button*, *Boots* and *Person* are known, in other words, the prior includes only models where the parents of these factors are correctly specified.

What is unknown is the structure of the transition function of the x-position, for the actions left and right. For this transition, the prior includes the following combinations of factors as parents for the x-position transitions, 1) x, 2) x and

Boots, 3) x, *Boots*, and *Door*, and 4) x, *Boots*, *Door*, and *Button*. Each of these combinations is assigned the same probability in the prior. The transition function of the door is also not fully known, with the prior specifying a 50% chance of *Button* being present as a parent. The prior is initialized optimistically, and so the agent initially overestimates its chances of success and the probability that the person that is present will open the door. It has to learn the correct structure and transition probabilities.

We consider the abstractions k_0, k_1, and the full model k_{inf}. The abstraction k_0 only includes the factor x. Abstraction k_1 includes x and, depending on the structures included in the belief, can also include the factors *Boots*, *Button* and *Door*.

Cracky Pavement Gridworld The Cracky Pavement Gridworld is a grid world as shown in Figure 4.3a. The DBN of the domain is shown in Figure 4.3b. The state space is factored into the x and y locations, *Rain*, and several extra binary factors. These extra factors could be global (e.g., light conditions) or local (e.g., presence of a chair in a specific location). In reality, only x, y, and *Rain* influence the movement of the agent, as can be seen by the incoming edges of x and y. The "Trap" and "Vendor" tiles depicted in Figure 4.3 do influence the movement of the agent but are not included as separate factors because their dynamics are already captured by x, y, and *Rain*. Their interaction is described in the next paragraph.

The agent is initially located at "Start" and is running low on battery, so it has to move to a charging station at one of the "Goal" locations. The agent only observes its x and y location and does so with a noisy sensor. For both x and y, the agent makes the correct observation 90% of the time. If an incorrect observation occurs, a randomly selected adjacent location is returned. At the edges, the probability of receiving the correct observation increases to 95%. The agent can move in all four directions, but the movements can fail. The chances of movement failure are influenced by a global factor called *Rain* which represents whether the tiles are dry or wet. The *Rain* factor is initialized randomly, and every timestep there is a 5% chance that the rain condition changes. On normal tiles, actions succeed 95% of the time when there is no rain and 66% of the time when it is raining. However, on the Trap tiles with cracked pavement, moves succeed only 10% of the time. Additionally, when it is not raining, a vendor with a cart occupies the Vendor tile, making it harder to move past with only a 10% success rate for moving. Conversely, during rain, the Vendor tile is vacant and functions like a normal tile. Therefore, to behave optimally, the agent should traverse the Vendor tile when it is raining and circumvent it when dry.

In this domain, we assume prior knowledge of all observation functions, the transition functions of *Rain* and the extra binary factors, and the start and goal locations. Additionally, we assume that it is known that the x and y factors both (at least) depend on each other. However, there is no prior knowledge of the trap locations and the vendor locations, meaning that none of the possible count tables in the belief space specify different movement probabilities on these locations. Some of the graph structures in the belief space include *Rain* and/or extra binary factors (three in Figure 4.3b) as parents of x and y, implying the agent does not know

whether or not these factors influence its movement. The extra binary factors are initialized randomly and have a 20% of changing at each step. In this problem, the agent has to learn that x, y, and *Rain* are the only relevant factors for its movement. This task is complicated by the interaction between the trap states and several uninformative factors, as the agent may mistakenly attribute its inability to move on trap states to the presence of some of these uninformative factors.

As shown in Table 4.6 on page 104, we run the experiments with two different amounts of extra binary factors: 20 and 80. The total amount of state factors is 23 and 83, respectively, since both settings also have the X, Y, and *Rain* factors. Scalability in the number of factors is very hard and important because the size of the state space, and the possible graph structures, grow exponentially with the number of factors. With 20 and 80 extra binary factors this domain has a state space of approximately 5×10^7 and 6×10^{25} states, respectively. Factored representations are needed to find solutions for problems of such sizes. Flat learning methods like Bayes-Adaptive POMCP [60] are not feasible here, even in the simple case of 20 extra binary factors. This is because representing the transition table for just 1 action, a table of size $|S|^2$, requires more than 9 million GB per particle.

We consider the abstractions k_0, k_1, and the full model k_{inf}. The abstraction k_0 includes only the factors x and y since they are the ones that directly influence the reward (Figure reffig:overviewgridb). Abstraction k_1 also includes the parents of x and y. We scale the number of extra binary factors to test the speedup and to see how it affects the performance.

Collision Avoidance In the Collision Avoidance domain, the agent flies from one side to the other in a 10 (width) x 5 grid. The episode ends when the agent reaches the last column, where it has to avoid colliding with a moving obstacle. This obstacle has a 20% chance to stay stationary and otherwise randomly moves either up or down. The agent can decide to move up, down, or stay level.

We increased the complexity of the original Collision Avoidance [61, 160] by adding additional factors. These additional factors influence those in the original problem. Figure 4.5 shows the resulting dynamics. We added the factors *Speed* (slow, fast), *Traffic* (low or high amount of traffic), *Time of Day* (day, night), and *Obstacle Type* (3 types, e.g., helicopter, plane). *Obstacle Type* and *Time of Day* are fully observable and do not change during the episode. The agent receives a noisy observation of the obstacle (accurate around 80% of the time). The agent has an 85% chance to move one cell forward. If the *Speed* is high, it has a 15% chance to move forward two columns. If the *Speed* is low, it has a 15% chance to stay in the same column.

The *Speed* is influenced by the *Traffic*. When the amount of *Traffic* is low, there is a 90% chance that the *Speed* changes to (or stay at) high and a 10% chance that the *Speed* changes to (or stay at) low. This is reversed when the *Traffic* is high. The *Traffic* is influenced by the *Time of Day* in a similar way, when it day there is a 90% chance that the *Traffic* changes to (or stay at) high and a 10% chance that the *Traffic* changes to (or stay at) low. This is reversed when it is night. The Time of Day is randomly chosen at the start of an episode, with a 50% chance of either day or night.

We assume prior knowledge of all transition and observation functions, except for the transition probabilities of the obstacle. There are four different graph structures for the obstacle transition function to which the prior assigns a positive probability. These structures include: 1) the structure where *Obstacle Y* is only influenced by itself; 2 and 3) the structures where it is influenced by itself and either *Speed* or *Obstacle Type*; and 4) the structure where it is influenced by itself, *Speed*, and *Obstacle Type*. Each structure has a 25% probability.

We consider the abstractions k_0, k_1, and k_2, which is equivalent to the full model k_{inf}. The abstraction k_0 includes the factors x, y, and *Obstacle Y*. The abstraction k_1 additionally includes the factors *Speed* and *Obstacle Type*, if *Obstacle Type* has a connection to *Obstacle Y*.

Room Configuration In the Room Configuration domain, the task of the agent is to set up a classroom in a desirable way for a teacher. We model this environment as a 4 (width) x 3 grid, where each tile contains a configurable item with two settings. The agent can change the configuration of these items. The teacher is concerned only with the configuration of three specific items and is happy once these are configured correctly.

The reward function is (largely) unknown to the agent in this domain, as the agent does not initially know configuration factors that influence the reward. To address this, we model the reward with a state factor called *Happy*, which takes on the same value as the reward and is fully observable. While *Happy* is fully observable, the agent does not know exactly which configuration factors influence it. That is, the prior belief assigns a non-zero probability to multiple sets of parents of *Happy*. Therefore, the agent must learn which configuration factors are relevant to the reward. The *Happy* factor has four different states: neutral (0 reward), slightly unhappy (-1), slightly happy ($+1$), and very happy (100). The agent receives a reward when it performs the switch action, and this reward depends on its location and the status of the configurable items. Specifically, the agent receives a small reward or penalty (± 1) when it changes the configuration of an item to the correct or incorrect setting, respectively. Once it sets the configuration of all three items the teacher desires, it receives a large reward (100).

While the configurations of the items are fully observable, the agent receives noisy observations of its own location. For both x and y, the agent makes the correct observation 95% of the time. If an incorrect observation occurs, a randomly selected adjacent location is returned. At the edges, the probability of receiving the correct observation increases to 97.5%. Movement actions have a 95% chance of moving in the intended direction, whereas the *switch* action to change the item settings is always successful. The agent has prior knowledge of the observation function and the transition function for switching. Additionally, all the possible initial belief states underestimate the probability that the move action is successful.

We consider the abstraction k_0 and the full model k_{inf}. The abstraction k_0 keeps the factors connected to the reward. The structures in the prior belief always include x, y as parents of *Happy*. The prior belief is defined such that it assigns a non-zero probability only to structures where the abstract models k_1 and k_0 are identical.

Thus, in the experiment, we only use k_0.

4.8.5. FURTHER EXPERIMENTS

Here, we demonstrate that abstraction can improve performance when we use the invigoration step from FBA-POMCP, and that k_1 and the full model k_{inf} can eventually outperform k_0. We conducted 100 runs for each setting and report the simple moving average of the return per episode, with the shaded areas showing the standard error.

We ran an experiment with invigoration over a larger number of episodes, as shown in Figure 4.9. Initially, k_0 outperforms k_1 and k_{inf}. However, with enough data, the k_1 and k_{inf} models become accurate enough to surpass k_0 in performance. The k_1 model matches the performance of k_{inf} because the number of simulations is fixed.

For larger problem sizes and with a fixed thinking time instead of a fixed number of simulations, k_1 is expected to initially outperform k_{inf}. Additionally, k_0 should show an even greater initial improvement over k_{inf}, and it could take even longer for k_{inf} to surpass the performance of k_0.

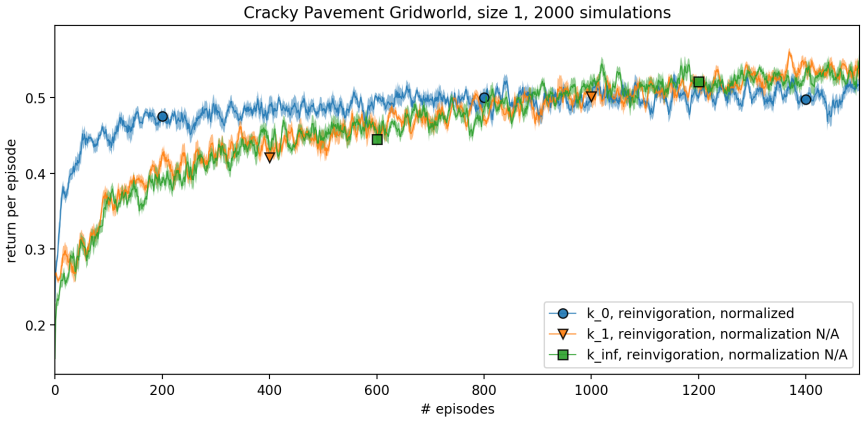


Figure 4.9: Comparison with invigoration for a longer number of episodes.

5

DISCUSSION AND CONCLUSION

We provide a summary of the key findings and their implications as explored in each chapter. We then review the scope of our work and current trends in the field. Next, we examine the limitations of our methodologies and findings, discussing recommendations for future research to address these challenges. Building on the results of Chapter 3, we propose a way to extend its application to alternative types of abstractions. Subsequently, we integrate the results from Chapters 3 and 4, providing a unified perspective on abstraction dynamics and their broader implications. Then, we go into the advancements presented in Chapter 4, investigating how varying abstraction levels can be utilized to balance the trade-off between accelerated learning and model accuracy. This incorporates insights from research on abstraction selection and multi-armed bandits. Finally, we address limitations in our current understanding of learning with abstractions from a theoretical and practical perspective.

5.1. CHAPTER CONCLUSIONS

This thesis has aimed to increase the efficiency of Reinforcement Learning (RL) by exploring the integration of abstraction and Model-based Reinforcement Learning (MBRL). In this setting of MBRL from Abstracted Observations (MBRLAO), the agent acts in an Markov decision process (MDP) but instead of observing the true states, the agent only observes abstract states through an abstraction function $\phi(s)$. Both abstraction and MBRL independently offer significant potential for improving learning efficiency. However, their combination introduces not only opportunities but also notable challenges, which have both been thoroughly analyzed and discussed throughout this work.

In Chapter 2, we introduce a novel framework to interpret existing MBRLAO approaches. Within this framework, we categorize approaches into two groups: those that utilize one or more given abstraction functions and those that concurrently learn abstractions while exploring and modeling the environment. Through this lens, we identified benefits of integrating MBRL with abstraction, as well as open challenges that warrant further investigation.

A key insight from our analysis is that MBRLAO can be conceptualized as

transforming the learning problem from a (fully observable) MDP to a Partially Observable Markov Decision Process (POMDP). This transformation to a POMDP has implications for both learning dynamics and solution strategies, as guarantees for methods in MDPs may no longer hold. This presents an important open question: how can we ensure efficient learning in MBRLAO?

Furthermore, we identified an interesting opportunity for leveraging abstraction in the context of MBRL with online planning. By simplifying state spaces, abstraction has the potential to lead to faster planning, which can result in better performance with limited planning time compared to planning with the full model. For RL it could be that improved planning performance with abstraction also leads to improved learning performance. This leads to an important open question: how can abstractions be effectively employed in domains requiring both learning and online planning? Specifically, under what conditions could a coarser model facilitate improved performance by balancing computational efficiency, learning speed, and model fidelity?

In Chapter 3, we examine the theoretical aspects and challenges associated with MBRLAO. Building on the insight from Chapter 2 that MBRLAO transforms a fully observable problem into a partially observable one, we investigate how using the abstraction function affects the characteristics of the observed samples. Specifically, we introduce an example that demonstrates that even in the context of an approximate model-similarity abstraction, the resulting transitions are no longer Markovian: for a given abstract state, the subsequent abstract states are neither guaranteed to be identically distributed nor independent samples. Since the Markov assumption is key in the performance guarantees of MBRL, these guarantees do not directly transfer to learning with abstraction.

To address this challenge, we show that the learning process constitutes a martingale difference sequence [53, 54]. Leveraging the properties of martingale processes, we prove that it is possible to provide guarantees on the quality of the learned model. For the first time, our results allow us to transfer the theoretical results for MBRL to MBRLAO, particularly when using an approximate model-similarity abstraction. We provide an explicit demonstration of this in the context of the well-known R-max algorithm [13]. Our analysis proves that it is indeed possible to integrate MBRL with abstraction, enabling more efficient learning of near-optimal solutions. This work directly addressed the open question posed in Chapter 2 regarding how we can ensure efficient learning in MBRLAO.

From a broader conceptual perspective, our results show that treating a non-Markov problem as if it were Markov is feasible, provided the dynamics are sufficiently similar. In this way, our findings provide a formal understanding of why MBRL can be applied with success even though the Markov property rarely holds exactly.

In Chapter 4, we address the open questions raised in Chapter 2 regarding the effective use of abstractions in domains that require both learning and online planning. We investigated the creation of abstractions and their application in learning and online planning within a partially observable setting.

Creating effective abstractions in RL is challenging due to the need to group states with similar characteristics. Creating such an abstraction requires knowledge

of the problem domain; however, in RL, we typically assume minimal or no prior knowledge of the environment. To overcome this challenge, we adopt a factored model framework combined with a Bayesian approach. The factored Bayesian model enables us to incorporate prior knowledge and leverage the structural characteristics of the problem to inform the abstraction process. The structural characteristics help in identifying which factors can be considered less relevant and safely removed, minimizing the loss of essential details. Our approach involves maintaining both the original and abstract models throughout the learning process. This methodology enables the abstract model to change over time, as we more accurately learn the structure of the problem through the original model.

Empirically, we evaluated our approach in several domains to investigate the potential benefits of abstraction. The experiments in these domains yielded several important insights. First, abstract models can enhance performance by enabling faster online planning, due to the reduction in the model size. This effect was observed across multiple domains, with the performance gains being stronger as the reduction in model size increased. Interestingly, this benefit was evident not only when the abstract model accurately represented the original problem but also when the abstract introduced a loss of information crucial for achieving optimal performance. In online planning scenarios, the increased speed of simulations provided by abstraction can compensate for the loss of model accuracy.

Furthermore, the experiments demonstrated that abstractions can improve performance even under a fixed number of simulations, due to their greater statistical strength. This advantage arises because abstract models can aggregate multiple samples that the original model treats independently, thereby using experience more efficiently. However, the full model may eventually surpass coarser abstract models in performance as it receives enough data, depending on the specific domain. Nevertheless, achieving this can take a significant amount of time, especially when the abstraction results in substantial planning speed improvements due to a reduction in model size. During this interim period, the performance of the abstract models can be significantly better, highlighting their importance in early-stage learning.

5.2. SCOPE AND RELEVANCE TO CURRENT TRENDS IN RL

We studied the combination of state abstraction and MBRL in MDPs and POMDPs, with the primary goal of increasing the efficiency of RL. To study this, we surveyed existing literature, with a particular focus on theoretical results. With the gained oversight on the topic we identified current weaknesses in the knowledge on the combination of MBRL and state abstraction and developed theory to address this. To demonstrate the power of this combination we performed experiments in several simulated domains.

Still, we would envision RL being more present in real life, and we have only demonstrated their usefulness in relatively simple domains that are far from real-world complexities. Despite this limitation, our theoretical and empirical results are evidence that the combination of MBRL and abstraction can lead to more

efficient learning. To demonstrate further usefulness in real-life scenarios limitations need to be addressed and further steps taken, some of which we discuss in Sections 5.3 and 5.4.

Since the success of deep RL in Atari games around 2015 [8], a clear trend in RL has been to direct efforts toward deep neural networks. Our work is largely independent of this trend, since we prioritize learning efficiency. As discussed in Chapter 1, most current deep RL approaches either lack efficient exploration or depend on simulators that may not always be available. That said, there is a connection with our work since deep RL methods can be viewed as learning an abstract representation through neural networks, and the insights developed here may therefore prove useful in that context. We briefly review model-free and model-based deep RL methods and consider how our contributions could benefit work in this area.

The initial focus with deep neural networks was mainly on model-free deep RL methods [8, 147, 189, 190]. In particular, deep Q-learning methods can be interpreted as learning an exact Q^* abstraction, since they attempt to learn optimal Q-values. Our results show that approximate abstractions can still achieve strong performance. Thus, it could be beneficial to concentrate on learning approximately optimal Q-values, which could, for instance, be done by changing the loss function. This could lead to improved learning speed and performance.

Despite these advances, model-free deep RL methods have often performed poorly in more challenging environments. To improve the performance in complex environments, where a good exploration strategy is more important, there has been an emphasis on *deep* exploration [63, 66, 80, 191–197]. These deep exploration methods essentially reimplement existing exploration techniques in deep RL and often use some measure of uncertainty. While the theoretical guarantees of existing exploration techniques can no longer be assured with neural networks, implementing these techniques generally leads to better exploration and performance. Nevertheless, recent studies have demonstrated that uncertainty estimation in deep RL can be unreliable [198, 199]. This can be especially problematic in real-world applications where mistakes can be costly.

To mitigate the difficulties associated with uncertainty estimation, current work has proposed learning without relying on uncertainty estimation [200]. For example, one line of work formulates the problem of RL in a pessimistic way by assuming the worst case transitions, creating a pessimistic, or adversarial, MDP [201]. In learning, they alternate between optimizing the policy and (adversarially) optimizing the model. This pessimistic approach provides a principled way to sidestep the need for explicit uncertainty estimation, and it may also prove valuable in MBRLAO, where abstraction naturally introduces uncertainty. Our work, by contrast, does not operate in the deep RL setting. Instead, we combine uncertainty estimation with an optimistic perspective to demonstrate that efficient learning in MBRLAO can be achieved not by avoiding uncertainty, but by exploiting it. While extensions to deep RL may be less straightforward, our results highlight optimism as a powerful and principled tool for addressing uncertainty in abstract models.

An emerging line of work has concentrated on addressing the lack of formal

guarantees in learning with deep RL [202–204]. These approaches use variational autoencoders [202, 205, 206] or Wasserstein autoencoders [203, 204, 207] to create discrete latent space models. In [203], such models enable the distillation of formally verifiable controllers from any RL policy, resulting in a so-called distilled policy. The Wasserstein autoencoder improves upon the variational autoencoder by addressing some of its limitations and providing abstraction and representation guarantees, which support a more principled compression of the state space. This approach also leads to bisimulation guarantees, representing a significant step towards providing formal guarantees in deep RL. Furthermore, [204] demonstrates that Wasserstein autoencoders can also provide guarantees for learning in POMDPs. These results are complementary to our work, in that we focus primarily on approximate (bisimulation) abstractions, and it may be possible to extend this line of research to use approximate rather than exact abstractions while still retaining formal guarantees. In this way, our results for MBRLAO with approximate abstractions could be extended to deep RL, enabling a greater reduction of the state space and potentially improving learning efficiency compared to the use of exact bisimulations.

The latest work shows growing interest in model-based deep RL [45, 46, 58, 79, 100, 117, 119, 124, 168, 208–212]. These methods learn a representation of the environment and often contain a planning component, using the learned model to plan actions. These methods are promising since they can guide exploration and find solutions more efficiently. Our work suggests that learning approximate representations instead of the exact dynamics could benefit both learning and planning. Approximating the dynamics could speed up learning by aggregating experience. For planning, while an approximate representation will not reduce the size of the model, since the neural network will remain the same size, it can still be beneficial since our results show that planning with an abstract model can improve performance even when there is no speedup in the simulations.

Recent studies have explored combining temporal abstraction with deep RL [28, 213]. In [28], rewards are assumed to be compositional: they can be expressed as a set of short-term subgoals with a final goal. The method employs a hierarchical approach, using high-level planning together with model-free RL to achieve the subgoals. For high-level planning, a graph is constructed from the reward specification, where edges correspond to subtasks and vertices correspond to states in which subtasks are achieved. A forward graph search algorithm is then applied to compute a policy over the completion of subtasks. In [213], subgoal regions (subsets of states) are also assumed to be specified and are used as a form of state abstraction. Options are then learned as actions that transition between subgoal regions, thereby integrating both state and action abstractions. Planning is done for the resulting abstract problem. This is similar to our MBRLAO approach, where planning is also performed in the abstract MDP, but instead of learning options, we use the ground actions directly to model transitions between abstract states. Options could be particularly useful for online planning, such as in the work of Chapter 4, by shortening the effective planning horizon. In contrast, ground actions provide more fine-grained control when precise behavior is needed.

Finally, ongoing development is the combination of planning and RL with

foundation models [214]. Foundation models are trained on large-scale, diverse data and can be adapted to a wide range of tasks. A prominent example is Large Language Models (LLMs) such as ChatGPT [215]. These models have demonstrated promising results for instance in zero-shot planning [216]. More recently, LLMs have been integrated with RL, where the LLM guides exploration through its use as a prior and the RL helps finetune performance in tasks through learning [217]. In this context, the LLM can be viewed as providing prior knowledge. This approach is particularly appealing for work on (deep) Bayesian learning, such as in Chapter 4 and related studies [167, 168], since reliable (even suboptimal) priors can enhance learning performance in deep RL [218].

5.3. LIMITATIONS

We mention several limitations of our work, including new limitations that became apparent upon further analysis and limitations already discussed in earlier chapters, which we expand upon here.

Revisiting the results of Chapter 3 provided the insight that state-action abstraction is a natural extension of state abstraction. While that chapter focused on state abstraction, much of its analysis centered on state-action pairs. Since state abstraction can be viewed as a special case of state-action abstraction, a greater focus on state-action abstraction would have been warranted in Chapter 2. Further, expanding the standard definitions of exact and approximate state abstractions to state-action abstractions could have been used in Chapter 3 to yield more generalizable results.

Chapter 3 establishes guarantees for learning with an approximate model-similarity abstraction. A limitation of this work is the implicit assumption that a good abstraction function is readily available. Guaranteeing the quality of such an abstraction in advance is particularly difficult in RL, as the framework typically operates under minimal prior knowledge of the problem domain. This was partially addressed in Chapter 4, where we investigated how to construct abstractions a priori. Still, it may not always be clear if an abstraction improves performance. In this case, we may be able to resort to abstraction selection. We further discuss this idea in Section 5.4.

While the results in Chapter 3 demonstrate that an abstract model can be learned accurately for any type of state abstraction, we have extended the results of MBRL to the setting with abstraction only for the approximate model-similarity abstraction. Since the most effective type of abstraction for improving performance may vary depending on the problem domain, it would be worthwhile to explore whether similar theoretical results can be derived for other types of abstractions. This possibility will be discussed further in Section 5.4.

In Chapter 4, our approach for creating abstract models decides which factors to remove based on their distance to the factors directly relevant to the reward. However, this may not always lead to good abstractions. For example, factors outside the abstract model could strongly influence those inside the abstract model. More sophisticated methods could be used for deciding which factors to exclude. Instead

of removing factors based only on distance, one could estimate the impact of a factor on the factors in the abstraction and only remove it if its impact is below a threshold. Alternatively, a combination with influence-based abstraction [32] could be effective, where we learn the influence of the factors outside the abstraction on the factors inside the abstraction.

There is a clear similarity between the theoretical results of Chapters 3 and 4, since both results relate to learning with abstraction and give learning results in the setting where we learn an abstract model. There are also important differences that were not discussed in the chapters. Chapter 3 highlighted that the abstract solution might change over time and that the (abstract) samples are not identically distributed and can not be expected to be independent. The results then showed how we can deal with these difficulties and still provide important results. So why does Chapter 4 seemingly ignore these difficulties? Two crucial differences allow the results of Chapter 4 to implicitly address the issues encountered in Chapter 3.

First, in Chapter 4 we learn *both* the ground model and an abstract model. Instead of directly learning an abstract model, we update the ground model and use this to update the abstract model at each timestep. Learning the ground model means we can rely on analysis without abstraction and do not directly need to deal with dependent and non-identically distributed samples. A downside compared to Chapter 3 is that this could be costly in terms of memory and is less scalable since we have to keep track of the ground model.

Second, in Chapter 3, we provide guarantees on the quality of the learned abstract model and demonstrate that abstractions can accelerate learning. In contrast, Chapter 4 neither guarantees the quality of the learned abstract model nor shows that we can learn quicker. Instead, it uses a different type of abstraction that only relies on its quality η. For this abstraction with quality η, we define that any ϵ-optimal solution π in the abstract model is an $(\epsilon+\eta)$-optimal solution for the original problem. This ensures that, at each time step, the solution converges to an action that is $(\epsilon+\eta)$-optimal in the original Factored Bayes-Adaptive POMDP (FBA-POMDP). While the empirical results showed that this can lead to quicker learning, this analysis does not directly prove this. An analysis similar to that in Chapter 3 could be used for the results in Chapter 4, we will discuss this further in Section 5.4.

5.4. FUTURE WORK

We give a concise discussion on potential extensions of important chapter results and propose ideas for future directions.

EXTENDING RESULTS OF CHAPTER 3

In Chapter 3, we demonstrated that performance guarantees of MBRL methods can be extended to the setting with abstracted observations using an approximate model-similarity abstraction. We also claimed that similar results could be obtained for at least one other type of abstraction: approximate Q^* abstractions.

The results from Chapter 3 show that at least two types of approximate abstractions can be effectively used in MBRLAO. It may be possible to further extend our result to other types of approximate state abstractions [16] and to create insights for approximate abstractions similar to those for exact abstractions [15]. Next steps could include investigating the relationships between the different abstractions. For example, approximate Q^* abstraction can compress more than approximate model-similarity abstractions, since all model-similarity abstractions are approximate Q^* abstractions, but not the other way around.

Further insights into different types of abstraction could improve our understanding of which is most useful for learning. The optimal choice depends on the balance between the degree of compression and the resulting loss in value, which are problem-specific. Understanding which abstraction is preferable in a particular environment can lead to more efficient learning.

Connecting Chapters 3 and 4

In Chapter 3 we used approximate model-similarity abstractions to show that we could efficiently learn in MDPs with abstraction. In Chapter 4 we used a more general definition of abstraction quality, where any ϵ-optimal solution of the abstract problem is $\epsilon+\eta$-optimal in the original problem. We then used this definition to show that applying Factored BA-POMCP (FBA-POMCP) with abstraction will lead to $\epsilon+\eta$-optimal solutions. There is a connection between these results, as an η approximate model similarity abstraction can give $\eta+\epsilon$-optimal solutions in the original problem.

It should be possible to extend the results for approximate model-similarity abstractions in MDPs to POMDPs. There are still some steps to be made, as it requires extending the definition of the abstraction to include the observations and observation functions. Subsequently, the theoretical bounds also need to be updated to take into account the observation space. Much of the analysis relies on simulation lemmas for MDPs, which could be adjusted with a simulation lemma for POMDPs [159]. These steps could more formally extend state-abstraction results for MDPs to POMDPs.

Multi-Armed Bandits for Abstraction Selection in RL

Chapter 4 showed that abstract models can accelerate learning and improve performance. However, it was also shown that the optimal level of abstraction (or no abstraction) varies across environments and over time. The optimal representation depends on several factors. For example, it is influenced by the model fidelity after abstraction, how accurately can the model still represent the true dynamics? Losing too much information can result in not performing at all, while a slight loss of information could improve performance through greater statistical strength. This introduces a trade-off between the model fidelity and statistical strength. In the context of online planning, there is an additional trade-off between the model fidelity and the computational speedup that can be achieved from the reduced model size through abstraction.

Since the best abstraction might vary over time, the question of how to decide which abstraction to select at what time is important for optimal performance. As suggested in Chapter 4, abstraction selection methods and (non-stationary) multi-armed bandits could be used for this. This suggestion deserves more attention for two reasons: there are some difficulties with using (current) abstraction selection techniques in our setting, and using (non-stationary) multi-armed bandit methods could be a promising novel direction.

In our setting, we assume problems are partially observable. Since these problems are huge, finding an optimal solution to the entire problem offline is often impossible, making it necessary to rely on online solution methods. We also assume we do not have access to the true model of the problem, and we want to use (approximate) abstractions to learn more efficiently. Most existing work on abstraction selection does not fit our setting for various reasons. As discussed in Sections 2.3.3 and 3.5, these reasons include among others a focus on finding solutions for a model offline, assuming access to the (abstract) model itself, or just focusing on finding a Markov model instead of focusing on finding the model most efficient for learning [39, 40, 48, 49, 51, 102–104].

We can view the problem of abstraction selection as a (non-stationary) multi-armed bandit problem in the sense that we have multiple representations, and we have to select one representation (abstract model) for each episode. The return obtained within an episode would be the reward for using the selected abstraction, or arm. This problem can be considered non-stationary since we update the models with the experience gained in each episode, this can change the policy and thus lead to different (expected) rewards. Multi-armed bandit methods are interesting since they tackle the fundamental exploration-exploitation dilemma to efficiently learn which arm is the most rewarding [219]. Using non-stationary multi-armed bandit methods could help to efficiently select a good abstraction at any point in time as they are flexible to changes in the model and different representations could be good at various times.

A specific proposition for abstraction selection in episodic RL with abstraction is to use the Sliding-Window UCB# algorithm [220]. This algorithm maintains an estimate of the returns within a sliding window and can be used in environments where the change in the expected rewards can be upper-bounded and is not too large. In FBA-POMCP, some factors make the change in the expected reward likely to be small. First, the change in the model within an episode is generally gradual, as this depends on the experience gained. When the change in the model is small, we expect the change in the policy, and thus the expected reward, to be small as well. Assuming the reward function is known, only the change in the transition model influences the policy. Furthermore, since we use online planning, the resulting policy is stochastic since it depends on the (random) roll-outs in the tree.

ABSTRACTION BOUNDS

In Chapters 2 and 3, we discussed and used theoretical bounds for abstractions. These theoretical bounds can quickly become vacuous, even when this might not be obvious from merely observing the problem. For instance, in a theoretical example,

it has been shown that using the R-MAX algorithm with an ϵ-approximate Q^* abstraction can fail to learn even for small values of $\epsilon = 0.1$ [41]. This example is shown in Figure 5.1, with actions to move to the *left* and *right* or to *loop*. In states 1 and 2 looping leads to some small reward κ, and looping in state 3 leads to the maximum reward $R_{\max}$.

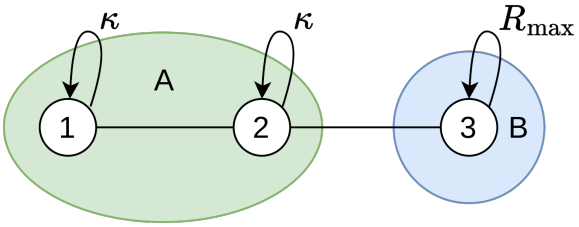


Figure 5.1: Example chain problem. The agent can move left, right, or loop. Looping results in a small reward of κ in states 1 and 2, and in a large reward $R_{\max}$ in state 3.

They grouped states 0 and 1, resulting in an abstract state space with two abstract states, abstract state A containing states 1 and 2 and B containing state 3. They assume that the R-MAX algorithm breaks ties in action selection by choosing actions in a specific order. In this case, by first choosing the action *left*, then *right*, and then *loop*. If this is the case, and the agent starts in state 1, it will only ever visit states 1 and 2 and never reaches state 3. Thus it will just learn endlessly looping for a reward of κ. This means that the R-MAX algorithm with this abstraction leads to making an unbounded number of mistakes.

This was used as an example of an unexpected negative result with abstraction. However, the theoretical bound for this example shows that the bound is vacuous, meaning it is not unexpected that the algorithm does not perform. The ϵ-approximate Q^* bound states that the loss in value when using the abstraction is at most $\frac{2\epsilon R_{\max}}{(1-\gamma)^2} = 80R_{\max}$, which in this problem is four times the maximum value in any state ($\frac{R_{\max}}{1-\gamma} = 20R_{\max}$).

Of course, in problems such as these treating the abstract problem as a POMDP and applying solutions for POMDPs would also quickly lead to an optimal solution. Interestingly, without treating it as a POMDP, a small change would still allow learning an optimal solution using the abstract representation. Where we typically focus on learning deterministic policies, it can lead to failure when using abstraction. This was the case in this example, as their analysis relied on the fact that the algorithm will select actions deterministically. As shown in POMDPs, stochastic policies can perform much better than deterministic ones when treating a POMDP as an MDP [132]. In the example problem, adding stochasticity could allow the algorithm to reach the goal state and achieve the reward, leading to the algorithm finding the correct solution. An interesting direction could be incorporating stochasticity in learning with abstraction. This can prove difficult but could lead to improved results.

Even without stochasticity, abstraction can still be beneficial and provide a good solution even when the bound is vacuous. This can be seen in the experiments of Chapter 4. In the Cracky Pavement Gridworld, the agent cannot distinguish whether or not the vendor is located on the tile. The differences in the transition probabilities with or without the vendor are quite large, making this problematic from a theoretical perspective. However, the agent can learn to simply avoid this tile since, in expectation, it will sometimes cross this tile when the vendor is there, which should lead to the agent learning that crossing this tile is worse than going around it. While this is slightly suboptimal it still leads to a good solution. In this way, even though an abstraction may not be "good" in the sense of the bounds for approximate model similarity, it is still a good abstraction in the sense that an optimal policy can still be learned. It could thus be interesting to look at bounds that can be obtained from the starting state(s) or the path that the abstract policy can learn to follow.

BIBLIOGRAPHY

[1] Merriam-Webster. *Intelligence*. Retrieved November 2, 2024. 2024. URL: https://www.merriam-webster.com/dictionary/intelligence.

[2] D. Silver, J. Schrittwieser, K. Simonyan, I. Antonoglou, A. Huang, A. Guez, T. Hubert, L. Baker, M. Lai, A. Bolton, *et al.* "Mastering the Game of Go Without Human Knowledge". In: *Nature* 550.7676 (2017), pp. 354–359.

[3] D. Silver, T. Hubert, J. Schrittwieser, I. Antonoglou, M. Lai, A. Guez, M. Lanctot, L. Sifre, D. Kumaran, T. Graepel, T. Lillicrap, K. Simonyan, and D. Hassabis. "A General Reinforcement Learning Algorithm That Masters Chess, Shogi, and Go Through Self-Play". In: *Science* 362.6419 (2018), pp. 1140–1144.

[4] O. Vinyals, I. Babuschkin, W. M. Czarnecki, M. Mathieu, A. Dudzik, J. Chung, D. H. Choi, R. Powell, T. Ewalds, P. Georgiev, *et al.* "Grandmaster Level in Starcraft II Using Multi-Agent Reinforcement Learning". In: *Nature* 575.7782 (2019), pp. 350–354.

[5] J. Perolat, B. De Vylder, D. Hennes, E. Tarassov, F. Strub, V. de Boer, P. Muller, J. T. Connor, N. Burch, T. Anthony, *et al.* "Mastering the Game of Stratego With Model-Free Multiagent Reinforcement Learning". In: *Science* 378.6623 (2022), pp. 990–996.

[6] N. Lazic, C. Boutilier, T. Lu, E. Wong, B. Roy, M. Ryu, and G. Imwalle. "Data Center Cooling Using Model-Predictive Control". In: *Advances in Neural Information Processing Systems* 31 (2018).

[7] R. S. Sutton and A. G. Barto. *Reinforcement Learning: an Introduction*. MIT press, 1998.

[8] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, *et al.* "Human-Level Control Through Deep Reinforcement Learning". In: *Nature* 518.7540 (2015), p. 529.

[9] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov. "Proximal Policy Optimization Algorithms". In: *arXiv Preprint arXiv:1707.06347* (2017).

[10] N. Brown and T. Sandholm. "Superhuman AI for Heads-Up No-Limit Poker: Libratus Beats Top Professionals". In: *Science* 359.6374 (2018), pp. 418–424.

[11] M. Jaderberg, W. M. Czarnecki, I. Dunning, L. Marris, G. Lever, A. García Castañeda, C. Beattie, N. C. Rabinowitz, A. S. Morcos, A. Ruderman, *et al.* "Human-Level Performance in 3D Multiplayer Games With Population-Based Reinforcement Learning". In: *Science* 364.6443 (2019), pp. 859–865.

[12] OpenAI, C. Berner, G. Brockman, B. Chan, V. Cheung, P. Dębiak, C. Dennison, D. Farhi, Q. Fischer, S. Hashme, *et al.* "Dota 2 With Large Scale Deep Reinforcement Learning". In: *arXiv Preprint arXiv:1912.06680* (2019).

[13] R. I. Brafman and M. Tennenholtz. "R-max - a General Polynomial Time Algorithm for Near-Optimal Reinforcement Learning". In: *Journal of Machine Learning Research* 3.Oct (2002), pp. 213–231.

[14] A. L. Strehl and M. L. Littman. "An Analysis of Model-Based Interval Estimation for Markov Decision Processes". In: *Journal of Computer and System Sciences* 74.8 (2008), pp. 1309–1331.

[15] L. Li, T. J. Walsh, and M. L. Littman. "Towards a Unified Theory of State Abstraction for MDPs". In: *Proceedings of the Ninth International Symposium on Artificial Intelligence and Mathematics*. 2006.

[16] D. Abel, D. Hershkowitz, and M. Littman. "Near Optimal Behavior Via Approximate State Abstraction". In: *Proceedings of the 33rd International Conference on International Conference on Machine Learning*. 2016, pp. 2915–2923.

[17] R. A. Howard. *Dynamic Programming and Markov Processes*. Cambridge, MA: The MIT Press, 1960.

[18] S. J. Russell and P. Norvig. *Artificial Intelligence: a Modern Approach*. 2016.

[19] L. P. Kaelbling, M. L. Littman, and A. R. Cassandra. "Planning and Acting in Partially Observable Stochastic Domains". In: *Artificial intelligence* 101.1-2 (1998), pp. 99–134.

[20] T. Jaksch, R. Ortner, and P. Auer. "Near-Optimal Regret Bounds for Reinforcement Learning". In: *Journal of Machine Learning Research* 11.Apr (2010), pp. 1563–1600.

[21] M. O. Duff. *Optimal Learning: Computational Procedures for Bayes-Adaptive Markov Decision Processes*. University of Massachusetts Amherst, 2002.

[22] P. Auer and R. Ortner. "Logarithmic Online Regret Bounds for Undiscounted Reinforcement Learning". In: *Advances in Neural Information Processing Systems*. 2007, pp. 49–56.

[23] M. Ghavamzadeh, S. Mannor, J. Pineau, and A. Tamar. "Bayesian Reinforcement Learning: a Survey". In: *Foundations and Trends® in Machine Learning* 8.5-6 (2015), pp. 359–483.

[24] M. G. Azar, I. Osband, and R. Munos. "Minimax Regret Bounds for Reinforcement Learning". In: *Proceedings of the 34th International Conference on Machine Learning - Volume 70*. JMLR. org. 2017, pp. 263–272.

[25] Z. Zhang, Y. Jiang, Y. Zhou, and X. Ji. "Near-Optimal Regret Bounds for Multi-Batch Reinforcement Learning". In: *Advances in Neural Information Processing Systems* 35 (2022), pp. 24586–24596.

[26] R. Schmöcker and A. Dockhorn. "A Survey of Non-Learning-Based Abstractions for Sequential Decision-Making". In: *IEEE Access* (2025).

[27] R. S. Sutton, D. Precup, and S. Singh. “Between MDPs and Semi-MDPs: a Framework for Temporal Abstraction in Reinforcement Learning”. In: *Artificial Intelligence* 112.1-2 (1999), pp. 181–211.

[28] K. Jothimurugan, S. Bansal, O. Bastani, and R. Alur. “Compositional Reinforcement Learning from Logical Specifications”. In: *Advances in Neural Information Processing Systems* 34 (2021), pp. 10026–10039.

[29] M. C. Machado, A. Barreto, D. Precup, and M. Bowling. “Temporal Abstraction in Reinforcement Learning With the Successor Representation”. In: *Journal of Machine Learning Research* 24.80 (2023), pp. 1–69.

[30] A. Bai, S. Srivastava, and S. Russell. “Markovian State and Action Abstractions for MDPs Via Hierarchical MCTS”. In: *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence*. 2016, pp. 3029–3039.

[31] A. Dockhorn and R. Kruse. “State and Action Abstraction for Search and Reinforcement Learning Algorithms”. In: *Artificial Intelligence in Control and Decision-making Systems: Dedicated to Professor Janusz Kacprzyk*. Springer, 2023, pp. 181–198.

[32] F. A. Oliehoek, S. J. Witwicki, and L. P. Kaelbling. “Influence-Based Abstraction for Multiagent Systems”. In: *Proceedings of the 26th AAAI Conference on Artificial Intelligence*. 2012, pp. 1422–1428.

[33] M. Suau, J. He, E. Congeduti, **Starre, R. A. N.**, A. Czechowski, and F. A. Oliehoek. “Influence-Aware Memory Architectures for Deep Reinforcement Learning in POMDPs”. In: *Neural Computing and Applications* (2022), pp. 1–17.

[34] S. P. Singh, T. Jaakkola, and M. I. Jordan. “Reinforcement Learning With Soft State Aggregation”. In: *Advances in Neural Information Processing Systems*. 1994.

[35] C. Allen, N. Parikh, O. Gottesman, and G. Konidaris. “Learning Markov State Abstractions for Deep Reinforcement Learning”. In: *Advances in Neural Information Processing Systems* 34 (2021), pp. 8229–8241.

[36] D. Abel, D. E. Hershkowitz, and M. L. Littman. “Near Optimal Behavior Via Approximate State Abstraction”. In: *arXiv Preprint arXiv:1701.04113* (2017).

[37] C. Paduraru, R. Kaplow, D. Precup, and J. Pineau. “Model-Based Reinforcement Learning With State Aggregation”. In: *Proceedings of the 8th European Workshop on Reinforcement Learning*. 2008.

[38] R. Ortner. “Adaptive Aggregation for Reinforcement Learning in Average Reward Markov Decision Processes”. In: *Annals of Operations Research* 208.1 (2013), pp. 321–336.

[39] R. Ortner, O.-A. Maillard, and D. Ryabko. “Selecting Near-Optimal Approximate State Representations in Reinforcement Learning”. In: *Proceedings of the 25th International Conference on Algorithmic Learning Theory*. Springer. 2014, pp. 140–154.

[40] N. Jiang, A. Kulesza, and S. Singh. “Abstraction Selection in Model-Based Reinforcement Learning”. In: *Proceedings of the 32nd International Conference on International Conference on Machine Learning*. 2015, pp. 179–188.

[41] D. Abel, D. Arumugam, L. Lehnert, and M. Littman. “State Abstractions for Lifelong Reinforcement Learning”. In: *Proceedings of the 35th International Conference on International Conference on Machine Learning*. 2018, pp. 10–19.

[42] T. S. Badings, A. Abate, N. Jansen, D. Parker, H. A. Poonawala, and M. Stoelinga. “Sampling-Based Robust Control of Autonomous Systems With Non-Gaussian Noise”. In: *Proceedings of the 36th AAAI Conference on Artificial Intelligence* (2022), pp. 9669–9678.

[43] **Starre, R. A. N.**, M. Loog, and F. A. Oliehoek. “Model-Based Reinforcement Learning With State Abstraction: a Survey”. In: *Proceedings of the 34th Benelux Conference on Artificial Intelligence (BNAIC) and the 30th Belgian Dutch Conference on Machine Learning (Benelearn), Revised Selected Papers*. Communications in Computer and Information Science. Springer, 2023, pp. 133–148.

[44] A. K. McCallum. “Learning to Use Selective Attention and Short-Term Memory in Sequential Tasks”. In: *From Animals to Animats 4: Proceedings of the Fourth International Conference on Simulation of Adaptive Behavior*. Vol. 4. MIT Press. 1996, p. 315.

[45] E. Van der Pol, T. Kipf, F. A. Oliehoek, and M. Welling. “Plannable Approximations to MDP Homomorphisms: Equivariance Under Actions”. In: *Proceedings of the 19th International Conference on Autonomous Agents and Multiagent Systems*. 2020, pp. 1431–1439.

[46] J. Schrittwieser, I. Antonoglou, T. Hubert, K. Simonyan, L. Sifre, S. Schmitt, A. Guez, E. Lockhart, D. Hassabis, T. Graepel, *et al.* “Mastering Atari, Go, Chess and Shogi by Planning With a Learned Model”. In: ().

[47] **Starre, R. A. N.**, M. Loog, E. Congeduti, and F. A. Oliehoek. “An Analysis of Model-Based Reinforcement Learning From Abstracted Observations”. In: *Transactions on Machine Learning Research* (2023).

[48] A. Hallak, D. Di-Castro, and S. Mannor. “Model Selection in Markovian Processes”. In: *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining*. 2013, pp. 374–382.

[49] O.-A. Maillard, P. Nguyen, R. Ortner, and D. Ryabko. “Optimal Regret Bounds for Selecting the State Representation in Reinforcement Learning”. In: *Proceedings of the 30th International Conference on International Conference on Machine Learning*. 2013, pp. 543–551.

[50] S. J. Majeed and M. Hutter. “On Q-Learning Convergence for Non-Markov Decision Processes”. In: *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence*. 2018, pp. 2546–2552.

[51] R. Ortner, M. Pirotta, A. Lazaric, R. Fruit, and O.-A. Maillard. “Regret Bounds for Learning State Representations in Reinforcement Learning”. In: *Advances in Neural Information Processing Systems*. 2019.

[52] S. Du, A. Krishnamurthy, N. Jiang, A. Agarwal, M. Dudik, and J. Langford. “Provably Efficient RL With Rich Observations Via Latent State Decoding”. In: *Proceedings of the 36th International Conference on International Conference on Machine Learning*. 2019, pp. 1665–1674.

[53] W. Hoeffding. “Probability Inequalities for Sums of Bounded Random Variables”. In: *Journal of the American Statistical Association* 58.301 (1963), pp. 13–30.

[54] K. Azuma. “Weighted Sums of Certain Dependent Random Variables”. In: *Tohoku Mathematical Journal, Second Series* (1967).

[55] **Starre, R. A. N.**, S. Katt, M. M. Çelikok, M. Loog, and F. A. Oliehoek. “Abstraction for Bayesian Reinforcement Learning in Factored POMDPs”. In: *Transactions on Machine Learning Research* (2025).

[56] R. Jonschkowski and O. Brock. “Learning State Representations With Robotic Priors”. In: *Autonomous Robots* (2015), pp. 407–428.

[57] T. De Bruin, J. Kober, K. Tuyls, and R. Babuška. “Integrating State Representation Learning Into Deep Reinforcement Learning”. In: *IEEE Robotics and Automation Letters* (2018), pp. 1394–1401.

[58] S. Katt, H. Nguyen, F. A. Oliehoek, and C. Amato. “Baddr: Bayes-Adaptive Deep Dropout RL for POMDPs”. In: *Proceedings of the 21st International Conference on Autonomous Agents and Multiagent Systems*. 2022, pp. 723–731.

[59] S. Ross, J. Pineau, B. Chaib-draa, and P. Kreitmann. “A Bayesian Approach for Learning and Planning in Partially Observable Markov Decision Processes”. In: *Journal of Machine Learning Research* 12.May (2011), pp. 1729–1770.

[60] S. Katt, F. A. Oliehoek, and C. Amato. “Learning in POMDPs With Monte Carlo Tree Search”. In: *Proceedings of the 34th International Conference on International Conference on Machine Learning*. PMLR. 2017, pp. 1819–1827.

[61] S. Katt, F. A. Oliehoek, and C. Amato. “Bayesian Reinforcement Learning in Factored POMDPs”. In: *Proceedings of the 18th International Conference on Autonomous Agents and Multiagent Systems*. International Foundation for Autonomous Agents and Multiagent Systems. 2019, pp. 7–15.

[62] M. Suau de Castro, E. Congeduti, **Starre, R.A.N.**, A. Czechowski, and F. Oliehoek. “Influence-Based Abstraction in Deep Reinforcement Learning”. In: *Proceedings of the AAMAS Workshop on Adaptive Learning Agents*. 2019.

[63] Y. Oren, **Starre, R. A. N.**, and F. A. Oliehoek. “Comparing Exploration Approaches in Deep Reinforcement Learning for Traffic Light Controls”. In: *Proceedings of the 34th Benelux Conference on Artificial Intelligence (BNAIC) and the 30th Belgian Dutch Conference on Machine Learning (Benelearn)*. 2022.

[64] H. R. TO and M. M. Barker. “White Paper European Transport Policy for 2010: Time to Decide”. In: *Commission of the European Communities Brussels* (2001).

[65] E. Van der Pol and F. A. Oliehoek. “Coordinated Deep Reinforcement Learners for Traffic Light Control”. In: *Proceedings of the NIPS’16 Workshop on Learning, Inference and Control of Multi-Agent Systems* 8 (2016), pp. 21–38.

[66] I. Osband, C. Blundell, A. Pritzel, and B. Van Roy. “Deep Exploration Via Bootstrapped Dqn”. In: *Advances in Neural Information Processing Systems*. 2016, pp. 4026–4034.

[67] I. Osband, J. Aslanides, and A. Cassirer. “Randomized Prior Functions for Deep Reinforcement Learning”. In: *Advances in Neural Information Processing Systems* 31 (2018).

[68] A. Wald. *Statistical Decision Functions*. John Wiley, 1950.

[69] T. M. Moerland, J. Broekens, and C. M. Jonker. “Model-Based Reinforcement Learning: a Survey”. In: *arXiv Preprint arXiv:2006.16712* (2020).

[70] A. J. Keith and D. K. Ahner. “A Survey of Decision Making and Optimization Under Uncertainty”. In: *Annals of Operations Research* 300.2 (2021), pp. 319–353.

[71] A. Plaat, W. Kosters, and M. Preuss. “High-Accuracy Model-Based Reinforcement Learning, a Survey”. In: *Artificial Intelligence Review* 56.9 (2023), pp. 9541–9573.

[72] T. Lesort, N. Díaz-Rodríguez, J.-F. Goudou, and D. Filliat. “State Representation Learning for Control: an Overview”. In: *Neural Networks* 108 (2018), pp. 379–392.

[73] N. Jie. “Representation Learning for Model-Based Reinforcement Learning: a Survey”. In: *\{ tinyurl.com/jierep }* (2021).

[74] M. L. Puterman. *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. John Wiley, 2014.

[75] R. Givan, T. Dean, and M. Greig. “Equivalence Notions and Model Minimization in Markov Decision Processes”. In: *Artificial Intelligence* (2003), pp. 163–223.

[76] T. Dean and R. Givan. “Model Minimization in Markov Decision Processes”. In: *Proceedings of the 14th AAAI Conference on Artificial Intelligence*. 1997.

[77] T. Dean, R. Givan, and S. Leach. “Model Reduction Techniques for Computing Approximately Optimal Solutions for Markov Decision Processes”. In: *Proceedings of the 13th Conference on Uncertainty in Artificial Intelligence*. 1997, pp. 124–131.

[78] E. Talvitie. “Model Regularization for Stable Sample Rollouts”. In: *Proceedings of the 30th Conference on Uncertainty in Artificial Intelligence*. 2014, pp. 780–789.

[79] W. Ye, S. Liu, T. Kurutach, P. Abbeel, and Y. Gao. “Mastering Atari Games With Limited Data”. In: *Advances in Neural Information Processing Systems* (2021).

[80] A. A. Taïga, A. Courville, and M. G. Bellemare. “Approximate Exploration Through State Abstraction”. In: *arXiv Preprint arXiv:1808.09819* (2018).

[81] A. K. McCallum. *Reinforcement Learning With Selective Perception and Hidden State*. University of Rochester, 1996.

[82] L. Li. “A Unifying Framework for Computational Reinforcement Learning Theory”. PhD thesis. Rutgers University-Graduate School-New Brunswick, 2009.

[83] M. Jaderberg, V. Mnih, W. M. Czarnecki, T. Schaul, J. Z. Leibo, D. Silver, and K. Kavukcuoglu. “Reinforcement Learning With Unsupervised Auxiliary Tasks”. In: *arXiv Preprint arXiv:1611.05397* (2016).

[84] Y. Wang and X. Tan. “Deep Recurrent Belief Propagation Network for POMDPs”. In: *Proceedings of the 35th AAAI Conference on Artificial Intelligence*. Vol. 35. 11. 2021, pp. 10236–10244.

[85] M. Igl, L. Zintgraf, T. A. Le, F. Wood, and S. Whiteson. “Deep Variational Reinforcement Learning for POMDPs”. In: *Proceedings of the 35th International Conference on International Conference on Machine Learning*. PMLR. 2018, pp. 2117–2126.

[86] S. Tschiatschek, K. Arulkumaran, J. Stühmer, and K. Hofmann. “Variational Inference for Data-Efficient Model Learning in POMDPs”. In: *arXiv Preprint arXiv:1805.09281* (2018).

[87] P. Karkus, D. Hsu, and W. S. Lee. “QMDP-Net: Deep Learning for Planning Under Partial Observability”. In: *Advances in Neural Information Processing Systems* (2017).

[88] X. Ma, P. Karkus, D. Hsu, W. S. Lee, and N. Ye. “Discriminative Particle Filter Reinforcement Learning for Complex Partial Observations”. In: *Proceedings of the 7th International Conference on Learning Representations*. 2019.

[89] M. Igl, G. Farquhar, J. Luketina, W. Boehmer, and S. Whiteson. “Transient Non-Stationarity and Generalisation in Deep Reinforcement Learning”. In: *Proceedings of the 9th International Conference on Learning Representations*. 2021.

[90] W. Wiesemann, D. Kuhn, and B. Rustem. “Robust Markov Decision Processes”. In: *Mathematics of Operations Research* (2013), pp. 153–183.

[91] R. Givan, S. Leach, and T. Dean. “Bounded-Parameter Markov Decision Processes”. In: *Artificial Intelligence* (2000), pp. 71–109.

[92] M. Suilen, T. Badings, E. M. Bovy, D. Parker, and N. Jansen. “Robust Markov Decision Processes: A Place Where AI and Formal Methods Meet”. In: *Principles of Verification: Cycling the Probabilistic Landscape: Essays Dedicated to Joost-Pieter Katoen on the Occasion of His 60th Birthday, Part III*. Springer, 2024, pp. 126–154.

[93] M. Petrik and D. Subramanian. “Raam: the Benefits of Robustness in Approximating Aggregated MDPs in Reinforcement Learning”. In: *Advances in Neural Information Processing Systems* (2014).

[94] S. Junges, J.-P. Katoen, G. A. Pérez, and T. Winkler. “The Complexity of Reachability in Parametric Markov Decision Processes”. In: *Journal of Computer and System Sciences* 119 (2021), pp. 183–210.

[95] V. Goyal and J. Grand-Clement. “Robust Markov Decision Process: Beyond Rectangularity”. In: *arXiv Preprint arXiv:1811.00215* (2018).

[96] S. Mannor, O. Mebel, and H. Xu. “Robust MDPs With K-Rectangular Uncertainty”. In: *Mathematics of Operations Research* (2016).

[97] S. H. Lim and A. Autef. “Kernel-Based Reinforcement Learning in Robust Markov Decision Processes”. In: *Proceedings of the 36th International Conference on International Conference on Machine Learning*. 2019, pp. 3973–3981.

[98] M. Hutter. “Extreme State Aggregation Beyond Markov Decision Processes”. In: *Theoretical Computer Science* 650 (2016), pp. 73–91.

[99] V. François-Lavet, G. Rabusseau, J. Pineau, D. Ernst, and R. Fonteneau. "On Overfitting and Asymptotic Bias in Batch Reinforcement Learning With Partial Observability". In: *Journal of Artificial Intelligence Research* (2019), pp. 1–30.

[100] I. V. Serban, C. Sankar, M. Pieper, J. Pineau, and Y. Bengio. "The Bottleneck Simulator: a Model-Based Deep Reinforcement Learning Approach". In: *Journal of Artificial Intelligence Research* (2020), pp. 571–612.

[101] Y. S. Chow and H. Teicher. *Probability Theory: Independence, Interchangeability, Martingales*. Springer Science & Business Media, 2003.

[102] O.-A. Maillard, D. Ryabko, and R. Munos. "Selecting the State-Representation in Reinforcement Learning". In: *Advances in Neural Information Processing Systems* (2011).

[103] P. Nguyen, O.-A. Maillard, D. Ryabko, and R. Ortner. "Competing With an Infinite Set of Models in Reinforcement Learning". In: *Proceedings of the 16th International Conference on Artificial Intelligence and Statistics*. 2013, pp. 463–471.

[104] T. Lattimore, M. Hutter, and P. Sunehag. "The Sample-Complexity of General Reinforcement Learning". In: *Proceedings of the 30th International Conference on International Conference on Machine Learning*. 2013, pp. 28–36.

[105] R. Fruit, M. Pirotta, and A. Lazaric. "Near Optimal Exploration-Exploitation in Non-Communicating Markov Decision Processes". In: *Advances in Neural Information Processing Systems*. 2018.

[106] L. Kuvayev and R. S. Sutton. "Model-Based Reinforcement Learning With an Approximate, Learned Model". In: *Proceedings of the Yale Workshop on Adaptive and Learning Systems*. 1996.

[107] A. Bernstein and N. Shimkin. "Adaptive-Resolution Reinforcement Learning With Polynomial Exploration in Deterministic Domains". In: *Machine Learning* 81 (2010), pp. 359–397.

[108] T. Mandel, Y.-E. Liu, E. Brunskill, and Z. Popovic. "Efficient Bayesian Clustering for Reinforcement Learning". In: *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence*. 2016, pp. 1830–1838.

[109] K. Azizzadenesheli, A. Lazaric, and A. Anandkumar. "Reinforcement Learning in Rich-Observation MDPs Using Spectral Methods". In: *Proceedings of the 29th Conference on Learning Theory* (2016), pp. 193–256.

[110] Z. D. Guo, S. Doroudi, and E. Brunskill. "A PAC RL Algorithm for Episodic POMDPs". In: *Proceedings of the 18th International Conference on Artificial Intelligence and Statistics*. 2016, pp. 510–518.

[111] A. Krishnamurthy, A. Agarwal, and J. Langford. "PAC Reinforcement Learning With Rich Observations". In: *Advances in Neural Information Processing Systems* (2016).

[112] A. Zhang, C. Lyle, S. Sodhani, A. Filos, M. Kwiatkowska, J. Pineau, Y. Gal, and D. Precup. "Invariant Causal Prediction for Block MDPs". In: *Proceedings of the 37th International Conference on International Conference on Machine Learning*. 2020, pp. 11214–11224.

[113] P. Sermanet, C. Lynch, Y. Chebotar, J. Hsu, E. Jang, S. Schaal, S. Levine, and G. Brain. “Time-Contrastive Networks: Self-Supervised Learning From Video”. In: *Proceedings of the 2018 IEEE International Conference on Robotics and Automation*. 2018, pp. 1134–1141.

[114] V. Thomas, E. Bengio, W. Fedus, J. Pondard, P. Beaudoin, H. Larochelle, J. Pineau, D. Precup, and Y. Bengio. “Disentangling the Independently Controllable Factors of Variation by Interacting With the World”. In: *arXiv Preprint arXiv:1802.09484* (2018).

[115] O. Biza and R. Platt. “Online Abstraction With MDP Homomorphisms for Deep Learning”. In: *arXiv Preprint arXiv:1811.12929* (2018).

[116] V. François-Lavet, Y. Bengio, D. Precup, and J. Pineau. “Combined Reinforcement Learning Via Abstract Representations”. In: *Proceedings of the 33rd AAAI Conference on Artificial Intelligence*. 2019, pp. 3582–3589.

[117] C. Gelada, S. Kumar, J. Buckman, O. Nachum, and M. G. Bellemare. “Deepmdp: Learning Continuous Latent Space Models for Representation Learning”. In: *Proceedings of the 36th International Conference on International Conference on Machine Learning*. 2019, pp. 2170–2179.

[118] A. Anand, E. Racah, S. Ozair, Y. Bengio, M.-A. Côté, and R. D. Hjelm. “Unsupervised State Representation Learning in Atari”. In: *Advances in Neural Information Processing Systems*. 2019.

[119] T. Kipf, E. van der Pol, and M. Welling. “Contrastive Learning of Structured World Models”. In: *Proceedings of the 9th International Conference on Learning Representations*. 2019.

[120] A. Van den Oord, Y. Li, and O. Vinyals. “Representation Learning With Contrastive Predictive Coding”. In: *arXiv Preprint arXiv:1807.03748* (2018).

[121] T. Lesort, M. Seurin, X. Li, N. Díaz-Rodríguez, and D. Filliat. “Deep Unsupervised State Representation Learning With Robotic Priors: a Robustness Analysis”. In: *Proceedings of the 2019 International Joint Conference on Neural Networks*. 2019, pp. 1–8.

[122] H. Van Hoof, N. Chen, M. Karl, P. van der Smagt, and J. Peters. “Stable Reinforcement Learning With Autoencoders for Tactile and Visual Data”. In: *Proceedings of the 2016 IEEE/RSJ International Conference on Intelligent Robots and Systems*. 2016, pp. 3928–3934.

[123] M. Watter, J. T. Springenberg, J. Boedecker, and M. Riedmiller. “Embed to Control: a Locally Linear Latent Dynamics Model for Control From Raw Images”. In: *Advances in Neural Information Processing Systems*. 2015.

[124] M. Zhang, S. Vikram, L. Smith, P. Abbeel, M. Johnson, and S. Levine. “Solar: Deep Structured Representations for Model-Based Reinforcement Learning”. In: *Proceedings of the 36th International Conference on International Conference on Machine Learning*. 2019, pp. 7444–7453.

[125] G. Konidaris. “On the Necessity of Abstraction”. In: *Current Opinion in Behavioral Sciences* (2019).

[126] P. Hernandez-Leal, M. Kaisers, T. Baarslag, and E. M. de Cote. "A Survey of Learning in Multiagent Environments: Dealing With Non-Stationarity". In: *arXiv Preprint arXiv:1707.09183* (2017).

[127] R. Fruit, M. Pirotta, A. Lazaric, and R. Ortner. "Efficient Bias-Span-Constrained Exploration-Exploitation in Reinforcement Learning". In: *International Conference on Machine Learning*. PMLR. 2018, pp. 1578–1586.

[128] M. S. Talebi and O.-A. Maillard. "Variance-Aware Regret Bounds for Undiscounted Reinforcement Learning in MDPs". In: *Algorithmic Learning Theory*. 2018.

[129] Z. Zhang and X. Ji. "Regret Minimization for Reinforcement Learning by Evaluating the Optimal Bias Function". In: *Advances in Neural Information Processing Systems* (2019).

[130] H. Bourel, O. Maillard, and M. S. Talebi. "Tightening Exploration in Upper Confidence Reinforcement Learning". In: *International Conference on Machine Learning*. 2020.

[131] S. Mahadevan. "Representation Discovery in Sequential Decision Making". In: *Proceedings of the 24th AAAI Conference on Artificial Intelligence*. 2010, pp. 1718–1721.

[132] S. P. Singh, T. Jaakkola, and M. I. Jordan. "Learning Without State-Estimation in Partially Observable Markovian Decision Processes". In: *Machine Learning Proceedings*. Elsevier, 1994, pp. 284–292.

[133] T. Weissman, E. Ordentlich, G. Seroussi, S. Verdu, and M. J. Weinberger. "Inequalities for the L1 Deviation of the Empirical Distribution". In: *Hewlett-Packard Labs, Technical Report* (2003).

[134] T. Dean, R. Givan, and S. Leach. "Model Reduction Techniques for Computing Approximately Optimal Solutions for Markov Decision Processes". In: *Proceedings of the Thirteenth Conference on Uncertainty in Artificial Intelligence*. 1997, pp. 124–131.

[135] R. Dearden and C. Boutilier. "Abstraction and Approximate Decision-Theoretic Planning". In: *Artificial Intelligence* 89.1-2 (1997), pp. 219–283.

[136] M. Kearns and S. Singh. "Near-Optimal Reinforcement Learning in Polynomial Time". In: *Machine Learning* 49.2 (2002), pp. 209–232.

[137] R. Ortner, D. Ryabko, P. Auer, and R. Munos. "Regret Bounds for Restless Markov Bandits". In: *Theoretical Computer Science* 558 (2014), pp. 62–76.

[138] R. Ortner, P. Gajane, and P. Auer. "Variational Regret Bounds for Reinforcement Learning". In: *Uncertainty in Artificial Intelligence*. PMLR. 2020, pp. 81–90.

[139] S. R. Sinclair, S. Banerjee, and C. L. Yu. "Adaptive Discretization in Online Reinforcement Learning". In: *Operations Research* (2022).

[140] J.-Y. Audibert, R. Munos, and C. Szepesvári. "Tuning Bandit Algorithms in Stochastic Environments". In: *International Conference on Algorithmic Learning Theory*. Springer. 2007, pp. 150–165.

[141] A. Maurer and M. Pontil. “Empirical Bernstein Bounds and Sample Variance Penalization”. In: *arXiv Preprint arXiv:0907.3740* (2009).

[142] K. Dzhaparidze and J. Van Zanten. “On Bernstein-Type Inequalities for Martingales”. In: *Stochastic Processes and Their Applications* 93.1 (2001), pp. 109–117.

[143] O. Biza and R. Platt. “Online Abstraction With MDP Homomorphisms for Deep Learning”. In: *Proceedings of the 18th International Conference on Autonomous Agents and Multiagent Systems*. 2019.

[144] G. Boole. *An Investigation of the Laws of Thought: on Which Are Founded the Mathematical Theories of Logic and Probabilities*. Dover Publications, 1854.

[145] D. A. Levin and Y. Peres. *Markov Chains and Mixing Times*. Vol. 107. American Mathematical Society, 2017.

[146] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. Van Den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot, *et al.* “Mastering the Game of Go With Deep Neural Networks and Tree Search”. In: *Nature* 529.7587 (2016), pp. 484–489.

[147] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine. “Soft Actor-Critic: Off-Policy Maximum Entropy Deep Reinforcement Learning With a Stochastic Actor”. In: *International Conference on Machine Learning*. PMLR. 2018, pp. 1861–1870.

[148] J. He, M. Suau de Castro, and F. Oliehoek. “Influence-Augmented Online Planning for Complex Environments”. In: *Advances in Neural Information Processing Systems* 33 (2020).

[149] R. Chitnis, T. Silver, B. Kim, L. Kaelbling, and T. Lozano-Perez. “Camps: Learning context-specific abstractions for efficient planning in factored mdps”. In: *Conference on Robot Learning*. PMLR. 2021, pp. 64–79.

[150] C. Boutilier and D. Poole. “Computing Optimal Policies for Partially Observable Decision Processes Using Compact Representations”. In: *Proceedings of the 13th AAAI Conference on Artificial Intelligence*. Citeseer. 1996, pp. 1168–1175.

[151] C. Boutilier, T. Dean, and S. Hanks. “Decision-Theoretic Planning: Structural Assumptions and Computational Leverage”. In: *Journal of Artificial Intelligence Research* 11 (1999), pp. 1–94.

[152] K. P. Murphy. *Dynamic Bayesian Networks: Representation, Inference and Learning*. University of California, Berkeley, 2002.

[153] B. A. Frigyik, A. Kapila, and M. R. Gupta. “Introduction to the Dirichlet Distribution and Related Processes”. In: *Uwee Technical Report* $uweetr-2010-0006$ (2010).

[154] D. Silver and J. Veness. “Monte-Carlo Planning in Large POMDPs”. In: *Advances in Neural Information Processing Systems*. 2010.

[155] S. Thrun. “Monte Carlo POMDPs”. In: *Advances in Neural Information Processing Systems*. Vol. 12. 1999, pp. 1064–1070.

[156] L. Kocsis and C. Szepesvári. “Bandit Based Monte-Carlo Planning”. In: *European Conference on Machine Learning*. Springer. 2006, pp. 282–293.

[157] C. B. Browne, E. Powley, D. Whitehouse, S. M. Lucas, P. I. Cowling, P. Rohlfshagen, S. Tavener, D. Perez, S. Samothrakis, S. Colton, *et al.* "A Survey of Monte Carlo Tree Search Methods". In: *IEEE Transactions on Computational Intelligence and AI in Games* 4.1 (2012), pp. 1–43.

[158] E. Congeduti and F. A. Oliehoek. "A Cross-Field Review of State Abstraction for Markov Decision Processes". In: *Proceedings of the 34th Benelux Conference on Artificial Intelligence (BNAIC) and the 30th Belgian Dutch Conference on Machine Learning (Benelearn)*. 2022.

[159] J. Lee, A. Agarwal, C. Dann, and T. Zhang. "Learning in POMDPs Is Sample-Efficient With Hindsight Observability". In: *Proceedings of the 40th International Conference on International Conference on Machine Learning*. PMLR. 2023, pp. 18733–18773.

[160] Y. Luo, H. Bai, D. Hsu, and W. S. Lee. "Importance Sampling for Online Planning Under Uncertainty". In: *The International Journal of Robotics Research* 38.2-3 (2019), pp. 162–181.

[161] F. Doshi-Velez. "The Infinite Partially Observable Markov Decision Process". In: *Advances in Neural Information Processing Systems* (2009), pp. 477–485.

[162] P. Dallaire, C. Besse, S. Ross, and B. Chaib-Draa. "Bayesian Reinforcement Learning in Continuous POMDPs With Gaussian Processes". In: *Proceedings of the 2009 IEEE/RSJ International Conference on Intelligent Robots and Systems*. 2009, pp. 2604–2609.

[163] M. Grzegorczyk and D. Husmeier. "Non-Homogeneous Dynamic Bayesian Networks for Continuous Data". In: *Machine Learning* 83 (2011), pp. 355–419.

[164] P. Poupart, N. Vlassis, J. Hoey, and K. Regan. "An Analytic Solution to Discrete Bayesian Reinforcement Learning". In: *Proceedings of the 23rd International Conference on Machine Learning*. 2006, pp. 697–704.

[165] S. Ross and J. Pineau. "Model-Based Bayesian Reinforcement Learning in Large Structured Domains". In: *Proceedings of the 24th Conference on Uncertainty in Artificial Intelligence*. Vol. 2008. NIH Public Access. 2008, pp. 476–483.

[166] N. A. Vien, W. Ertel, V.-H. Dang, and T. Chung. "Monte-Carlo Tree Search for Bayesian Reinforcement Learning". In: *Applied Intelligence* 39.2 (2013), pp. 345–353.

[167] T. Hoang and N. A. Vien. *Bayes-Adaptive Deep Model-Based Policy Optimisation*. 2020.

[168] L. Zintgraf, S. Schulze, C. Lu, L. Feng, M. Igl, K. Shiarlis, Y. Gal, K. Hofmann, and S. Whiteson. "Varibad: Variational bayes-adaptive deep rl via meta-learning". In: *Journal of Machine Learning Research* 22.289 (2021), pp. 1–39.

[169] L. T. Dung, T. Komeda, and M. Takagi. "Reinforcement Learning for POMDP Using State Classification". In: *Applied Artificial Intelligence* 22.7-8 (2008), pp. 761–779.

[170] M. Hausknecht and P. Stone. "Deep Recurrent Q-Learning for Partially Observable MDPs". In: *AAAI Fall Symposium Series*. 2015.

[171] P. Zhu, X. Li, P. Poupart, and G. Miao. “On Improving Deep Reinforcement Learning for POMDPs”. In: *arXiv Preprint ArXiv:1704.07978* (2017).

[172] L. Meng, R. Gorbet, and D. Kulić. “Memory-Based Deep Reinforcement Learning for POMDPs”. In: *Proceedings of the 2021 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE. 2021, pp. 5619–5626.

[173] G. Shani, R. I. Brafman, and S. E. Shimony. “Model-Based Online Learning of POMDPs”. In: *Machine Learning: Ecml 2005: 16th European Conference on Machine Learning, Porto, Portugal, October 3-7, 2005. Proceedings 16*. Springer. 2005, pp. 353–364.

[174] Y. Liu and J. Zheng. “Online Learning and Planning in Partially Observable Domains Without Prior Knowledge”. In: *Proceedings of the ICML 2019 Workshop on the Generative Modeling and Model-Based Reasoning for Robotics and AI* (2019).

[175] A. Bennett and N. Kallus. “Proximal Reinforcement Learning: Efficient Off-Policy Evaluation in Partially Observed Markov Decision Processes”. In: *Operations Research* 72.3 (2024), pp. 1071–1086.

[176] J. Hostetler, A. Fern, and T. Dietterich. “State Aggregation in Monte Carlo Tree Search”. In: *Proceedings of the 28th AAAI Conference on Artificial Intelligence*. 2014, pp. 2446–2452.

[177] A. Anand, R. Noothigattu, and P. Singla. “OGA-UCT: On-the-Go Abstractions in UCT”. In: *Proceedings of the International Conference on Automated Planning and Scheduling*. Vol. 26. 2016.

[178] M. Mutti, R. D. Santi, M. Restelli, A. Marx, and G. Ramponi. “Exploiting Causal Graph Priors with Posterior Sampling for Reinforcement Learning”. In: *Proceedings of the 12th International Conference on Learning Representations*. 2024. URL: https://openreview.net/forum?id=M0xK8nPGvt.

[179] J. Zhang, J. Wang, H. Hu, Y. Chen, C. Fan, and C. Zhang. “Learn to Effectively Explore in Context-Based Meta-RL”. In: *arXiv Preprint arXiv:2006.08170* (2020).

[180] L. Ambrogioni. “Knowledge Is Reward: Learning Optimal Exploration by Predictive Reward Cashing”. In: *arXiv Preprint arXiv:2109.08518* (2021).

[181] E. Z. Liu, A. Raghunathan, P. Liang, and C. Finn. “Decoupling Exploration and Exploitation for Meta-Reinforcement Learning Without Sacrifices”. In: *International Conference on Machine Learning*. PMLR. 2021, pp. 6925–6935.

[182] D. Grover, D. Basu, and C. Dimitrakakis. “Bayesian Reinforcement Learning Via Deep, Sparse Sampling”. In: *International Conference on Artificial Intelligence and Statistics*. PMLR. 2020, pp. 3036–3045.

[183] A. Garivier and E. Moulines. “On Upper-Confidence Bound Policies for Switching Bandit Problems”. In: *International Conference on Algorithmic Learning Theory*. Springer. 2011, pp. 174–188.

[184] W. C. Cheung, D. Simchi-Levi, and R. Zhu. “Learning to Optimize Under Non-Stationarity”. In: *The 22nd International Conference on Artificial Intelligence and Statistics*. PMLR. 2019, pp. 1079–1087.

[185] F. Trovo, S. Paladino, M. Restelli, and N. Gatti. "Sliding-Window Thompson Sampling for Non-Stationary Settings". In: *Journal of Artificial Intelligence Research* 68 (2020), pp. 311–364.

[186] W. R. Thompson. "On the Likelihood that One Unknown Probability Exceeds Another in View of The Evidence of Two Samples". In: *Biometrika* 25.3/4 (1933), pp. 285–294.

[187] A. Bai, F. Wu, Z. Zhang, and X. Chen. "Thompson Sampling Based Monte-Carlo Planning in POMDPs". In: *Proceedings of the International Conference on Automated Planning and Scheduling*. Vol. 24. 2014, pp. 29–37.

[188] K. P. Murphy. *Machine Learning: a Probabilistic Perspective*. MIT press, 2012.

[189] M. Hessel, J. Modayil, H. Van Hasselt, T. Schaul, G. Ostrovski, W. Dabney, D. Horgan, B. Piot, M. Azar, and D. Silver. "Rainbow: Combining Improvements in Deep Reinforcement Learning". In: *Proceedings of the 32nd AAAI Conference on Artificial Intelligence*. Vol. 32. 1. 2018.

[190] N. Rudin, D. Hoeller, P. Reist, and M. Hutter. "Learning to Walk in Minutes Using Massively Parallel Deep Reinforcement Learning". In: *Conference on Robot Learning*. PMLR. 2022, pp. 91–100.

[191] M. Bellemare, S. Srinivasan, G. Ostrovski, T. Schaul, D. Saxton, and R. Munos. "Unifying Count-Based Exploration and Intrinsic Motivation". In: *Advances in Neural Information Processing Systems*. 2016, pp. 1471–1479.

[192] G. Ostrovski, M. G. Bellemare, A. van den Oord, and R. Munos. "Count-Based Exploration With Neural Density Models". In: *Proceedings of the 34th International Conference on Machine Learning - volume 70*. JMLR. org. 2017, pp. 2721–2730.

[193] D. Pathak, P. Agrawal, A. A. Efros, and T. Darrell. "Curiosity-Driven Exploration by Self-Supervised Prediction". In: *Proceedings of the 34th International Conference on International Conference on Machine Learning*. Vol. 2017. 2017, pp. 2778–2787.

[194] H. Tang, R. Houthooft, D. Foote, A. Stooke, X. Chen, Y. Duan, J. Schulman, F. DeTurck, and P. Abbeel. "# Exploration: a Study of Count-based Exploration for Deep Reinforcement Learning". In: *Advances in Neural Information Processing Systems*. 2017, pp. 2753–2762.

[195] K. Azizzadenesheli, E. Brunskill, and A. Anandkumar. "Efficient Exploration Through Bayesian Deep Q-Networks". In: *Proceedings of the Information Theory and Applications Workshop*. IEEE. 2018, pp. 1–9.

[196] M. C. Machado, M. G. Bellemare, and M. Bowling. "Count-Based Exploration With the Successor Representation". In: *arXiv Preprint arXiv:1807.11622* (2018).

[197] J. Hao, T. Yang, H. Tang, C. Bai, J. Liu, Z. Meng, P. Liu, and Z. Wang. "Exploration in Deep Reinforcement Learning: From Single-Agent to Multiagent Domain". In: *IEEE Transactions on Neural Networks and Learning Systems* 35.7 (2023), pp. 8762–8782.

[198] T. Yu, A. Kumar, R. Rafailov, A. Rajeswaran, S. Levine, and C. Finn. "COMBO: Conservative Offline Model-Based Policy Optimization". In: *Advances in Neural Information Processing Systems* 34 (2021), pp. 28954–28967.

[199] C. Lu, P. Ball, J. Parker-Holder, M. Osborne, and S. J. Roberts. “Revisiting Design Choices in Offline Model Based Reinforcement Learning”. In: *International Conference on Learning Representations*. 2022.

[200] M. Rigter, B. Lacerda, and N. Hawes. “RAMBO-RL: Robust Adversarial Model-Based Offline Reinforcement Learning”. In: *Advances in Neural Information Processing Systems* 35 (2022), pp. 16082–16097.

[201] M. Uehara and W. Sun. “Pessimistic Model-based Offline Reinforcement Learning under Partial Coverage”. In: *International Conference on Learning Representations*. 2022.

[202] F. Delgrange, A. Nowé, and G. A. Pérez. “Distillation of RL Policies with Formal Guarantees via Variational Abstraction of Markov Decision Processes”. In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 36. 6. 2022, pp. 6497–6505.

[203] F. Delgrange, A. Nowe, and G. A. Pérez. “Wasserstein Auto-encoded MDPs: Formal Verification of Efficiently Distilled RL Policies with Many-sided Guarantees”. In: *The Eleventh International Conference on Learning Representations: ICLR 2023*. 2023, pp. 1–30.

[204] R. Avalos, F. Delgrange, A. Nowe, G. Perez, and D. M. Roijers. “The Wasserstein Believer: Learning Belief Updates for Partially Observable Environments through Reliable Latent Space Models”. In: *The Twelfth International Conference on Learning Representations*. 2024.

[205] M. D. Hoffman, D. M. Blei, C. Wang, and J. Paisley. “Stochastic Variational Inference”. In: *Journal of Machine Learning Research* 14.1 (2013), pp. 1303–1347.

[206] D. P. Kingma and M. Welling. “Auto-Encoding Variational Bayes”. In: (2014).

[207] I. Tolstikhin, O. Bousquet, S. Gelly, and B. Schoelkopf. “Wasserstein Auto-Encoders”. In: *International Conference on Learning Representations*. 2018.

[208] D. Ha and J. Schmidhuber. “World Models”. In: *arXiv Preprint arXiv:1803.10122* (2018).

[209] D. Hafner, T. Lillicrap, I. Fischer, R. Villegas, D. Ha, H. Lee, and J. Davidson. “Learning Latent Dynamics for Planning From Pixels”. In: *International Conference on Machine Learning*. PMLR. 2019, pp. 2555–2565.

[210] D. Hafner, T. Lillicrap, J. Ba, and M. Norouzi. “Dream to Control: Learning Behaviors by Latent Imagination”. In: *Proceedings of the 10th International Conference on Learning Representations*. 2020.

[211] J. Wu, Z. Huang, and C. Lv. “Uncertainty-Aware Model-Based Reinforcement Learning: Methodology and Application in Autonomous Driving”. In: *IEEE Transactions on Intelligent Vehicles* 8.1 (2022), pp. 194–203.

[212] R. R. Hossain, T. Yin, Y. Du, R. Huang, J. Tan, W. Yu, Y. Liu, and Q. Huang. “Efficient Learning of Power Grid Voltage Control Strategies via Model-Based Deep Reinforcement Learning”. In: *Machine Learning* 113.5 (2024), pp. 2675–2700.

[213] K. Jothimurugan, O. Bastani, and R. Alur. “Abstract Value Iteration for Hierarchical Reinforcement Learning”. In: *International Conference on Artificial Intelligence and Statistics*. PMLR. 2021, pp. 1162–1170.

[214] R. Bommasani, D. A. Hudson, E. Adeli, R. Altman, S. Arora, S. von Arx, M. S. Bernstein, J. Bohg, A. Bosselut, E. Brunskill, *et al.* “On the Opportunities and Risks of Foundation Models”. In: *arXiv Preprint arXiv:2108.07258* (2021).

[215] OpenAI, J. Achiam, S. Adler, S. Agarwal, L. Ahmad, I. Akkaya, F. L. Aleman, D. Almeida, J. Altenschmidt, S. Altman, *et al.* “GPT-4 Technical Report”. In: *arXiv Preprint arXiv:2303.08774* (2023).

[216] W. Huang, P. Abbeel, D. Pathak, and I. Mordatch. “Language Models as Zero-Shot Planners: Extracting Actionable Knowledge for Embodied Agents”. In: *International Conference on Machine Learning*. PMLR. 2022, pp. 9118–9147.

[217] R. Ma, J. Luijkx, Z. Ajanović, and J. Kober. “ExploRLLM: Guiding Exploration in Reinforcement Learning with Large Language Models”. In: *2025 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2025, pp. 9011–9017.

[218] K. Rana, V. Dasagi, J. Haviland, B. Talbot, M. Milford, and N. Sünderhauf. “Bayesian Controller Fusion: Leveraging Control Priors in Deep Reinforcement Learning for Robotics”. In: *The International Journal of Robotics Research* 42.3 (2023), pp. 123–146.

[219] P. Auer, N. Cesa-Bianchi, and P. Fischer. “Finite-Time Analysis of the Multiarmed Bandit Problem”. In: *Machine Learning* 47.2-3 (2002), pp. 235–256.

[220] L. Wei and V. Srivatsva. “On Abruptly-Changing and Slowly-Varying Multiarmed Bandit Problems”. In: *2018 Annual American Control Conference acc*. IEEE. 2018, pp. 6291–6296.

ACKNOWLEDGEMENTS

I would like to express my deepest gratitude to everyone who has supported me and made this PhD journey such a meaningful and enjoyable experience.

I am deeply grateful to my supervisors, who guided and supported me throughout this journey. To Frans: thank you for our many discussions, even before this PhD began, back when I was working on my master's thesis. You sparked my interest in pursuing a PhD and encouraged me to carve out my own direction in this field. To Catholijn: thank you for your insightful perspectives and guidance, which helped me navigate the academic world. You made me feel welcome in the group and helped me grow with confidence. To Marco: thank you for guiding me already during my master's thesis and continuing to support me throughout my PhD. I always appreciated your willingness to listen, your advice, and the beers shared at the Burgemeester and elsewhere. Working with you has been both valuable and enjoyable, and I am grateful for it.

This dissertation was carried out as part of the Influence project. I would like to give special thanks to Aleksander, Alexander, Elena, Jinke, and Miguel. Sharing an office with you, along with our many discussions and trips to Den Bosch, Lille, Montreal, and beyond, made this journey all the more memorable and enjoyable. Thank you, Elena, for your mathematical support and the laughs, and Aleks, for your insights and our regular bouldering trips.

Thank you to everyone at Interactive Intelligence: Amir, Bernd, Carolina, Catharine, Ding, Elie, Enrico, Emma, Fran, Frank, Ilir, Jasper, Luciano, Malte, Mani, Masha, Merijn, Michiel, Mike, Mohammed, Morita, Myrte, Nele, Pei-Yu, Pradeep, Rijk, Ruben, Siddharth, Sietze, Thomas, Vincent, Willem-Paul, Wouter, and Zuzanna. The chats, board game nights, drinks, dinners, ping pong matches, cooking, and countless other events have been a blast! Thank you, Anita, Ruud, and Bart, for always providing support. Special thanks to Ding for being both an academic big brother and a worthy ping pong rival. To Ilirthank you for the hangouts, your job-hunting advice, and for always being willing to help. And to Carolina: thank you for the coffee chats, for sharing perspectives, and for the fantastic tips about Japan.

At the RL group, thank you to Canmanie, Davide, Greg, Matthijs, Mert, Oussama, Qisong, Robert, Thiago, Wendelin, and Yaniv for our engaging meetings and discussions. A very warm thank you to Sammie for your invaluable help on the Bayesian Adaptive project, your patience with my many questions, and our friendly chats. And to Mert, for our stimulating discussions and your support.

Beyond academia, I owe much to my friends and family. Bedankt aan Wybren en Kati, voor alle vriendschap door de jaren heen en de vele leuke momenten samen, er is altijd nog one next turn. Frank, dank voor het meebetrekken bij het fietsen en de pooltjes, het sporten en puzzelen was vaak een welkome afleiding. Laura en Susan, mijn lieve zussen, en Mark en Simon: bedankt voor jullie steun en gezelligheid, ik weet dat ik altijd

op jullie kan rekenen. De vele escaperooms, spelletjes en evenementen zoals Sinterklaas zorgen altijd voor veel plezier, vooral als er een slacentrifuge bij betrokken is! Papa, bedankt voor je luisterende oor, je advies en dat je er altijd voor me bent. Onze activiteiten samen, zoals boulderen of een wandeling, zijn altijd bijzonder fijne momenten. Mama, bedankt voor je steun en voor het vertrouwen dat je me altijd hebt gegeven dat ik alles kan bereiken wat ik wil. Ik ben altijd welkom bij je en je staat altijd klaar voor een spelletje. Beiden hebben jullie me altijd gesteund en vertrouwen gegeven, wat wil een zoon nog meer?

A mi familia Peruana, la familia Arévalo: muchísimas gracias por acogerme como a uno de los suyos. Ustedes representan de verdad el espíritu de la familia y la alegría. ¡Que vengan muchas más chelas juntos!

Y finalmente, a mi novia, ahora mi esposa: cuando empecé este camino, jamás habría imaginado que me casaría antes de terminar el doctorado, pero aquí estamos. Estos últimos años contigo han sido maravillosos. Contigo he aprendido tanto, y tu amor y apoyo significan el mundo para mí. Nuestras aventuras juntos han sido increíbles, desde Perú hasta Japón y todo lo que hay en medio. ¡Que sigan muchas más aventuras juntos!

CURRICULUM VITÆ

Rolf Andries Nicodemo STARRE

10-08-1989 Born in Leiden, Netherlands.

EDUCATION

2018–2025 PhD Computer Science

Delft University of Technology, Netherlands

Thesis: Lost in Abstraction: Exploring Our Way to Efficient Reinforcement Learning
Promotor: Prof. dr. F.A. Oliehoek
Promotor: Prof. dr. M. Loog

2016–2018 Master in Computer Science

Delft University of Technology, Netherlands

2013–2016 Bachelor in Computer Science

Delft University of Technology, Netherlands

2016–2018 Master in Human Movement Sciences

Vrije Universiteit Amsterdam, Netherlands

2013–2016 Bachelor in Human Movement Sciences

Vrije Universiteit Amsterdam, Netherlands

List of Publications

6. **Starre, R. A. N.**, S. Katt, M. M. Çelikok, M. Loog, and F. A. Oliehoek. "Abstraction for Bayesian Reinforcement Learning in Factored POMDPs". In: *Transactions on Machine Learning Research* (2025)

5. **Starre, R. A. N.**, M. Loog, E. Congeduti, and F. A. Oliehoek. "An Analysis of Model-Based Reinforcement Learning From Abstracted Observations". In: *Transactions on Machine Learning Research* (2023)

4. **Starre, R. A. N.**, M. Loog, and F. A. Oliehoek. "Model-Based Reinforcement Learning With State Abstraction: a Survey". In: *Proceedings of the 34th Benelux Conference on Artificial Intelligence (BNAIC) and the 30th Belgian Dutch Conference on Machine Learning (Benelearn), Revised Selected Papers*. Communications in Computer and Information Science. Springer, 2023, pp. 133–148

3. M. Suau, J. He, E. Congeduti, **Starre, R. A. N.**, A. Czechowski, and F. A. Oliehoek. "Influence-Aware Memory Architectures for Deep Reinforcement Learning in POMDPs". In: *Neural Computing and Applications* (2022), pp. 1–17

2. Y. Oren, **Starre, R. A. N.**, and F. A. Oliehoek. "Comparing Exploration Approaches in Deep Reinforcement Learning for Traffic Light Controls". In: *Proceedings of the 34th Benelux Conference on Artificial Intelligence (BNAIC) and the 30th Belgian Dutch Conference on Machine Learning (Benelearn)*. 2022

1. M. Suau de Castro, E. Congeduti, **Starre, R.A.N.**, A. Czechowski, and F. Oliehoek. "Influence-Based Abstraction in Deep Reinforcement Learning". In: *Proceedings of the AAMAS Workshop on Adaptive Learning Agents*. 2019