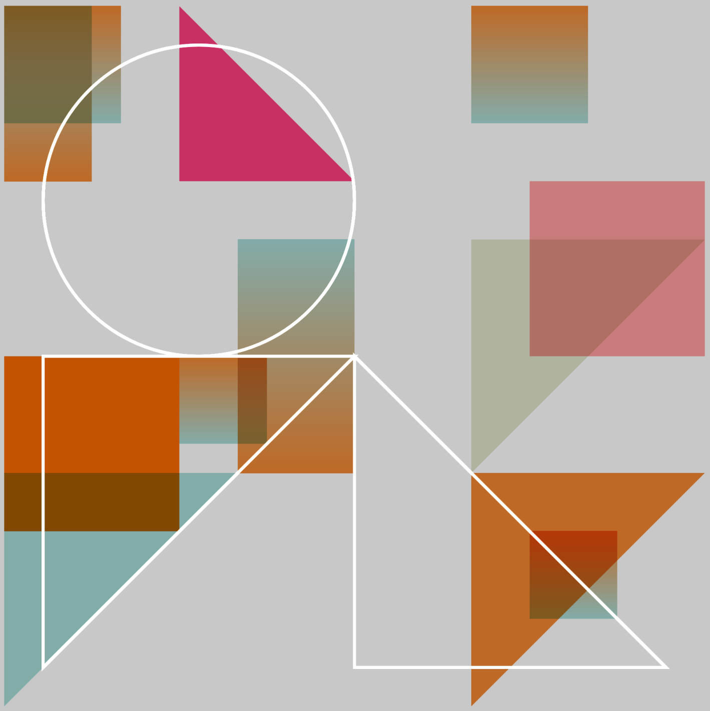
SAFE ONLINE and OFFLINE REINFORCEMENT LEARNING



Thiago D. Simão

Safe Online and Offline Reinforcement Learning

SAFE ONLINE AND OFFLINE 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
Monday 16 January 2023
at 15:00 o'clock

by

Thiago DIAS SIMÃO

Master of Science in Computer Science,
University of São Paulo, Brazil,
born in Santa Rita de Caldas, Brazil.

The research reported in this dissertation has been funded by the Netherlands Organisation for Scientific Research (NWO), as part of the Energy System Integration: planning, operations, and societal embedding program.

Cover: Elementals #44 by Michael Connolly (CC BY-NC 4.0) .

ISBN 978-94-6384-406-2

An electronic version of this dissertation is available at
`http://repository.tudelft.nl/`.

Contents

SUMMARY

Reinforcement Learning (RL) agents can solve general problems based on little to no knowledge of the underlying environment. These agents learn through experience, using a trial-and-error strategy that can lead to effective innovations, but this randomized process might cause undesirable events. Therefore, to enable the adoption of RL in our daily lives, we must ensure their reliability and safety. Safety requirements are often incompatible with the naive random exploration usually performed by RL agents. *Safe RL* studies how to make such agents more reliable and how to ensure they behave appropriately. We investigate these issues in *online* settings, where the agent interacts directly with the environment, and in *offline* settings, where the agent only has access to historical data and does not interact directly with the environment.

While safety has numerous facets in RL, in this thesis, we focus on two of them. First, the *safe policy improvement* problem, which considers how to compute a policy offline reliably. Second, the *constrained reinforcement learning* problem, which investigates how to learn a policy that satisfies a set of safety constraints. Next, we detail these perspectives and how we approach them.

The first perspective is of particular interest in offline settings. In this setting, we can imagine some decision mechanism has been operating the system, we refer to this mechanism as the *behavior policy*. Assuming these past decisions were recorded in a database, we would like to use RL to compute a new policy using such database. It would be difficult to convince stakeholders to switch to the policy computed by RL if there were chances that the new policy would cause considerable performance loss compared to the behavior policy. Therefore, developing algorithms that reliably compute policies that outperform the behavior policy is essential as this gives confidence to decision-makers that the new policy will not degrade the performance of the underlying system. The safe policy improvement problem formalizes these issues.

Considering that real-world data is limited and costly, in Chapter 3, we investigate how to improve the sample complexity of safe policy improvement algorithms by exploiting the factored structure of the underlying problem. In particular, we consider problems where the dynamics of each state variable depend only on a small subset of the state variables. Exploiting this structure, we develop RL algorithms that require orders of magnitude fewer data to find better policies than their counterparts that ignore such structure. This method also generalizes samples from one state to another, which allows us to compute improved policies if the data only partially cover the problem.

In many real-world applications such as dialogue systems, pharmaceutical tests, and crop management, data is collected under human supervision, and the behavior policy remains unknown. In Chapter 4, we apply safe policy improvement algorithms with an estimated policy built from data. We formally provide safe policy improvement guarantees over the behavior policy even without direct access to it. Our empirical experiments on tasks with finite and continuous states support the theoretical findings.

The second safety perspective is relevant for online RL agents. Engineering a reward signal that allows the agent to maximize its performance while remaining safe is not trivial. Therefore, it is better to decouple safety from reward using constrained Markov decision processes (CMDPs), where an independent signal models the safety aspects. In this setting, an RL agent can autonomously find trade-offs between performance and safety. Unfortunately, most RL agents designed for the constrained setting only guarantee safety after the learning phase, which prevents their direct deployment.

In Chapter 6, we investigate settings where a concise abstract model of the safety aspects is given, a reasonable assumption since a thorough understanding of safety-related matters is a prerequisite for deploying RL in typical applications. We propose an RL algorithm that uses this abstract model to learn policies safely. During the training process, this algorithm can seamlessly switch from a conservative to a greedy policy without violating the safety constraints. We prove that this algorithm is safe under the given assumptions. Empirically, we show that even if safety and reward signals are contradictory, this algorithm always operates safely, while when they are aligned, this approach also improves the agent's performance. Finally, we study how to reduce the performance regret of this algorithm without sacrificing the safety guarantees.

To summarize, we develop new RL methods exploiting prior knowledge about the structure of the problem. We propose reliable offline algorithms that can improve the policy using fewer data and online algorithms that comply with safety constraints while learning. Besides safety and reliability, we also touch on other issues preventing the deployment of RL to real-world tasks, such as data efficiency and learning with a fixed batch of data. Nevertheless, we must recall that other challenges, such as partial-observability and explainability, still require attention. We hope this thesis serves as a stepping stone toward combining different types of prior knowledge to improve various aspects of RL.

Samenvatting

Reinforcement Learning (RL) agents zijn in staat om generieke problemen op te lossen met weinig tot geen kennis van de onderliggende omgeving. Deze agents leren door ervaring op te doen met behulp van een trial-and-error strategie die tot vernieuwende ontdekkingen kan leiden, maar dit willekeurige proces kan ook ongewilde effecten veroorzaken. Om dagelijks gebruik van RL mogelijk te maken moeten we zeker zijn van de betrouwbaarheid en veiligheid. Veiligheidseisen zijn vaak in strijd met de naïeve ontdekkingsstrategie die wordt gebruikt in RL. *Safe RL* onderzoekt hoe hoe RL agents betrouwbaarder te maken en te garanderen dat ze zich daar naar gedragen. Wij onderzoeken deze problemen in een *online* context, waar de agent directe interactie met de omgeving heeft, en in een *ofline* context, waar de agent alleen historische data kan gebruiken en geen directe interactie met de omgeving heeft.

Veiligheid in RL kent meerdere aspecten. In dit proefschrift richten we ons op twee van hen. Ten eerste, het *safe policy improvement* probleem, wat draait om het betrouwbaar berekenen van een *policy* in de offline context. Ten tweede, het *constrained reinforcement learning* probleem, waarin we onderzoeken hoe een policy die aan een aantal veiligheidseisen dient te voldoen geleerd kan worden. In wat volgt beschrijven we deze twee perspectieven en onze aanpak in detail.

Het eerste perspectief is in het bijzonder van belang in een offline context. In een offline context stellen we ons voor hoe een beslissingsmechanisme het systeem bestuurt, en noemen dit mechanisme de *behavior policy*. Aannemende dat beslissingen uit het verleden zijn opgeslagen in een database, willen we RL gebruiken om een nieuwe policy the berekenen op basis van zo'n database. Het is moeilijk betrokken partijen te overtuigen om over te stappen naar de nieuwe policy berekend door middel van RL als er een kans is dat deze nieuwe policy aanzienlijk slechter presteert dan de behavior policy. Daarom is het ontwikkelen van algoritmen die betrouwbaar nieuwe policies die beter presteren dan de behavior policy kunnen berekenen essentieel, aangezien dit beleidsbepalers vertrouwen geeft dat de nieuwe policy niet slechter zal presteren in het onderliggende systeem. Het safe policy improvement probleem formaliseert deze problemen.

Aangezien data afkomstig uit de echte wereld kostbaar en beperkt is, onderzoeken we in Hoofdstuk 3 hoe we de *sample complexity* van safe policy improvement algoritmen kunnen verbeteren door gebruik te maken van de gefactoriseerde structuur van het onderliggende probleem. In het bijzonder kijken we naar problemen waar het gedrag van elke *toestandsvariabele* enkel afhankelijk is van een kleine deelverzameling van alle toestandsvariabelen. Door gebruik te maken van deze structuur bouwen we RL algoritmen die aanzienlijk minder data nodig hebben om betere policies te vinden dan vergelijkbare methoden die zulke structuur ongebruikt laten. Deze methode is ook in staat om willekeurige trekkingen van een toestand te veralgeminiseren naar andere toestanden, wat ons in staat stelt om ook verbeterde policies te berekenen wanneer de data niet het volledige probleem dekt.

In veel realistische toepassingen, zoals spraaksystemen, farmaceutische tests, en landbouwmanagement, wordt data verzameld onder menselijk toezicht en is de behavior policy onbekend. In Hoofdstuk 4 passen we safe policy improvement algoritmen toe op een op basis van data geschatte policy. We geven formele garanties op de verbetering van de safe policy improvement op de behavior policy, zelfs zonder directe toegang tot die behavior policy. Empirische experimenten op taken met eindige en continue states ondersteunen de theoretische resultaten.

Het tweede perspectief is relevant voor online RL agents. Het is moeilijk om een een *rewardsignaal* te maken dat de agent in staat stelt zijn prestaties te maximaliseren en tegelijkertijd veiligheid te garanderen. Daarom is het beter om veiligheid los te koppelen van de reward en *constrained Markov decision processes* (CMDPs) te gebruiken, waar een onafhankelijk signaal de veiligheidsaspecten beschrijft. Dit stelt een RL agent in staat om op autonome wijze een balans vinden tussen prestaties en veiligheid. De meeste RL agents zijn echter ontworpen om alleen veiligheid te garanderen na de leerfase, wat direct gebruik onmogelijk maakt.

In Hoofdstuk 6 onderzoeken we situaties waar een beknopt abstract model van de veiligheidsaspecten is gegeven, wat een redelijke aanname is aangezien een goed begrip van aan veiligheid gerelateerde zaken een vereiste is om RL te gebruiken in praktische toepassingen. Wij geven een RL algoritme dat dit abstracte model gebruikt om op veilige wijze policies te leren. Tijdens het trainingsproces kan dit algoritme naadloos schakelen tussen een behoudende en een *greedy* policy zonder de veiligheidseisen te breken. We bewijzen dat dit algoritme veilig is onder de gegeven aannames. Empirische resultaten laten zien dat zelfs wanneer veiligheids- en rewardsignalen tegenstrijdig zijn, dit algoritme altijd veilig werkt, en wanneer de signalen niet tegenstrijdig zijn zal deze methode ook de prestaties van de agent verbeteren. Als laatste onderzoeken we hoe de prestatieverlies van dit algoritme verminderd kan worden zonder de veiligheidsgaranties op te offeren.

Samenvattend, we ontwikkelen nieuwe RL methoden die bestaande kennis van de structuur uitbuiten. We geven betrouwbare offline algoritmen die de policy kunnen verbeteren met minder data dan online algoritmen die voldoen aan veiligheidseisen tijdens het leren. Naast veiligheid en betrouwbaarheid kijken we ook naar andere problemen die het gebruik van RL voor realitische toepassingen tegenhouden, zoals data efficiëntie en leren van een vaste hoeveelheid data. Desondanks moeten we erop wijzen dat andere uitdagingen, zoals gedeeltelijke observeerbaarheid en uitlegbaarheid nog steeds aandacht vereisen. We hopen dat dit proefschrift een opstap biedt richting het combineren van verschillende soorten bestaande kennis om verscheidende aspecten van RL te verbeteren.

1

INTRODUCTION

In this chapter, we briefly showcase the potential of Reinforcement Learning (RL), discuss the challenges around safety that are preventing the wider adoption of RL, and give an overview of how this thesis aims to address these challenges.

1.1. RELIABLE ARTIFICIAL INTELLIGENCE

Every new piece of technology has the potential to improve our lives, however novel technology must be safe and reliable, otherwise, it will render itself useless. Some Artificial Intelligence (AI) techniques have already proven themselves reliable enough to become part of our daily lives. Other techniques, such as Reinforcement Learning (RL; Sutton and Barto, 2018), are still in development. As we adopt these new techniques, their reliability will be challenged due to the increasing diversity of scenarios they encounter. Therefore, to guarantee their success, particularly for RL, we must ensure they are safe and reliable.

Some AI algorithms are so present in our lives that we may not identify them as AI anymore. When we receive mail in a foreign language we may use our phones to instantly translate it to our native language (Biersdorfer, 2021). This involves using computer vision for character recognition to identify the text of the letter (Lecun et al., 1998) as well as natural language processing to translate the text to the target language (Sutskever et al., 2014). These approaches are often on a realm called supervised learning, where the AI system has access to examples that it can use to generalize to new situations. In particular, the feedback of a decision is immediate, which allows these algorithms to quickly adapt the future decisions in case of a mistake.

RL algorithms can handle problems that require a sequence of decisions, taking into account the uncertainty of future outcomes. These algorithms have shown fascinating results in confined settings, such as playing board games and controlling simulations of high-dimensional robots (Mnih et al., 2015; Schulman et al., 2016). While they are still mostly restricted to the research lab, we hope to eventually use them to handle complex tasks in the real world like driving, energy optimization, and treatment recommendation for healthcare. However, these applications are critical, so a new algorithm will only be adopted if its safety is guaranteed. Unfortunately, providing safety guarantees for these tasks is not easy, since in RL it is difficult to predict the consequences of certain decisions without trying them first.

1.2. REINFORCEMENT LEARNING: PROMISES & CHALLENGES

Reinforcement Learning (RL; Sutton and Barto, 2018) is a paradigm of extremely general algorithms for sequential decision making in unknown environments. In RL, an agent interacts with an environment by repeatedly observing the state of the environment and taking an action, which in turn changes the state of the environment. After each action, the agent obtains a reward indicating the progress made towards completing its task. Through these interactions, often in a trial-and-error fashion, the agent computes a *policy*, which prescribes the actions to be executed in each state. The goal of the agent is to find a policy that collects the most reward over time.

Following this relatively simple framework, RL algorithms have completed astonishing tasks, for example, playing Atari games at a super-human level (Mnih et al., 2015), performing locomotion tasks with complex simulated robots (Schulman et al., 2016), playing abstract board games like chess and go (Silver et al., 2016, 2017), and playing strategy multi-player video games (Baker et al., 2020; Vinyals et al., 2019). We may notice a common characteristic among these accomplishments, namely a simulated environment, where the trial-and-error strategy used by RL algorithms is not harmful, which makes it look like RL is confined to simulated environments. Nevertheless, the complexity of these tasks raises hopes that RL can solve real world tasks.

RL has the potential to solve a diverse set of complex tasks beyond simulations and research to bring it to real world applications has already begun. To exemplify this diversity, we may cite surveys of RL being used in specific applications, such as energy (Glavic et al., 2017; Vázquez-Canteli and Nagy, 2019; Wang and Hong, 2020; Yang et al., 2020a), healthcare (Coronato et al., 2020; Liu et al., 2020), autonomous driving (Aradi, 2020; Kiran et al., 2021), traffic signal control (Wei et al., 2021; Yau et al., 2017), logistics (Althamary et al., 2019; Haydari and Yilmaz, 2020), production systems (Panzer and Bender, 2021; Yoo et al., 2021), distributed systems (Chen et al., 2021a; Jameel et al., 2020; Uprety and Rawat, 2021; Wu et al., 2021), and communication networks (Qian et al., 2019). Nevertheless, the use of RL remains largely a research topic due to numerous assumptions made by RL algorithms that do not match the complexity of the real world.

To bypass the complexity of the real world, we might consider applying RL in a simulator and later transfer the policy found to the real system. For instance, RL can learn to imitate animal motion in simulations and eventually transfer the policy learned to physical robots (Peng et al., 2020). However, this requires meticulous engineering, for example, to model the robot and its physics (Batra et al., 2021; Zhao et al., 2020). Another issue is that even small inaccuracies in the simulator can be propagated causing performance losses in the real system (Ha et al., 2020). Hence, we may prefer to deploy the RL agent directly in the environment, avoiding the gap between the simulator the real system, but this approach has its challenges.

Interacting directly with the system can be costly and risky. Unfortunately, RL algorithms are notorious for requiring a large number of interactions with the environment to find good policies, which can get to the order of millions (Mnih et al., 2016; Schulman et al., 2017). While that may not be a problem in simulated environments, interactions in the real world incur costs, for instance to power and maintain an electric vehicle, which would accumulate over these interactions. Similarly, such interactions may lead to undesirable situations, such as colliding with another vehicle or driving out of the road, and the probability of such events increases with each interaction between the agent and the environment. So, to alleviate these issues, we must minimize the number of experiences necessary to learn a reasonable policy.

Some applications already have a policy in execution, in which case we could use realistic historical data to learn a reasonable initial policy, reducing the costs and risks of RL algorithms. Consider for instance a recommender system that already has a policy giving recommendations to a user. In this situation, the algorithm can rely on historical data to predict if a user is interested in a certain item, which might be more effective than to try recommending that item to the user (Chen et al., 2019). However, basic RL algorithms can be unreliable in this setting, computing policies with low performance. Therefore, finding reliable methods is fundamental to unleash the potential of the available historical data. We will investigate the so-called *safe policy improvement* problem, where we must compute a new policy using the historical data that outperforms the policy in execution.

When no historical data is available, the agent must start from scratch. This is particularly challenging because of the random nature of the exploration performed by RL agents. Some systems do not allow the agent to take arbitrary actions or visit certain states that could have catastrophic consequences, such as breaking a robot or crashing an autonomous vehicle. We will investigate this problem using the *constrained reinforcement learning* framework, which explicitly captures the constraints of the system.

Recently, Dulac-Arnold et al. (2021) compiled a set of nine challenges that are preventing the direct deployment of RL to real world tasks, often related to discrepancies between the environments where RL is being tested and real world tasks, such as partially observable states, poorly specified reward functions, or system delays. Among them, we also have the problems addressed in this thesis: *offline training* and *learning with safety constraints*. This compilation reinforces the need to handle these challenges to enable the wide adoption of RL.

1.3. THE SAFETY CHALLENGE

Since RL often relies on a trial-and-error strategy major concerns around safety arise when we consider its deployment (Amodei et al., 2016). Safe Reinforcement Learning (SRL) is a wide subfield within RL research that investigates how to mitigate such issues (García and Fernández, 2015). It has two main branches, changing the optimization criteria to ensure reasonable performance and changing the exploration process to prevent undesirable outcomes, as Figure 1.1 illustrates. In this section, we present a short overview of each direction. We will discuss our take on each of them in the next section.

1.3.1. Alternative Criteria

Due to the stochastic nature of the environment, in each trajectory, the agent might observe a different *return*, the accumulated reward of that trajectory. Traditionally, RL algorithms aim to maximize the return in expectation. However, this does not give any guarantee about the return of the worst trajectories. So, an agent might return policies that have some probability of observing unacceptably low returns. This could for instance lead a company to bankruptcy. In that case, the agent should have a more conservative behavior to reduce such risks.

Alternative criteria can be used to prevent such undesirable outcomes. For instance, using a risk-averse criterion (Howard and Matheson, 1972) or a min-max criterion that considers the worst-case outcome (Heger, 1994). Such criteria may also account for the uncertainty the agent has about the environment. In this situation, one may choose to be pessimistic with respect to the different dynamics the environment can present.

1.3.2. Safe Exploration

In safe exploration, the goal of the agent is to learn without experiencing catastrophic outcomes. A catastrophe event can have different definitions. One may consider preventing the agent from entering undesirable states (Turchetta et al., 2016), others might define safety using temporal logic constraints (Junges et al., 2016). We could also ensure the total cost of a certain trajectory does not exceed a given threshold (Moreira et al., 2021). In summary, this line of research aims to guarantee that the policy executed in a given episode does not violate the safety constraints.

To accomplish this task, one usually must rely on some type of prior knowledge. This can take numerous forms, such as a teacher that provides advice, an initial safe policy, a set of safe states, or demonstrations from an expert. Without prior knowledge, it would be infeasible to ensure the safety of the system.

1.4. Scope

Figure 1.1 help us understand the scope of this thesis. It puts the RL branch of machine learning in focus, showing our interest in sequential decision making. Then, it shows some of the challenges to take RL to the real world and expands safe reinforcement learning in the two branches described before. In the bottom of the diagram, we see the two main problems addressed on this thesis. We notice that both problems are alternative criteria to the standard reinforcement learning objective, but they also share connections with other challenges in RL and with the safe exploration branch in SRL, shown in dashed lines. We discuss each problem next.

1.4.1. Reliability in Offline Reinforcement Learning

In offline RL, also known as batch RL, the agent only has access to a fixed dataset of historical data and does not interact directly with the environment (Lange et al., 2012; Levine et al., 2020). This approach can circumvent the data collection challenges in setting where interactions with the environment are costly or risky. Usually, we assume the dataset was collected following a single policy, possibly hand-designed, which we refer to as the *behavior policy* π_b. From this perspective, it is important that the new policy π'

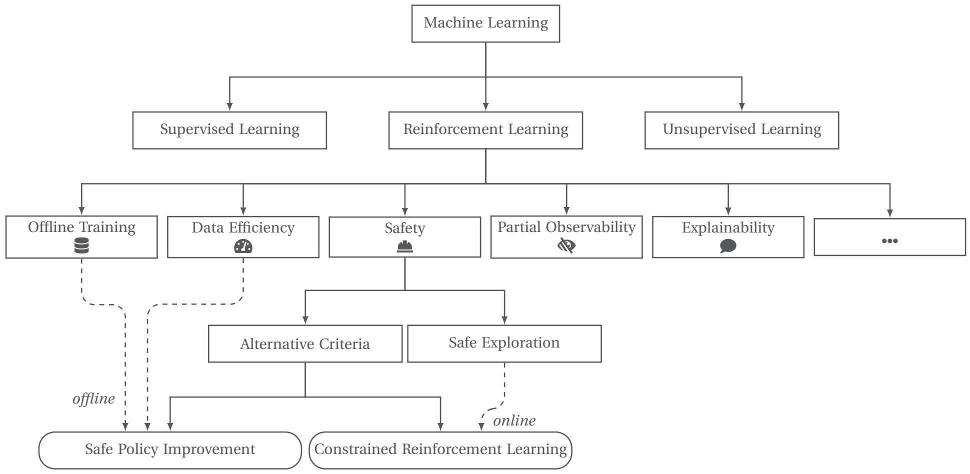


Figure 1.1: Thesis scope, showing RL as the machine learning branch in focus. Then, it shows some of the challenges to bring RL to the real-world. Finally, it shows the the main problems (rounded boxes) under consideration in this thesis and how they fit within the safe RL literature.

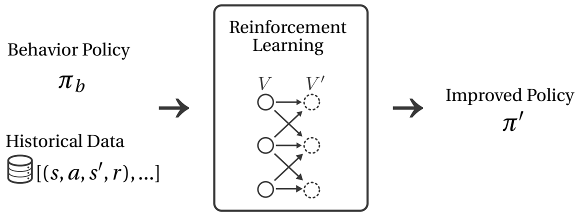


Figure 1.2: Interface of a reinforcement learning agent solving the safe policy improvement problem.

outperforms π_b, otherwise, a decision maker would prefer to simply keep executing the behavior policy.

We will consider the Safe Policy Improvement (SPI) criterion (Thomas et al., 2015b), which formalizes this improvement requirement. Figure 1.2 shows the interface of an agent that optimizes the SPI criterion. We observe that besides the dataset of historical data, the agent also gets as input the behavior policy π_b. If an algorithm has a high probability of returning a policy π' that outperforms π_b, we say this algorithm satisfies the SPI criterion.

In Figure 1.1, the connection between the offline training challenge and the safe policy improvement problem is clear from the problem definition. The connection from the data efficiency challenge is related to the limited amount of data available, an inherent property of offline RL since the agent does not interact with the environment. This issue impacts directly the performance of the policy computed by the agent.

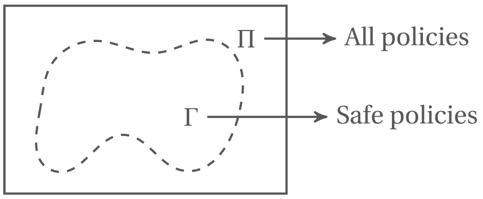


Figure 1.3: Policy space and set of safe policies.

1.4.2. SAFETY CONSTRAINTS IN REINFORCEMENT LEARNING

We also investigate the Constrained Reinforcement Learning (CRL) problem, where the optimal policy must satisfy a set of safety constraints. Relying on the reward function to account for safety can be impractical. Consider for instance an autonomous vehicle that wants to minimize the time to reach its destination, it could end up driving at high speeds putting the lives of its passengers at risk. In this task, safety has a higher priority than performance, so it is better to drive cautiously and avoid risking the passengers' well-being. At the same time, driving too slow may lead to a failure in delivering the passenger on time. Therefore, we must find a trade-off between safety and performance, which is not easily compiled in the reward function. It is more practical to consider a signal dedicated to safety and define constraints with respect to this safety signal, letting the algorithm find this trade-off automatically (Hayes et al., 2021; Roy et al., 2021).

Many algorithms have been proposed for CRL (Achiam et al., 2017; Efroni et al., 2020; HasanzadeZonuzy et al., 2021; Ray et al., 2019; Yang et al., 2021, 2020b), but they often only focus on the safety of the final policy, meaning they might violate the safety constraints during the learning process. This approach may be suitable for situations where a perfect simulator is available, which, unfortunately, is not always the case as we mentioned earlier. Therefore, we investigate how to learn without violating the safety constraints.

Figure 1.3 shows the space of policies in CRL. Π represents the policy space and Γ indicates the set of policies that satisfy the safety constraints. The objective in CRL is to find a policy in Γ with the best performance. From the safe exploration perspective, the challenge is to ensure that in every episode a safe policy $\pi \in \Gamma$ is executed during the learning process.

As Figure 1.1 shows, we consider this problem from both branches of SRL. As García and Fernández (2015) observe, the constraints change which policies are optimal, therefore, the problem falls in the alternative criteria branch. But, we also consider this problem from the safe exploration perspective, since we are interested in always ensuring the policy executed respects the safety constraints, even while the agent is still learning.

1.5. RESEARCH GOALS

As discussed so far, we are interested in making RL algorithms more safe and reliable, such that they eventually can be deployed directly to real world tasks. In other words, we would like to alleviate the dependency on simulators while training RL agents. As already mentioned, many challenges could be tackled to help with this task, but we narrow our scope to two of them:

i. expanding the reach of reliable offline RL, and

ii. developing new methods to ensure safety during exploration of online RL.

Considering that without making assumptions about the underlying problem, it is hardly possible to make any progress in safe RL, our first research question (RQ) is related to what kind of prior knowledge we may have about the problem.

> **RQ 1.** *What kinds of prior knowledge can an expert reasonably provide regarding the underlying problem? How to integrate such knowledge into safe reinforcement learning algorithms?*
> Overall, RL handles an extensive set of problems. While this expands the potential applications of RL, it also contributes to its instability. It is difficult to make claims about the safety and reliability of RL algorithms given such a large class of problems. Assuming an expert provides a model of the underlying problem would bypass this issue. However, such a model might contain errors that could make this approach unreliable. Nevertheless, an expert probably has insights regarding the structure present on the problem of interest. Such structure would allow a safe RL agent to reason about specific problems. This way, we may consider specific classes of problems to make statements about the safety and reliability of RL algorithms. This approach also exposes our assumptions, which, in the future, can help decision makers evaluate if the algorithms we are proposing are suitable for their applications.

Regarding the first challenge, previous offline RL algorithms guarantee to return a policy with reasonable performance, by relying on the behavior policy (Laroche et al., 2019; Petrik et al., 2016; Thomas et al., 2015b). This way, when historical data and the behavior policy are available, we can use these algorithms to compute an improved policy with high confidence. Unfortunately, the amount and quality of such data are not guaranteed, there might be little data available or the coverage of the problem might be limited. This can make these algorithms overly conservative, leading them to always rely on the behavior policy. The dependency on the behavior policy is another concern since the absence of such policy precludes the use of these algorithms. These issues lead us to the following three research questions.

> **RQ 2.** *Considering that real world data is limited, how to improve the sample complexity of safe policy improvement algorithms to make the most of the data available?*
> When we consider the costs of interacting with the environment to collect the historical data, we may notice that many applications have limited amounts of historical data. However, it is still important to compute improved policies and avoid the overly conservative strategy that always relies on the behavior policy. We may use some prior knowledge regarding the

problem class and compute new policies even if the amount of data available is small. This approach helps us ensure we make the best use of the data available.

RQ 3. *Since we do not control the quality of the data available, how to ensure we still can compute improved policies even when the coverage of the problem is limited?*
In offline RL, the agent has no control over how the data was collected. Therefore, the data might be unbalanced. This means that some regions of the environment can have little support, in which case being able to generalize the available data to the unsupported states is crucial while looking for a better policy.

RQ 4. *Assuming we only have access to a set of past trajectories, that is without access to the behavior policy, can we still develop offline RL with improvement guarantees?*
While a considerable number of RL algorithms have guarantees of returning a reasonable policy in the offline setting, these algorithms require access to the policy that collected the data, which may not be available. In this case, it is interesting to investigate if access to the behavior policy is strictly required. Finding algorithms with improvement guarantees even when the behavior policy is another way we can also expand the reach of offline RL.

There is also some prior work related to our second challenge, for instance, some methods can learn a policy that respects the safety constraints by interacting with the environment (Efroni et al., 2020; Zheng and Ratliff, 2020). Unfortunately, most of these approaches do not satisfy the safety constraints during the learning process. Therefore, they are mostly confined to settings where a perfect simulator is available and can be used to train a safe policy that only later is deployed. This makes these algorithms inadequate for safe exploration. As we mentioned before, the usual way to develop methods for safe exploration is to rely on some kind of domain knowledge. Finding new types of prior knowledge that provide safe exploration diversifies the repertoire of SRL algorithms. In this way, we hope the user can easily check what type of prior knowledge is available on its application and deploy the corresponding SRL agent safely. This motivates the next two research questions.

RQ 5. *Given a partial model related to the safe dynamics, how can a CRL agent learn to optimize its task without violating its safety constraints?*
Considering that the deployment of RL typically requires a reasonable understanding of safety-related matters, we may assume that such knowledge is given by an expert. This can be expressed by a concise abstract model. Since we do not make any assumption on the task at hand, it is still necessary to learn how to optimize the underlying task. The challenge is to

ensure that, even while still learning, the policy being executed by the agent is *always safe*.

RQ 6. *What is the impact of the safety constraints on the performance of an agent that explores safely?*
Even considering some type of prior knowledge, this agent still has some uncertainty about the environment dynamics. Therefore, the agent still must explore. This leads to a new perspective on the classic exploration-exploitation dilemma faced by RL agents, where the agent must visit new states cautiously, to avoid violating the safety constraints. We want to investigate how this caution affects the learning process, comparing for instance with an unconstrained agent.

In summary, Research Question 1 provides the basis to Research Questions 2 and 3 as well as Research Question 5. Research Questions 2 to 4 are dedicated to the reliability of offline RL, while Research Questions 5 and 6 focus on safe exploration of CRL algorithms.

1.6. CONTRIBUTIONS

In this section, we present an overview of the thesis' contributions.

Data efficiency in safe policy improvement (Chapter 3). Considering that real world data is limited and expensive, we investigate how to improve the sample complexity of safe policy improvement algorithms exploiting the structure of the problem. Most algorithms with improvement guarantees in the offline setting consider unrestricted problems. This leads to general bounds that depend on the size of the state space. Unfortunately, the state space can grow exponentially in the number of variables describing the problem, and, consequently, the amount of data required by these algorithms to compute better policies can be prohibitively large. Therefore, we investigate this problem for different classes of problems, which allows us to obtain better estimates of the underlying dynamics. In particular, we consider problems where the dynamics of each variable depend only on a small subset of the remaining variables. Exploiting this structure, we develop RL algorithms that require orders of magnitude fewer data to find better policies than their counterparts that ignore such structure.

Generalization in safe policy improvement (Chapter 3). Exploiting the structure of the problems allows us to generalize samples from one state to another. This is critical for safe policy improvement when the behavior policy does not have good coverage of the state-action pairs. For instance, if the behavior policy is deterministic, the agent is not able to deploy a new policy without some generalization capability. By exploiting the structure of the problem, we can infer the effects of an action in an arbitrary state if this action was applied in similar states. For example, if in a state s_1 all the data observed takes action a_1, the agent cannot apply action a_2. But, when the agent has some observations where a_2 was executed in s_2, it may be able to infer the effects of a_2 in the state s_1, if the agent knows that the states s_1 and s_2 are similar. This gives the agent the unique ability to confidently return policies that are significantly different from the policy that collected the data.

Safe policy improvement without the baseline policy (Chapter 4). In the safe policy improvement literature, most algorithms with improvement guarantees require access to the behavior policy as a fallback mechanism, however, in some applications this policy might not be available, for instance in cases where the decisions were made by a human expert. We study how to handle this problem when the behavior policy is not available, using an estimate of this policy. This brings such reliable algorithms closer to the basic offline RL setting, where, in general, we do not assume to have access to the policy used during data collection.

Exploiting partial knowledge of the system dynamics for safe exploration (Chapter 5). We study how to ensure safety during the learning process in an online setting, that is, the agent must learn but it is not allowed to violate the safety constraints. We propose to use an abstraction of the safety dynamics that allows the agent to reason about the safety of the policies it is deploying. Our agent keeps track of a set of probable models of the environment and acts pessimistically with respect to them. This agent can explore the environment optimistically with respect to the reward, without ever violating the safety constraints.

1.7. OUTLINE

Chapter 2 starts by presenting a review of the basic models for sequential decision making, how to represent the structure of the underlying problem, and how to describe the desired behavior of the agent using constraints. Next, it gives a basic overview of how to solve these problems through planning as well as reinforcement learning. Finally, it presents a discussion of the safety challenges in reinforcement learning.

Chapters 3 to 5 contain the main contributions of the thesis. Chapters 3 and 4 are dedicated to the reliability of offline RL algorithms, while Chapter 5 investigates the safety of online RL algorithms.

In Chapter 3 we formalize the SPI problem, review the literature dealing with this problem as well as algorithms that can solve it. Then, we present the factored SPI framework, which can take into account the structure of the underlying problem. We show that this approach has a lower sample complexity and better generalization capabilities.

Following up, in Chapter 4 we investigate the SPI in settings where the behavior policy is unknown. We show that using an estimate of the behavior policy, we can also get safe policy improvement guarantees.

In Chapter 5 we focus on the online setting. We investigate how an RL agent can learn by interacting with the environment without violating the safety constraints.

To conclude, Chapter 6 provides a discussion of the main results of the thesis and pointers for future work. To help the reader the thesis includes lists of notations and acronyms in Appendices A and B.

2

Background

In this chapter, we review the framework used for sequential decision making under uncertainty. We describe how to model such problems and how to represent their solutions. Then, we present algorithms to solve them in settings with a known model. We also discuss how to solve these problems when the model is unknown and learning from experiences is necessary. Finally, we discuss the safety challenges in learning without the model.

2.1. Markov Decision Processes

The Markov Decision Process (MDP; Puterman, 1994) is a mathematical framework commonly used to model the interaction between an *agent* and its *environment* (Figure 2.1). Denoting the simplex set by $\mathscr{P}(\mathbb{Y})$, which indicate the set of probability distributions over the finite set $\mathbb{Y}$, we define an MDP $\mathscr{M}$ by a tuple $\langle \mathbb{S}, \mathbb{A}, P, R, \gamma, \mu, H \rangle$, where:

- $\mathbb{S}$ is a discrete set of states of the world,
- $\mathbb{A}$ is a discrete set of actions the agent can execute,
- $P : \mathbb{S} \times \mathbb{A} \to \mathscr{P}(\mathbb{S})$ is a transition function that describes how the environment evolves, so $P(s' \mid s, a)$ is the probability of moving to $s' \in \mathbb{S}$ after executing action $a \in \mathbb{A}$ in state $s \in \mathbb{S}$,
- $R : \mathbb{S} \times \mathbb{A} \to [R_\perp, R^\top]$ is a reward function, so $R(s, a)$ indicates the reward obtained after executing action $a \in \mathbb{A}$ in state $s \in \mathbb{S}$,
- $\gamma \in [0, 1]$ is a discount factor that captures the agent's preference for immediate rewards over future rewards,
- $\mu \in \mathscr{P}(\mathbb{S})$ is a distribution over initial states, so $\mu(s)$ is the probability of the interaction between the agent and the environment start on state $s \in \mathbb{S}$, and
- $H \in \mathbb{N} \cup \{\infty\}$ is the horizon that indicates the number of times the agent interacts with the environment.

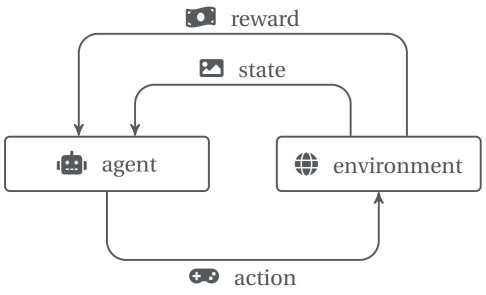


Figure 2.1: Interaction between an agent and the environment described by an MDP.

Figure 2.1 shows how an agent interacts with its environment. Given the current state of the environment, the agent chooses an action to execute. Then, the environment changes to a new state according to the transition function. Finally, the agent observes the new state and the reward obtained from the last interaction. This interaction repeats until a termination criterion is met.

A solution for an MDP describes how the agent chooses the actions to be executed in each interaction with the environment, and we call it a *policy*. In this section, we will focus on infinite-horizon MDPs with the expected total discounted reward optimality criterion, that is, we assume $H = \infty$ and $\gamma < 1$.[1] In this case, the optimal solution can be Markovian and stationary, meaning the decision at a given state is independent of the states visited previously in any time step and does not change over time. Although a deterministic policy is enough to represent the optimal policy for this setting, we will consider randomized policies, which facilitates the exposition of this thesis. We define a stationary randomized policy as a mapping from states to a distribution over actions:

$$\pi : \mathbb{S} \to \mathscr{P}(\mathbb{A}).$$

This way, when the environment is on a state $s \in \mathbb{S}$, an agent following policy $\pi(s) \in \mathscr{P}(\mathbb{A})$ will execute action $a \in \mathbb{A}$ with probability $\pi(a \mid s)$. We use $\pi(s) = a$ to indicate a deterministic policy $\pi : \mathbb{S} \to \mathbb{A}$, that is, a policy that puts all the probability mass on a single action $\pi(a \mid s) = 1$. We denote the set of policies by Π.

An optimal policy maximizes the *expected return*, defined by the the expected sum of discounted rewards:

$$V(\pi, \mathscr{M}) = \mathop{\mathbb{E}}_{\substack{S_0 \sim \mu, \\ S_t \sim P(\cdot \mid S_{t-1}, A_t), \\ A_t \sim \pi(\cdot \mid S_t)}} \left[\sum_{t=0}^{\infty} \gamma^t R(S_t, A_t) \right],$$

where the random variables S_t and A_t are distributed according to the policy π and the dynamics of the MDP $\mathscr{M}$.

We can define the expected value of following a policy π starting from state s as follows (Puterman, 1994):

$$V^{\pi}(s) = \sum_{a \in \mathbb{A}} \pi(a \mid s) Q^{\pi}(s, a),$$

[1] We refer to Puterman (1994) and Mausam and Kolobov (2012) for alternative settings.

Algorithm 1 Value Iteration.

```
Input: MDP $\mathscr{M} = \langle \mathbb{S}, \mathbb{A}, P, R, \gamma, \mu, H \rangle$
Input: $\epsilon$ : a precision parameter
1: $V_0(s) = 0, \quad \forall s \in \mathbb{S}$
2: $i = 0$
3: repeat
4:     $i \leftarrow i + 1$
5:     $V_i(s) \leftarrow \max_{a \in \mathbb{A}} \left[ R(s,a) + \sum_{s' \in \mathbb{S}} \gamma P(s' \mid s,a) V_{i-1}(s') \right], \quad \forall s \in \mathbb{S}$
6: until $|V_i(s) - V_{i-1}(s)| < \epsilon, \quad \forall s \in \mathbb{S}$
7: $\pi(s) \leftarrow \arg\max_{a \in \mathbb{A}} \left[ R(s,a) + \sum_{s' \in \mathbb{S}} \gamma P(s' \mid s,a) V_i(s') \right], \quad \forall s \in \mathbb{S}$
8: return $\pi$
```

where, $Q^\pi(s,a)$ is the value of taking an action a on state s, assuming we keep following the policy π:

$$Q^\pi(s,a) = R(s,a) + \sum_{s' \in \mathbb{S}} \gamma P(s' \mid s,a) V^\pi(s').$$

Intuitively, the function Q^π is the immediate reward plus the discounted value of each future state weighted by the probability of reaching that state.

Given the value V of each state, we can compute the value of a policy π on the MDP $\mathscr{M}$ based on the initial state distribution μ:

$$V(\pi, \mathscr{M}) = \sum_{s \in \mathbb{S}} \mu(s) V^\pi(s).$$

The optimal solution for an MDP can be represented by a deterministic stationary Markov policy π^* whose value, denoted here by V^*, dominates the value of any other policy π in all states of the MDP (Puterman, 1994):

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

Now, given the optimal value function V^*, we can extract the optimal policy π^*:

$$\pi^*(s) \in \arg\max_{a \in \mathbb{A}} \left[R(s,a) + \sum_{s' \in \mathbb{S}} \gamma P(s' \mid s,a) V^*(s') \right], \qquad \forall s \in \mathbb{S}.$$

One way to compute an optimal policy for an MDP is to estimate the optimal value function V^*. We can accomplish that using the value iteration algorithm (Algorithm 1). This algorithm starts with an arbitrary estimate of the value function (Line 1). Then, it iteratively refines this estimate to make it approach the optimal value function by successively applying the Bellman backup operator (Line 5). This process stops when the value functions of two consecutive iterations are sufficiently close (Line 6).

The MDP framework is highly general and can represent a heterogeneous set of applications (Feinberg and Shwartz, 2002). However, specifying such models can be challenging when the state space is too large. In the next section, we present a framework that facilitates this task, in which neither the states, transition function, or reward function are built explicitly (Sabbadin et al., 2020).

2.2. FACTORED MARKOV DECISION PROCESSES

A Factored Markov Decision Process (FMDP; Boutilier et al., 1995) is a model that compactly describes the state space $\mathbb{S}$ of an MDPs with a set of *state variables* $\mathbb{X} = \{X_1, \cdots, X_n\}$, and each variable X might assume a value x from its domain $\mathrm{dom}(X)$. We define the domain of a set of variables $\Delta \subseteq \mathbb{X}$ as the Cartesian product of their respective domains $\mathrm{dom}(\Delta) = \times_{X \in \Delta} \mathrm{dom}(X)$. Then, we can define the state space of the MDP as the domain of the set of state variables: $\mathbb{S} = \mathrm{dom}(\mathbb{X})$. Two assumptions are commonly made that allow us to compactly represent the transition function using a Dynamic Bayesian Network (DBN; Dean and Kanazawa, 1989).

First, the outcome of each variable is independent of the outcome of the remaining variables:

$$P(s' \mid s, a) = \prod_{X \in \mathbb{X}} P(s'[X] \mid s, a),$$

where $s[\Delta] \in \mathrm{dom}(\Delta)$ denotes the value of the variables $\Delta \subseteq \mathbb{X}$ on the state $s \in \mathbb{S}$[2] and $P(x \mid s, a)$ is the probability of observing $x \in \mathrm{dom}(X)$ conditioned on the state action pair $(s, a) \in \mathbb{S} \times \mathbb{A}$.

Second, the dynamics of a state variable $X \in \mathbb{X}$ for a given action $a \in \mathbb{A}$ depend only on its *parents*, a subset of the variables $\mathrm{Pa}_a(X) \subseteq \mathbb{X}$. For example, on Figure 2.2a the parents of the state variable X_2 are X_1 and X_2 as indicated by the highlighted arrows leading to X_2'. In this way, the probability distribution of each variable $X \in \mathbb{X}$ can be conditioned only on $\mathrm{Pa}_a(X)$, instead of the the complete state s:

$$P(x \mid s, a) = P(x \mid s[\mathrm{Pa}_a(X)], a), \qquad \forall x \in \mathrm{dom}(X).$$

We can use a Conditional Probability Table (CPT) to represent this probability distribution, as Figure 2.2b exemplifies.

These assumptions yield a compact representation of the transition function:

$$P(s' \mid s, a) = \prod_{X \in \mathbb{X}} P(s'[X] \mid s[\mathrm{Pa}_a(X)], a). \tag{2.1}$$

We denote the support of transition components that describe an FMDP by

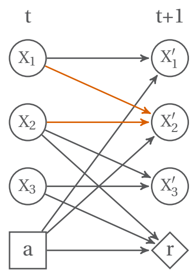
$$\mathcal{Q} = \left\{ (X, a, \mathbf{x}) \in \mathbb{X} \times \mathbb{A} \times \bigcup_{\Delta \in 2^X} \mathrm{dom}(\Delta) \;\middle|\; \mathbf{x} \in \mathrm{dom}(\mathrm{Pa}_a(X)) \right\}, \tag{2.2}$$

where 2^X the power set of the set of state variables $\mathbb{X}$. The size of $\mathcal{Q}$ denotes the number of transition components that must be estimated by a factored RL algorithm. We also define $D_{s,a} \subseteq \mathcal{Q}$ as the relevant transition components of a state-action pair $(s, a) \in \mathbb{S} \times \mathbb{A}$:

$$D_{s,a} = \{ (X, a, \mathbf{x}) \in \mathcal{Q} \mid \mathbf{x} = s[\mathrm{Pa}_a(X)] \}, \qquad \forall s, a \in \mathbb{S} \times \mathbb{A}. \tag{2.3}$$

We may notice that different states may have transition components in common. This is what allows a compact representation of the transition function.

[2]We use the brackets to emphasize that given a state s, we are interested in the values of a subset of the state variables in s. In favor of clarity, we omit the set notation of singletons, so $s[X] = s[\{X\}] : \forall X \in \mathbb{X}$.



(a) DBN of action a.

X_1	X_2	X'_2	
		■	□
■	■	0.5	0.5
■	□	0.3	0.7
□	■	0.1	0.9
□	□	0.2	0.8

(b) CPT of the state variable X_2 and action a.

X_2	X_3	r
■	■	3
■	□	1
□	■	2
□	□	0

(c) Local reward function.

Figure 2.2: A FMDP with 3 state variables.

Denoting $\mathbb{N}_n = \{1, \ldots, n\}$, $\forall n \in \mathbb{N}$, we can also succinctly represent the reward function by the sum of R *local* functions (Degris et al., 2008), respectively:

$$R(s,a) = \sum_{i \in \mathbb{N}_{\mathsf{R}}} R_i(s[\Delta_i], a),$$

where R_i is the i-th local reward function that depends for the features $\Delta_i \subseteq \mathbb{X}$. Intuitively, these local function only only depends a subset of the state variables as shown in the following example.

Example 1 (An FMDP represented by a DBN.). *Figure 2.2 shows an FMDP with 3 binary state variables* $\{X_1, X_2, X_3\}$ *and a single local reward function. For an action* $a \in \mathbb{A}$*, it shows the dependencies between state variables represented by a DBN, where the first layer contains the state variables of the current time step* t*, while the second contains the state variables of the next time step* $t+1$ *(Figure 2.2a). The figure also shows the future probability distribution of the state variable* X_2 *represented by a CPT which is only conditioned on the current values of the parents* X_1 *and* X_2 *(Figure 2.2b), and the local reward function that is independent of the state variable* X_1 *(Figure 2.2c).*

While the size of each CPT is exponential on the number of parents, typically, the parents of a state variable and action pair are only a subset of the state variables, ensuring the size of the CPT remains small (Oliehoek et al., 2008). Representing the problem with smaller CPTs can help an expert define the problem since, in this framework, it is not necessary to enumerate all the combinations of state variables values (Sabbadin et al., 2020). Moreover, different tools and languages can be used to specify FMDPs (Mausam and Kolobov, 2012; Sanner, 2010; Younes and Littman, 2004). This representation also facilitates the development of planning algorithms that exploit these problems' structure to solve large MDPs efficiently. For instance, symbolic probabilistic planners use decision diagrams to represent the value function and policy (Boutilier et al., 1999; Feng and Hansen, 2002; Hansen, 2021; Hoey et al., 1999). This structure has also been exploited to develop heuristics for tree search algorithms (Geißer and Speck, 2018), to compute reactive policies for problems with continuous state variables (Bueno et al., 2019), and to generate abstractions that reduce the computational cost of the planning algorithms (Chitnis et al., 2020; Dearden and Boutilier, 1997).

Due to their expressiveness, FMDPs have been used in several applications, for instance, an intelligent assistant that helps maintain a safe operation of a power plant and (Reyes et al., 2009), a hydroelectric reservoir system that regulates the water flow in a dam (Reyes et al., 2015), and in recommender systems that identifies the topic of a user's session (Tavakol and Brefeld, 2014).

Besides benefiting planning algorithms, this structure can also be exploited by agents learning the model of the environment, as we discuss in Section 2.4.

2.3. Constrained Markov Decision Processes

The MDP framework is primarily designed for problems with a single reward function. However, many problems have multiple objectives that are not easily combined into a single scalar (Roijers et al., 2013). For instance, it is not easy to define a signal that combines efficiency and safety. Consider the case of an autonomous robot that must navigate the surface of Mars without any accidents (Moldovan and Abbeel, 2012). To increase the expressiveness of this model, one may consider *constraining* some of these objectives, such that the agent still aims to maximize the main objective but still keeps the remaining objectives bounded. This approach allows practitioners to directly specify the desired behavior of the agent (Kamran et al., 2022; Roy et al., 2021). Therefore, in this setting, the agent's goal is to find the policy with the highest return among those that respect the constraints. These problems are usually modeled by a Constrained Markov Decision Process (CMDP; Altman, 1999).

A CMDP is defined by a tuple $\mathcal{M} = \langle \mathbb{S}, \mathbb{A}, P, R, \gamma, \mu, H, C, \hat{c} \rangle$ where most elements are the same as in an MDP except for

- $C \colon \mathbb{S} \times \mathbb{A} \to [C_\perp, C^\top]$ that is a cost function, and
- $\hat{c} \in \mathbb{R}$ which is an upper bound on the expected accumulated cost.

Although we present CMDPs with a single cost function in favor of clarity, this model can easily be extended to problems with multiple cost functions.

In this section, we will focus on problems with a finite horizon ($H < \infty$) and no discount ($\gamma = 1$). This way the optimal behavior might be different depending on how far it is from the end of the episode, so we consider a policy indexed by the current time step $\pi \colon \mathbb{S} \times \mathbb{N}_H \to \mathscr{P}(\mathbb{A})$.

A policy π induces a state-action *occupancy* $y_t(s,a) = \mu_t(s)\pi(a \mid s,t)$, where $\mu_t(s)$ is the occupancy of the state s at time step t:

$$\mu_t(s) = \begin{cases} \mu(s) & \text{if } t = 1, \\ \sum_{s^\circ, a^\circ \in \mathbb{S} \times \mathbb{A}} y_{t-1}(s^\circ, a^\circ) P(s \mid s^\circ, a^\circ) & \text{otherwise.} \end{cases}$$

An optimal policy π^* for a CMDP maximizes the expected accumulated reward and has an expected accumulated cost lower than the upper bound $\hat{c}$:

$$\begin{aligned} \max_\pi V_R^\pi(\mu) &= \sum_{s \in \mathbb{S}} \mu(s) V_R^\pi(s,1) = \mathbb{E}_\pi \left[\sum_{t \in \mathbb{N}_H} R_t \mid \mu \right] \\ \text{s.t. } V_C^\pi(\mu) &= \sum_{s \in \mathbb{S}} \mu(s) V_C^\pi(s,1) = \mathbb{E}_\pi \left[\sum_{t \in \mathbb{N}_H} C_t \mid \mu \right] \le \hat{c}, \end{aligned}$$

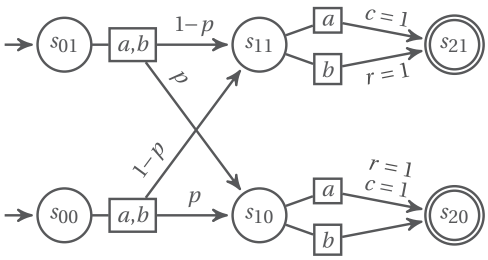


Figure 2.3: A CMDP with 6 states and two actions $\mathbb{A} = \{a, b\}$. Costs and rewards with value 0 are omitted as well as the probability of deterministic transitions.

where R_t and C_t are random variables indicating the reward and cost the agent receives at time step t, respectively. The expected cost of following a policy π starting from state s at time step t can be computed according to its occupancy measure y as follows:

$$V_C^\pi(s,t) = \sum_{s,a,k \in \mathbb{S}\times\mathbb{A}\times\{t,\cdots,H\}} y_k(s,a)C(s,a).$$

The expected value $V_R^\pi(s,t)$ is defined similarly replacing the cost function C by the reward function R.

Example 2 (The optimal policy for a CMDP)**.** *Consider the CMDP from Figure 2.3. In state s_{11} action b has no cost and gives a reward of 1, so the optimal policy always assigns $\pi(b \mid s_{11}) = 1$, which maximizes the reward and does not incur any cost. This way, all the cost would come from the state s_{10}: so we have $V_C^\pi(\mu) = p\pi(a \mid s_{10})$. Since only action a gives a reward in state s_{10}, the problem reduces to $\max \pi(a \mid s_{10})$ s.t. $p\pi(a \mid s_{10}) \leq \hat{c}$. For $p \leq \hat{c}$ the constraint cannot be violated, so the solution is $\pi(a \mid s_{10}) = 1$ which gives highest expected reward. For $p > \hat{c}$ we can reformulate the constraint, obtaining $\pi(a \mid s_{10}) \leq \frac{\hat{c}}{p}$. Since our objective is to maximize $\pi(a \mid s_{10})$, we find that the optimal policy for this problem is $\pi^*(a \mid s_{10}) = \frac{\hat{c}}{p}$.*

The following Linear Program (LP) solves a CMDP (Altman, 1999):

$$\max \sum_{s,a,t \in \mathbb{S}\times\mathbb{A}\times\mathbb{N}_H} y_t(s,a)R(s,a) \text{ s.t. C1–C6.} \tag{LP1}$$

$$\sum_{s,a,t \in \mathbb{S}\times\mathbb{A}\times\mathbb{N}_H} y_t(s,a)C(s,a) \leq \hat{c}. \tag{C1}$$

$$y_t(s,a) \geq 0 \qquad \forall s,a,t \in \mathbb{S}\times\mathbb{A}\times\mathbb{N}_H. \tag{C2}$$

$$y_t(s,a) = \sum_{s' \in \mathbb{S}} x_t(s,a,s') \qquad \forall s,a,t \in \mathbb{S}\times\mathbb{A}\times\mathbb{N}_H. \tag{C3}$$

$$\sum_{s^\circ,a^\circ \in \mathbb{S}\times\mathbb{A}} x_{t-1}(s^\circ,a^\circ,s) = \sum_{a\in\mathbb{A}} y_t(s,a) \qquad \forall s,t \in \mathbb{S}\times\mathbb{N}_H\setminus\{1\}. \tag{C4}$$

$$\sum_{a\in\mathbb{A}} y_1(s,a) = \mu(s) \qquad \forall s \in \mathbb{S}. \tag{C5}$$

$$x_t(s,a,s') = P(s' \mid s,a)y_t(s,a) \qquad \forall s,a,s',t \in \mathbb{S}\times\mathbb{A}\times\mathbb{S}\times\mathbb{N}_H. \tag{C6}$$

In LP1, C1 bounds the expected cost, C2 ensures the occupancy measure is positve, C3 shows the linear relation between y and x, C4 controls the inflow and outflow of each state at each time step, C5 is the initial state distribution and C6 ensures the flow respects the transition function. A solution for LP1 induces an optimal stochastic policy

$$\pi(a \mid s, t) = \frac{y_t(s,a)}{\sum_{a' \in \mathbb{A}} y_t(s,a')} \qquad \forall s, a, t \in \mathbb{S} \times \mathbb{A} \times \mathbb{N}_H. \tag{2.4}$$

We refer to de Nijs et al. (2021) for a comprehensive discussion of the optimization of CMDPs.

2.4. REINFORCEMENT LEARNING

Reinforcement Learning (RL; Sutton and Barto, 2018) is a research area concerned with sequential decision making where the dynamics of the environment are unknown. Intuitively, it considers an agent interacting with an environment that can be modeled as an MDP but without access to the transition and reward functions. In general, the agent must learn to optimize its long-term return by interacting with the environment. This raises a fundamental challenge in RL, the exploration-exploitation trade-off. Let us assume an agent already has performed some interactions with the environment and it has found a reasonable policy. Now it faces the question, should it *explore* new areas of the environment in the hope of learning more and finding even better policies, or should it *exploit* the current knowledge and keep executing the best policy it has at hand.

Numerous exploration strategies have been proposed to ensure the RL agent eventually finds an optimal policy (Amin et al., 2021; Tijsma et al., 2016). For instance, a Q-learning agent might perform random exploration, such that, with a small probability, it executes a uniformly random action; otherwise, it executes a greedy action with respect to its current knowledge (Watkins, 1989).

Typically, the performance of RL algorithms is evaluated by two metrics (Li, 2012):

i. the *sample complexity*, related to the number of interactions the agent requires to find an approximately optimal policy, and

ii. the *regret*, the difference between the performances of the policy executed and the optimal policy.

The exploration strategy and performance of an RL algorithm are tightly connected to how the agent computes its policy and the representations it uses. Overall, there are two main approaches to update the policy: model-based and model-free (Sutton and Barto, 2018, Chapter 8). In short, a *model-based* algorithm creates an estimate of the MDP, which is used to compute the policy through planning, while a *model-free* approach computes the policy directly from the experiences with the environment.

Although model-based RL has a computational cost higher than model-free RL, both in terms of memory, since it stores the model, as in terms of processing, since it includes a planning phase, it provides many benefits that may offset these drawbacks (Moerland et al., 2021, Section 7). For instance, it provides good *data efficiency* requiring a limited number of interactions with the environment to find a near-optimal policy (Brafman and Tennenholtz, 2002; Kearns and Singh, 2002). It also allows an agent to *transfer* knowledge from a source task to a similar target task, for instance, through the transition estimate when only the reward function changes (Atkeson and Santamaría, 1997), or by reusing experiences from the source task to estimate the model of the target task (Taylor et al., 2008). Furthermore, such a strategy can also provide some *generalization* capabilities, for example, allowing an agent to make predictions of unseen states and recover from unexplored regions of the environment (Ponnambalam et al., 2021).

In the following sections, we will see how we can estimate the model of the environment and how this estimate can guide exploration. Later chapters further explore the data efficiency of model-based approaches and exemplify other benefits, such as reliability and safety.

2.4.1. Model Learning

In this section, we review methods to estimate the transition function of an MDP from experiences, focusing on how they can exploit the structure of the environment. In the follow-up section, we will see how these methods can be used to develop sample-efficient RL algorithms.

One way to estimate such models is using the Knows What It Knows (KWIK; Li et al., 2011) framework, developed to help active agents stay aware of under-explored parts of the environment. A KWIK learner must only return ϵ-accurate predictions with high probability $1-\delta$, otherwise it should return "I do not know" ($\oslash$). However, a KWIK learner can only return $\oslash$ a limited number of times bounded by a function polynomial in the parameters of the problem. We say an algorithm is *KWIK-admissible* if it satisfies such requirements.

This way, a KWIK learner can be used by model-based RL algorithms to learn the transition function of a flat MDP or the transition components of an FMDP. Furthermore, this framework is particularly suitable to model-based RL since it explicitly indicates the *epistemic* uncertainty of the model, that is, the lack of knowledge about certain parts of the environment (Hüllermeier and Waegeman, 2021). In other words, a KWIK learner provides the means to learn the dynamics of the environment and is able to indicate which regions of the state space can be further explored.

When queried for the transition function of a state-action s,a a KWIK algorithm $\mathscr{K}$ returns the estimated distribution or $\oslash$ depending if s,a is known or not:

$$\mathscr{K}(s,a) = \begin{cases} \widehat{P}(\cdot \mid s,a) & \text{if } (s,a) \in \mathbb{K} \\ \oslash & \text{otherwise,} \end{cases}$$

where $\mathbb{K} \subseteq \mathbb{S} \times \mathbb{A}$ is the set of known state-action pairs and $\widehat{P}(\cdot \mid s,a)$ is the Maximum Likelihood Estimate (MLE) of the transition function of the state action pair $s,a \in \mathbb{S} \times \mathbb{A}$.

In the remainder of this section, we describe how to define $\mathbb{K}$ and $\widehat{P}(\cdot \mid s,a)$ depending on the structure of the problem and the available prior knowledge.

Learning (flat) MDPs

To estimate the transition of a flat MDP, we must keep track of two types of counters:

- $\eta(s,a)$: number of times action $a \in \mathbb{A}$ was executed on state $s \in \mathbb{S}$, and

- $\eta(s,a,s')$: number of times state $s' \in \mathbb{S}$ was observed after executing action $a \in \mathbb{A}$ on state $s \in \mathbb{S}$.

Then, the MLE of the transition function is defined as:

$$\widehat{P}(s' \mid s,a) = \frac{\eta(s,a,s')}{\eta(s,a)}, \qquad \forall s,a,s' \in \mathbb{S}\times\mathbb{A}\times\mathbb{S}. \tag{2.5}$$

In this case a KWIK learn algorithm defines the set of known state-action pairs as follows:

$$\mathbb{K}_m = \left\{ (s,a) \in \mathbb{S}\times\mathbb{A} \,\middle|\, \eta(s,a) \geq m \right\}, \tag{2.6}$$

where m is a minimum number of observations to consider the state-action pair (s,a) as known, defined according to the required precision ϵ and the expected level of confidence $1-\delta$.

LEARNING FACTORED MDPS

To estimate the dynamics of an FMDP, we will first consider the case where the structure is given, that is, we know the parents of each state variable and action pairs $\mathrm{Pa}_a(X)$. We start showing how to estimate the dynamics of individual components and later show how to combine them to estimate the full transition function.

We use two types of counters to estimate the CPT of a variable $X \in \mathbb{X}$ and action $a \in \mathbb{A}$:

- $\eta(\mathbf{x},a)$: number of times action $a \in \mathbb{A}$ was executed on a state $s \in \mathbb{S}$ where the parents of X were on the configuration $\mathbf{x} \in \mathrm{dom}(\mathrm{Pa}_a(X))$, that is $s[\mathrm{Pa}_a(X)] = \mathbf{x}$, and
- $\eta(\mathbf{x},a,x')$: number of times a state $s' \in \mathbb{S}$ for which the value of X is $x' \in \mathrm{dom}(X)$ was observed after action $a \in \mathbb{A}$ was executed on a state $s \in \mathbb{S}$ where the parents of X have the configuration $\mathbf{x} \in \mathrm{dom}(\mathrm{Pa}_a(X))$, that is, $s'[X] = x'$ and $s[\mathrm{Pa}_a(X)] = \mathbf{x}$.

Using these counters we can estimate the distribution of each entry of the CPT:

$$\widehat{P}(x' \mid \mathbf{x},a) = \frac{\eta(\mathbf{x},a,x')}{\eta(\mathbf{x},a)}, \qquad \forall (x',\mathbf{x}) \in \mathrm{dom}(X)\times\mathrm{dom}(\mathrm{Pa}_a(X)). \tag{2.7}$$

Now, we can define a KWIK algorithm $\mathscr{K}_{X,a}$ that learns the distribution of a component of the FMDP related to the variable-action pair $(X,a) \in \mathbb{X}\times\mathbb{A}$. Given a state $s \in \mathbb{S}$, depending on the counter $\eta(s[\mathrm{Pa}_a(X)],a)$, this algorithm returns the estimated distribution or $\oslash$:

$$\mathscr{K}^{m}_{X,a}(s) = \begin{cases} \oslash & \text{if } \eta(s[\mathrm{Pa}_a(X)],a) < m, \\ \widehat{P}(\cdot \mid s[\mathrm{Pa}_a(X),a]), & \text{otherwise,} \end{cases}$$

where m is defined according to the precision and confidence level required.

Next we need to define a way to estimate the transition function of the MDP. With that purpose, we define a new KWIK algorithm that uses a subalgorithm $\mathscr{K}_{X,a}$ for each variable-action pair $(X,a) \in \mathbb{X}\times\mathbb{A}$. This algorithm only considers a state-action pair $(s,a) \in \mathbb{S}\times\mathbb{A}$ as known, if all the subalgorithms related to the action a consider s known:

$$\mathbb{K}_{\vec{m}} = \left\{ (s,a) \in \mathbb{S}\times\mathbb{A} \,\middle|\, \mathscr{K}^{\vec{m}_i}_{X_i,a}(s) \neq \oslash, \forall X_i \in \mathbb{X} \right\}. \tag{2.8}$$

Finally, replacing each transition component from Equation 2.1 by the respective estimate, we obtain a new estimate of the known components of the transition function:

$$\widehat{P}(s' \mid s, a) = \prod_{X \in \mathbb{X}} \widehat{P}(s'[X] \mid s[\mathrm{Pa}_a(X)], a), \qquad \forall s, a, s' \in \mathbb{K}_{\vec{m}} \times \mathbb{S}. \tag{2.9}$$

Similar to the KWIK algorithm for flat MDPs, when queried for the transition function of a state-action $s, a \in \mathbb{S} \times \mathbb{A}$ the KWIK algorithm of an FMDP returns the estimated distribution or $\oslash$ depending if s, a is known or not:

$$\mathscr{K}(s,a) = \begin{cases} \widehat{P}(\cdot \mid s, a) & \text{if } (s,a) \in \mathbb{K} \\ \oslash & \text{otherwise.} \end{cases}$$

In the next section, we present algorithms that can be used when the structure of the problem is unknown.

Structure Learning

When the structure is unknown the subalgorithm $\mathscr{K}_{X,a}$ needs to initially search for the set of parents $\mathrm{Pa}_a(X)$. This problem has been extensively studied (Chakraborty and Stone, 2011; Degris et al., 2006; Diuk et al., 2009; Strehl et al., 2007). Here we present two structure learning algorithms with KWIK guarantees.

Given a maximum in-degree d, which bounds the size of the set of parents $\mathrm{Pa}_a(X)$, these algorithms need to choose a candidate from $\mathrm{C}_d^{\mathbb{X}}$ that contains the true set of parents, where $\mathrm{C}_d^{\mathbb{X}}$ is the collection of subsets of $\mathbb{X}$ of size d. For example, Figure 2.4 shows the set of possible parents for a given state variable and action. Given the maximum in-degree d, the structure learning algorithm can consider only the candidates to the respective column of the diagram.

The *Structure Learning* (SL) algorithm (Strehl et al., 2007) relies on the fact that the probability distribution of the variable X and action a is independent of non-parents given the true parents. To choose the set of parents the SL algorithm estimates the distribution for each pair of candidates $(\Delta_i, \Delta_j) \in \mathrm{C}_d^{\mathbb{X}} \times \mathrm{C}_d^{\mathbb{X}}$. Given a state s and action a, this algorithm chooses Δ_i as parents if two conditions are met:

i. it has collected enough samples of the realizations of Δ_i, a and all the other candidates were in the same configuration as in state s:

$$\eta(s[\Delta_i \cup \Delta_j], a) \geq m, \qquad \forall \Delta_j \in \mathrm{C}_d^{\mathbb{X}} \setminus \{\Delta_i\}, \text{ and} \tag{2.10}$$

ii. the distribution estimate of all pairs of candidates that include Δ_i are similar, that is, the distribution divergence between different pairs that include Δ_i is smaller than a given ϵ_1:

$$\left\| \widehat{P}(\cdot \mid s[\Delta_i \cup \Delta_j], a) - \widehat{P}(\cdot \mid s[\Delta_i \cup \Delta_k], a) \right\|_1 \leq \epsilon_1, \qquad \forall j, k \neq i. \tag{2.11}$$

In this way, given a state-action pair s, a, the SL algorithm returns the probability distribution of X if there is a candidate Δ_i that satisfies (denoted by $\models$) both conditions:

$$\mathscr{K}_{X,a}(s) = \begin{cases} \oslash & \text{if } \nexists \Delta_i \in \mathrm{C}_d^{\mathbb{X}} \text{ s.t. } \Delta_i \models (2.10) \text{ and } \Delta_i \models (2.11), \\ \widehat{P}(\cdot \mid s[\Delta_i \cup \Delta_j], a) \text{ where } j \neq i, & \text{otherwise.} \end{cases}$$

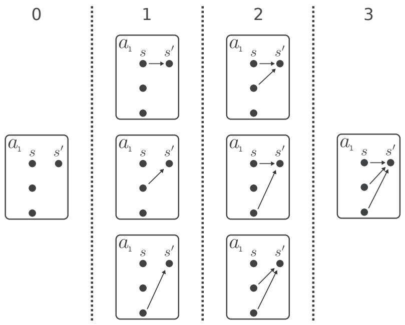


Figure 2.4: Set of candidate parents for an arbitrary state variable and action, grouped by size.

The *k-meteorologists* algorithm (Diuk et al., 2009) makes the same assumptions as the SL algorithm and initializes the set of tracked candidates Cand with $C_d^{\mathbb{X}}$. However, it relies on a different fact to find the best candidate: the squared error of the distribution function computed according to a candidate containing the true parents is smaller than the one computed without the true parents.

Given a transition sample $(s, a, s') \in \mathbb{S} \times \mathbb{A} \times \mathbb{S}$, the squared error of the set of parents $\Delta \subseteq \mathbb{X}$ is given by

$$e = (1 - \widehat{P}(s'[X] \mid s[\Delta], a))^2.$$

Therefore, this algorithm keeps track of the accumulated squared error for each pair of candidate parents and their number of mismatches $c_{i,j}$. After two candidates have disagreed enough times ($c_{i,j} \geq c$), the k-meteorologists algorithm discards the one with the largest accumulated error. When asked for the distribution of variable X_i for a given state-action pair, the k-meteorologists algorithm returns the average of the probability distribution of the remaining candidates (Cand), if they are confident in the distribution ($\eta(s[\Delta_i], a) \geq m$) and if they agree on such:

$$\left\| \widehat{P}(\cdot \mid s[\Delta_i], a) - \widehat{P}(\cdot \mid s[\Delta_j], a) \right\|_1 < \epsilon_1, \qquad \forall \Delta_i, \Delta_j \in \text{Cand} \times \text{Cand};$$

otherwise, it returns $\oslash$.

Both the SL and k-meteorologists algorithms can estimate the FMDP's transition function following the scheme from Equations 2.8 and 2.9 to combine the estimate of all the state variables. In the next section, we describe RL agents that use the algorithms from this section to explore the environment effectively.

2.4.2. Model-based Exploration

Sophisticated exploration strategies can reduce the number of interactions necessary to find an optimal policy. For instance, the R-max algorithm (Brafman and Tennenholtz, 2002) is a model-based RL algorithm that incentivizes the agent to visit under-explored parts of the environment to find an accurate estimate of the transition function quickly. The R-max, proposed for flat representations, has been adapted for different classes of MDPs (Guestrin et al., 2003).

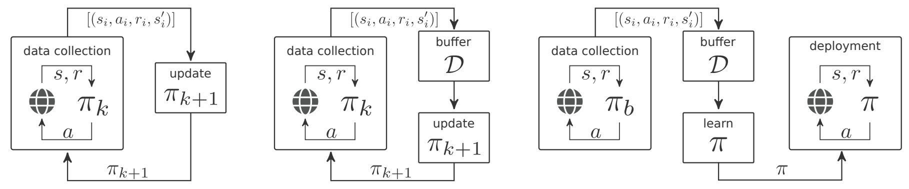
Algorithm 2 shows a generalization of the R-max algorithm coupled with a KWIK algorithm to estimate the transition function and keep track of the set of state-action pairs

Algorithm 2 KWIK-R-max (Li et al., 2011).

```
Input: $\mathscr{K}$: a KWIK algorithm with precision $\epsilon$ and confidence $\delta$.
 1: for $s, a \in \mathbb{S} \times \mathbb{A}$ do
 2:     Initialize $\mathscr{K}(s,a)$
 3:     $R_{\text{total}}(s,a) \leftarrow 0$
 4:     $\eta(s,a) \leftarrow 0$
 5: end for
 6: for $t \in 1, 2, \cdots$ do
        ▷ Update the empirical MDP $\widehat{\mathscr{M}}$
 7:     for $s, a \in \mathbb{S} \times \mathbb{A}$ do
 8:         if $\mathscr{K}(s,a) = \varnothing$ then
 9:             $\widehat{P}(s' \mid s,a) \leftarrow \begin{cases} 1 \text{ if } s' = s \\ 0 \text{ othewise} \end{cases} \quad : \forall s' \in \mathbb{S}$
10:             $\widehat{R}(s,a) \leftarrow R^{\top}$
11:         else
12:             $\widehat{P}(\cdot \mid s,a) \leftarrow \mathscr{K}(s,a)$
13:             $\widehat{R}(s,a) \leftarrow \frac{R_{\text{total}}(s,a)}{\eta(s,a)}$
14:         end if
15:     end for
        ▷ Compute new policy.
16:     $\pi \leftarrow$ Value Iteration($\widehat{\mathscr{M}}$)
        ▷ Interact with environment.
17:     Observe the current state of the environment $s_t$
18:     Execute action $a_t \sim \pi(\cdot \mid s_t)$
19:     Observe reward $r_t$ and new state of the environment $s_{t+1}$
        ▷ Update counters.
20:     $\eta(s_t, a_t) \leftarrow \eta(s_t, a_t) + 1$
21:     $R_{\text{total}}(s_t, a_t) \leftarrow R_{\text{total}}(s_t, a_t) + r_t$
22:     Present $s_{t+1}$ to algorithm $\mathscr{K}(s_t, a_t)$
23: end for
```

that are still considered unknown (Li et al., 2011). This KWIK algorithm must be chosen according to the class of the underlying MDP. The overall strategy of this algorithm is to incentivize the agent to visit states that can reduce the parametric uncertainty of the model. Therefore, it computes a policy (Line 16) using an augmented version of the estimated MDP $\widehat{\mathscr{M}}$ assuming that unknown state-action pairs provide high reward (Line 10). This way, the agent may visit such state-action pairs, improving the estimate of the respective transition function (Line 22). Although we describe this algorithm with an explicit estimate of the reward function, this task could also be executed by a KWIK algorithm.

R-max has a sample complexity polynomial in the number of states, which means exponential in the number of state variables. Factored RL algorithms can have a sample complexity that scales only polynomially in the number of parameters of the FMDP (Guestrin et al., 2002; Kearns and Koller, 1999; Strehl, 2007). That is, an RL agent that knows the structure of its environment dynamics can take less sub-optimal actions. R-max's extensions also have been proposed for FMDPs with different assumptions regard-



(a) On-policy. (b) Off-policy. (c) Offline.

Figure 2.5: Types of reinforcement learning algorithms (adapted from Levine et al., 2020).

ing the prior knowledge about the structure of the DBN. As we have seen in Section 2.4.1, Strehl et al. (2007) and Diuk et al. (2009) consider settings where the structure is unknown, assuming a bound on the number of parents. Chakraborty and Stone (2011) also consider settings where the structure is unknown, assuming the MDP is ergodic, also called a uni-chain MDP (Gattami et al., 2021), where all states are reachable from any state following any policy.

There are also more refined exploration methods based on the model estimate. For instance, the agent might choose optimistically from a set Σ of probable MDPs (Auer and Ortner, 2006; Jaksch et al., 2010; Strehl and Littman, 2008). This method is similar to solving bounded-parameter Markov decision processes optimistically (Givan et al., 2000). Since the size of Σ is inversely proportional to the visits counter, the incentive to visit each state varies according to the number of times the state has been visited. Jonsson and Barto (2007) propose an exploration strategy to learn the DBN structure of the FMDPs based on the local entropy gain of each action.

2.4.3. FROM ONLINE TO OFFLINE

We can categorize RL algorithms according to how they use the experiences (transitions or trajectories) collected to update the policy (Levine et al., 2020).

- *On-policy* algorithms update the policy immediately after each experience is collected. This way, the policy is always adjusted using data collected following the current policy. An example is the SARSA algorithm (Rummery and Niranjan, 1994).

- *Off-policy* algorithms store the experiences and update the policy in the future. In this setting, the data used to update the current policy may have been collected from a different policy, which can increase the variance of the algorithm. An example is the Q-learning algorithm (Watkins, 1989).

- *Offline* algorithms are designed for settings where the agent does not have direct interactions with the environment. In particular, the agent only has access to experiences collected by a behavior policy. An example is the Fitted Q Iteration algorithm (Ernst et al., 2005).

Figure 2.5 shows the schematics of each category. In Figure 2.5a, after a few episodes, the agent uses the trajectories collected to update the policy. In Figure 2.5b, the agent stores the trajectories in a buffer, which is used to update the policy, effectively allowing the agent to use older trajectories to update the current policy. Finally, in Figure 2.5c, a

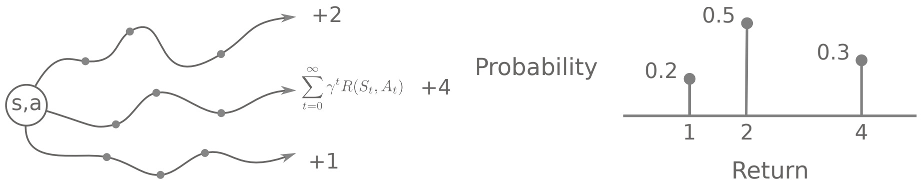


Figure 2.6: Trajectories generated by a policy, inducing a distribution of returns (adapted from Munos, 2018).

buffer of trajectories $\mathscr{D}$ is collected using a *behavior policy* π_b, which is fed to the agent. Then, the agent must compute a new policy without getting feedback from the environment. After the agent finishes, the new policy is deployed in the environment.

Both on-policy and off-policy algorithms can be considered online since the agent eventually receives some feedback from the environment in both cases. The model-based algorithms described in the previous section may be considered off-policy since the model's estimate works as a buffer of experiences.

We could use an off-policy algorithm directly on the offline setting; however, an optimistic approach, such as the one used by the R-max algorithm, could return a policy inadequate for the offline setting that would direct the agent to unknown parts of the environment. Another challenge in the offline setting is the overestimation of the values in parts of the problem that have not been seen enough, which can be propagated, causing a large variance in the performance of the algorithms (Fujimoto et al., 2019; Jin et al., 2021). We will investigate this issue further in Chapters 3 and 4.

2.5. SAFE RL

In RL, the concept of safety has been used extensively. It may refer to the most intuitive idea of avoiding undesirable outcomes, but it can also refer to how reliable the policy executed by the agent is (García and Fernández, 2015; Pecka and Svoboda, 2014). In the following sections, we discuss both directions from the perspective of this thesis.

2.5.1. ALTERNATIVE CRITERIA

Traditionally, RL optimizes the expected return. However, this criterion neglects other attributes on the return distribution, such as the variance and the worst case. To account for these aspects, we can use alternative criteria that make the policy more reliable and stable.

RL agents must handle two sources of uncertainty (Clements et al., 2019; Rigter et al., 2021). For each, there are different criteria that lead to more reliable performance.

i. The *aleatoric uncertainty* is related to the inherent randomness of the environment. This means a policy may generate different trajectories, which induces a distribution of returns, see Figure 2.6 for an example. A high return variance implies a larger risky so a risk-averse agent might prefer policies with lower variance in the return distribution (Gosavi, 2009). Another option is to bound the tail of the return distribution, which can be formalized with the conditional value-at-risk criterion (Chow et al., 2018a; Yang et al., 2021, 2022).

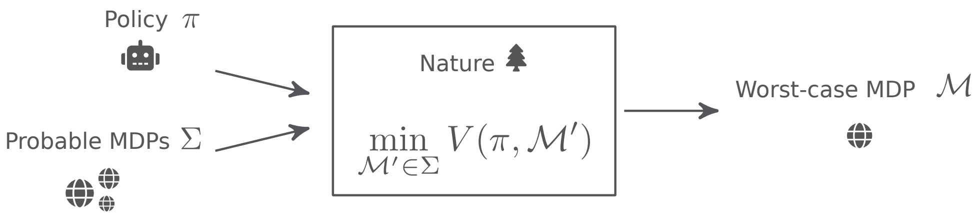

Policy π
Probable MDPs Σ
Nature
$\min_{\mathcal{M}' \in \Sigma} V(\pi, \mathcal{M}')$
Worst-case MDP $\mathcal{M}$

Figure 2.7: Pessimistic perspective of an RL agent with epistemic uncertainty. The agent has a set of probable MDP and assumes nature will choose the worst.

ii. The *epistemic uncertainty* is related to the agent's lack of knowledge about the dynamics of the environment, in other words, the model uncertainty (Sharma et al., 2019). As the agent interacts with the environment and collects new observations, the epistemic uncertainty can be reduced. One may imagine that the epistemic uncertainty induces a set of probable environments Σ (Figure 2.7). In this case, a conservative agent might prefer a robust policy, assuming that nature will choose the worst environment from Σ (Lim et al., 2016; Wiesemann et al., 2013).

The epistemic uncertainty is an inherent issue in the offline setting (Figure 2.5c). Since the agent only has access to a fixed dataset of past trajectories, it is only able to reduce its epistemic uncertainty to a limited degree. This limitation makes it challenging to ensure a policy computed offline will have a reasonable performance when deployed. Such lack of confidence could prevent decision makers from effectively deploying a policy proposed by an offline RL algorithm. Fortunately, an emergent area of RL is developing reliable agents guaranteed to return policies with reasonable performance (Thomas et al., 2015b).

There are two main strategies to handle these issues (Jin et al., 2021):

S.*i.* constraining the new policy to stay close to the policy that collected the data, and

S.*ii.* penalizing actions that appear less frequently in the dataset.

In Chapters 3 and 4, we focus on the first strategy (S.*i.*), considering the reliability of the offline algorithms with respect to the *behavior policy* π_b. In particular, we will consider the setting where both the experiences with the environment $\mathscr{D}$ and the behavior policy π_b are available to compute a new policy π'. We will investigate how to ensure the new policy π' does not underperform compared to π_b.

Another way to compute a policy more reliable is to define constraints explicitly, for instance, using CMDPs (Section 2.3). This model allows us to impose constraints on the behavior of the agent through the cost function. When the components of the CMDP are unknown (Geibel, 2006), we call it the Constrained Reinforcement Learning (CRL) problem. Since the policy with the highest expected return may violate the constraints, the agent may also need to sacrifice some of its performance to achieve safe behavior, meaning this method changes the optimality criterion. This problem has also been investigated from the safe exploration perspective, where the cost function is interpreted as a safety signal, for instance, indicating that the agent entered an unsafe state (Calvo-Fullana et al., 2021), as we discuss in the next section.

2.5.2. SAFE EXPLORATION

In certain applications, physical constraints must be taken into account. For instance, while controlling a robotic arm, we must avoid overheating. These constraints can create challenges around interacting directly with the environment in an online setting since the only way a vanilla RL observes the effects of a particular policy is by executing it. Therefore, to avoid violating the safety constraints, one often must use some prior knowledge, which allows the agent to infer if a specific policy is safe or not.

Such prior knowledge can take multiple forms. An agent *learning from demonstrations* gets early access to promising and safe trajectories, which reduces the time randomly exploring the environment and the risks involved in doing so (Argall et al., 2009; Ravichandar et al., 2020). The *advice from a teacher* can provide a set of safe actions for the agent to choose from or correct the actions taken by the agent (Alshiekh et al., 2018; Jansen et al., 2020). The agent can use an *initial safe policy* to collect trajectories until it is confident enough to deploy a new policy (Berkenkamp et al., 2017) or, as a fallback mechanism, in case the agent visits unexpected states (Mao et al., 2019).

In Chapter 5, we consider the setting where knowledge about the *safety dynamics* is provided, so the agent can evaluate the safety of a policy and avoid violating the safety constraints in a CMDP (Section 2.3). In this setting, we consider a policy with an expected cost larger than $\hat{c}$ unsafe. Our goal is to ensure that, at any episode, the policy executed does not violate the cost-bound.

2.6. SUMMARY

This chapter presents a short introduction to data-driven sequential decision making from a probabilistic planning perspective using a model-based approach. We refer to Mausam and Kolobov (2012) for alternative probabilistic planning algorithms and Sutton and Barto (2018) for a comparison between model-based and model-free in reinforcement learning. Finally, García and Fernández (2015) extensively discuss the different perspectives on safe reinforcement learning.

3

SAFE POLICY IMPROVEMENT IN FACTORED ENVIRONMENTS

In this chapter, we investigate how safe policy improvement (SPI) algorithms can exploit the structure of factored Markov decision processes. To facilitate the application of reinforcement learning in the real world, SPI provides probabilistic guarantees that policy changes in a running process will improve the performance of this process. However, most SPI algorithms require large amounts of historical data to become confident enough to change the policy. Factored reinforcement learning, on the other hand, makes good use of the data provided. These algorithms achieve better sample complexity by exploiting independence between features of the environment, but they lack improvement guarantees.

We propose a factored SPI algorithm that improves the sample efficiency of the safe policy improvement with baseline bootstrapping algorithm by exploiting the factored structure of the environment. Our first result is a theoretical bound that is linear in the number of parameters of the factored representation instead of the number of states. The empirical analysis shows that our method can improve the policy using a number of samples potentially one order of magnitude smaller than the flat counterpart.

This algorithm requires prior knowledge of the underlying structure, therefore, to overcome this limitation, we enhance it with different structure learning methods. In well-factorized domains, the new algorithms need fewer samples to improve the policy compared to a flat SPI algorithm, demonstrating a sample complexity closer to the factored SPI algorithm that knows the structure. This indicates that the combination of factored SPI and structure learning algorithms is a promising solution to real world problems involving many variables.

This chapter is based on papers presented at AAAI-19 and IJCAI-19 (Simão and Spaan, 2019a,b). An extended abstract appeared at the IJCAI-19 Doctoral Consortium (Simão, 2019).

As discussed in Sections 1.3 and 2.5, Safe Reinforcement Learning (SRL) aims to mitigate undesirable effects of RL algorithms (García and Fernández, 2015). In this chapter, we consider the safety of offline RL algorithms and investigate how to make them more reliable. In particular, we focus on the reliability of these algorithms compared to the policy used to collect the data. Concerning Research Question 1 on which class can facilitate the development of SRL, we constrain the problem to a Factored Markov Decision Process (FMDP). This allows us to address Research Question 2 regarding the amount of data these algorithms require and Research Question 3 on dealing with datasets that have low coverage of the problem.

In Chapter 2, we saw that Reinforcement Learning (RL; Sutton and Barto, 2018) is a framework for sequential decision-making and most RL research focuses on the online setting, where the RL agent interacts directly with the environment and can learn from the feedback it gets (Mnih et al., 2015; van Seijen et al., 2017). Nevertheless, learning from historical data can help bring RL to the real-world (Dulac-Arnold et al., 2021). While the online setting might be the most efficient in simulations and in uni-device system control such as drones or complex industrial flow optimization, many real world tasks involve a distributed architecture. We may cite a few: demand-supply balance in energy systems with distributed generation (Vázquez-Canteli and Nagy, 2019), choice of medication in health-care (Liu et al., 2020), caching in communication networks (Qian et al., 2019), and authentication of mobile devices in wireless networks (Liu et al., 2017; Uprety and Rawat, 2021). These tasks entail a high parallelization of the trajectory collection and strict communication constraints both in bandwidth and privacy (Féraud et al., 2019). Therefore, rather than spending a small amount of computation after collecting each sample/trajectory, it is more practical to collect a dataset using a behavior policy and then train a new policy from it. This setting is known as batch RL (Lange et al., 2012; Xie and Jiang, 2020), offline RL (Levine et al., 2020), fixed-dataset policy optimization (Buckman et al., 2021), and batch policy optimization (Xiao et al., 2021b).

3.1. Safety in Offline Reinforcement Learning

In the offline RL setting, the agent only has access to historical data collected with a *behavior policy* π_b, which might be unknown, as we will see in Chapter 4. In other words, the agent does not interact directly with the environment (Section 2.4.3). In this setting, there are two major tasks:

- *off-policy evaluation*, where, given a batch of past experiences, the RL agent must estimate the performance of a candidate policy π, and

- *policy optimization*, where, given a batch of past experiences, the RL agent must compute a candidate policy π.

Both these tasks have risks. In policy evaluation, we might overestimate the candidate's performance, while in policy optimization, we might compute a candidate with low performance. In these two scenarios, the system's performance could decrease compared to the performance during data collection. Therefore, it is essential to ensure that π outperforms the behavior policy π_b; otherwise, one would prefer to keep executing the behavior policy π_b to avoid such risks. From a reliability perspective, each of these tasks imposes different challenges.

For the off-policy evaluation problem, it is necessary to have confidence in the estimated performance, considering the random nature of the previous experiences. Different methods have been proposed to improve the confidence in off-policy evaluation algorithms using flat and factored representations of the problem (Hallak et al., 2015; Thomas et al., 2015a).

For the policy optimization task, the RL algorithm must compute a new policy at least as good as the behavior policy, and if it has a high probability of returning an improved policy, this algorithm is considered safe (Cohen et al., 2018; Laroche et al., 2019; Petrik et al., 2016; Thomas et al., 2015b). This problem is called Safe Policy Improvement (SPI; Thomas et al., 2015b), and it will be the focus of this chapter.

However, a major challenge for SPI algorithms is that they rely on flat representations, which limits their scalability. In particular, when a set of features describe the state space, the number of states grows exponentially in the number of features (Section 2.2). In this case, the number of samples necessary to estimate the model or the performance of a policy precisely might be prohibitive, making the application of flat algorithms infeasible.

As we observed in Section 2.4.2, factored reinforcement learning can exploit independence present in the environment and generalize past experiences to new states, which allows an online agent to reduce the number of non-optimal actions it takes (Degris et al., 2006; Ross and Pineau, 2008; Strehl et al., 2007). However, such algorithms have been proposed for online RL settings that ignore safety and in which an agent can explore freely, often following the *optimism in the face of uncertainty* principle (Munos, 2014), which guides exploration towards less-visited parts of the environment. Unfortunately, this strategy is not compatible with the offline setting since the agent does not get any feedback from the environment to reduce its epistemic uncertainty. This chapter aims to bridge the gap between safe and factored RL algorithms following a *pessimism in the face of uncertainty* principle (Buckman et al., 2021; Jin et al., 2021; Rashidinejad et al., 2021).

The first contribution of this chapter is a *factored SPI* algorithm that uses a factored representation to estimate the dynamics of the environment, assuming that the structure of the problem is known a priori. We prove that by exploiting independence between state variables, our safe RL better estimates the environment dynamics and requires fewer samples to improve the behavior policy. These results are demonstrated empirically in three experiments with different domains and behavior policies. A highlight of this algorithm is its capability to improve even when the behavior policy is deterministic, which is a substantial limitation of previous SPI algorithms.

Nevertheless, this factored approach assumes that the local structure is known a priori, which might be impractical. For instance, in applications with many variables, it is difficult for an expert to provide the exact structure of the problem. Therefore, it is necessary to investigate how to relax this assumption without resorting to flat representations to maintain a good sample complexity. We aim to develop SPI algorithms that are sample efficient even when the structure of the FMDP is unknown.

The problem of learning the structure has already been investigated from different perspectives. For example, algorithms have been proposed to improve exploration efficiency in the online setting (Diuk et al., 2009; Strehl et al., 2007). However, these methods do not consider the safety of the learning agent also following the optimism in the face of uncertainty, which is highly undesirable in a safe RL setting. Furthermore, off-policy evaluation algorithms also have been coupled with structure learning algorithms to reach higher accuracy (Hallak et al., 2015), but this approach does not indicate how the candidate policy is computed.

Our second contribution addresses safety in environments with an unknown structure. We propose an SPI framework that can use different structure learning algorithms to estimate the dynamics of the environment. We provide two algorithms that instantiate the new SPI framework using different structure learning algorithms and prove that they have safety guarantees. Our experiments compare these algorithms to the SPI algorithm that has access to the structure of the problem (Section 3.3) and to an SPI algorithm that ignores the underlying structure (Laroche et al., 2019). They show that, depending on the structure learning algorithm and how well the problem can be factorized, this framework can yield algorithms with performance competitive to an algorithm that knows the structure.

Chapter Structure. First, we formalize the SPI problem and review previous algorithms dealing with it (Section 3.2). Then, we present the new SPI algorithm for environments with known factored dynamics, proving that this method is safe, and provide an empirical evaluation of the new algorithm (Section 3.3). Next, we investigate the SPI problem on factored environments with an unknown structure (Section 3.4). To conclude the chapter, we describe a realistic case study that used these algorithms (Section 3.5) and present some final remarks (Section 3.6).

3.2. SAFE POLICY IMPROVEMENT

This section reviews the offline policy optimization problem and state-of-the-art methods to solve it, which will be extended in the follow-up sections to factored MDPs.

3.2.1. RELIABLE OFFLINE RL: AN OVERVIEW

Classically, offline RL algorithms apply dynamic programming on the samples in the dataset (Ernst et al., 2005; Lagoudakis and Parr, 2003). Laroche et al. (2019) showed that in finite-state MDPs, all these algorithms converge to the same policy: the one that is optimal in the MDP with the maximum likelihood given the batch of data. Petrik et al. (2016) show that this policy is approximately optimal to the order of the inverse square root of the minimal state-action pairs count in the dataset. Unfortunately, Laroche et al. (2019) show that, even on small tasks, this minimal amount is almost always zero, and that, as a consequence, it gravely impairs the reliability of the approach: naive dynamic programming on the batch may return policies that perform terribly in the real environment. If a poor-performing policy were executed in distributed architectures such as the ones mentioned above, the consequences would be disastrous as it would jeopardize a high number of systems or even lives.

Several attempts have been made to design reliable offline RL algorithms, starting with robust MDPs (Iyengar, 2005; Nilim and El Ghaoui, 2005), which considers the set of plausible MDPs Σ given the dataset, also called the *uncertainty set* or robust MDP. Overall, these methods search for the policy for which the minimal performance over Σ is maximal. However, this approach traditionally tends to return overly conservative policies (Russel and Petrik, 2019).

Xu and Mannor (2009) considered robust regret over the optimal policy: the algorithm searches for a policy that minimizes the maximal gap with respect to the optimal performance in every MDP of the uncertainty set Σ. However, they proved that even evaluating the robust optimal regret for a fixed policy is already NP-complete with respect to the state and action sets' size and the uncertainty constraints in Σ.

Later, Petrik et al. (2016) considered the regret with respect to the behavior policy performance over the uncertainty set Σ. The behavior policy is called *baseline* in this context. Similarly, they proved that simply evaluating the robust baseline regret is already NP-complete. Concurrently, they also proposed the Reward-adjusted MDP (RaMDP) algorithm following a strategy that penalizes less-visited state-action pairs (S.*ii*., Section 2.5.1), although without theoretical guarantees. In this algorithm, the immediate reward for each transition in the dataset is penalized by the inverse square root of the number of samples in the dataset that have the same state and action as the transition under consideration.

Recently, Laroche et al. (2019) proposed Safe Policy Improvement with Baseline Bootstrapping (SPIBB), the first tractable algorithm with approximate policy improvement guarantees. Its principle consists in guaranteeing safe policy improvement by constraining the trained policy as follows: it has to reproduce the baseline policy in the uncertain state-action pairs. This chapter follows this research track, developing SPIBB algorithms for FMDPs (Simão and Spaan, 2019a,b). Nadjahi et al. (2019) proposed Safe Policy Improvement with Soft Baseline Bootstrapping (Soft-SPIBB), which further improves SPIBB's empirical performance by adopting soft constraints. Note that this thread of research is distinct from online safe policy iteration, such as (Kakade and Langford, 2002; Papini et al., 2017; Pirotta et al., 2013; Schulman et al., 2015, 2017), because the online setting allows them to perform more conservative updates.

High Confidence Policy Improvement (HCPI; Mandel et al., 2014; Paduraru, 2013; Thomas et al., 2015b) is another theoretically-grounded and tractable family of frequentist algorithms that rely on importance sampling estimates of the trained policy performance. The algorithm by Mandel et al. (2014), based on concentration inequalities, tends to be conservative and requires hyper-parameter optimization. The algorithms by Thomas et al. (2015a) rely on the assumption that the importance sampling estimate is normally distributed, which is false when the number of trajectories is small. The algorithm by Paduraru (2013) is based on bias-corrected and accelerated bootstrap and tends to be too optimistic. In contrast to the robust approaches, from robust MDPs to Soft-SPIBB, HCPI may be readily applied to infinite MDPs with guarantees. However, it is well known that the importance sampling estimates have high variance, exponential with the horizon of the MDP. The SPIBB algorithm has a linear horizon dependency, given a fixed known maximal value and the common horizon/discount factor equivalence: $H = \frac{1}{1-\gamma}$ (Kocsis and Szepesvári, 2006). Soft-SPIBB suffers a cubic upper bound, but the empirical results indicate a linear dependency.

Nadjahi et al. (2019) performed a benchmark on randomly generated finite MDPs, baselines, and datasets. They report that the SPIBB and Soft-SPIBB algorithms are significantly the most reliable, tying with RaMDP as the highest average performing algorithms. Additionally, they perform a benchmark on a task with continuous state space, where the SPIBB and Soft-SPIBB algorithms significantly outperform RaMDP (Petrik et al., 2016) and Double-DQN (van Hasselt et al., 2016) both in reliability and average performance. Soft-SPIBB particularly shines in the continuous state experiments, which we revisit in Chapter 4.

3.2.2. Optimization Criterion

SPI addresses the question of how to compute a new policy π that outperforms the behavior policy π_b with high confidence $1-\delta$, given a batch of previous interactions $\mathscr{D}$ and an admissible error ζ. Before formalizing the safety criterion used in this chapter we present a few definitions.

An episode τ is a sequence of interactions between the agent following a policy π and the environment represented by $[s_t, a_t, r_t, s_{t+1}, \cdots]$ where $a_t \sim \pi(\cdot \mid s_t)$ is the action executed in the state s_t, $r_t = R(s_t, a_t)$ is the reward obtained and $s_{t+1} \sim P(\cdot \mid s_t, a_t)$ is the state observed afterwards. A dataset of previous experiences is represented by a set of N episodes $\mathscr{D} = \{\tau_i \mid i \in \mathbb{N}_N\}$, collected following a behavior policy π_b.

Let $\widehat{\mathscr{M}}$ be the maximum likelihood estimate of the underlying MDP built according to the past experiences $\mathscr{D}$ (Section 2.4.1). Let $e : \mathbb{S} \times \mathbb{A} \to \mathbb{R}$ be an arbitrary error function, such that $e(s,a)$ represents the uncertainty over the parameters of the estimated transition function $\hat{P}(\cdot \mid s,a)$. The uncertainty set $\Sigma(\widehat{\mathscr{M}}, e)$ is the set of MDPs with transition function P', such that the L_1 distance between $P'(\cdot \mid s,a)$ and $\hat{P}(\cdot \mid s,a)$ is smaller than $e(s,a)$ for every state-action pair, that is

$$\left\| \widehat{P}(\cdot \mid s,a) - P'(\cdot \mid s,a) \right\|_1 \leq e(s,a) \qquad \forall (s,a) \in \mathbb{S} \times \mathbb{A}.$$

Intuitively, we must define the error function e wide enough such that $\Sigma(\widehat{\mathscr{M}}, e)$ includes the true MDP with high probability $1-\delta$.

Laroche et al. (2019) proposed the SPIBB criterion, which is defined only over the maximum likelihood estimate of the MDP $\widehat{\mathscr{M}}$ and is easier to solve. According to this criterion, an RL algorithm is considered *safe* if, given a confidence level δ and a level of precision ζ, it has a high probability $1-\delta$ of returning a policy that is ζ-approximate as good as the behavior policy π_b on all MDPs in $\Sigma(\widehat{\mathscr{M}}, e)$:

$$\begin{aligned} &\max_{\pi \in \Pi} V(\pi, \widehat{\mathscr{M}}) \\ &\text{s.t. } \; V(\pi, \mathscr{M}') \geq V(\pi_b, \mathscr{M}') - \zeta \qquad \forall \mathscr{M}' \in \Sigma(\widehat{\mathscr{M}}, e). \end{aligned} \tag{3.1}$$

Note that an algorithm that always returns the behavior policy is considered safe according to this criterion. This line of work is motivated by situations where it is risky to apply a new policy, such as when the policy updates are infrequent, and any change in the behavior policy would represent a firm commitment.

Algorithm 3 Π_b-SPIBB.

```
Input: Previous experiences $\mathscr{D}$
Input: Parameters $\epsilon, \delta$
Input: Behavior policy $\pi_b$
Output: Safe Policy
1: Estimate $\hat{P}$                                ▷ Equation 2.5
2: Compute $\mathbb{B}_m = \overline{\mathbb{K}_m}$  ▷ Equation 2.6
3: Compute $\Pi_b$                                   ▷ Equation 3.2
4: return $\arg\max_{\pi \in \Pi_b} V(\pi, \widehat{\mathscr{M}})$
```

3.2.3. SPI WITH BASELINE BOOTSTRAPPING ALGORITHMS

The SPIBB framework is a model-based approach that guarantees safety by bootstrapping unknown parts of the approximated model with the behavior policy π_b (Laroche et al., 2019). Formally, the set of bootstrapped state-action pairs $\mathbb{B}_m$ is the complement of $\mathbb{K}_m$ (Equation 2.6). This way, the SPIBB algorithms guarantee to perform at least as well as the behavior policy and does not rely on a safety test, in contrast to other SPI algorithms.

The SPIBB algorithm has two variants that bootstrap the behavior policy in different ways. The *value-based* algorithm uses the estimated performance of the elements of $\mathbb{B}_m$ during the planning phase and if a state-action pair from $\mathbb{B}_m$ is used during execution, control is returned to the behavior policy. The *policy-based* Π_b-SPIBB algorithm attributes the same probability to bootstrapped pairs as the behavior policy, which restricts the policy space to

$$\Pi_b = \{\pi \in \Pi \mid \pi(a \mid s) = \pi_b(a \mid s) : \forall (s,a) \in \mathbb{B}_m\}. \tag{3.2}$$

Laroche et al. (2019) proved that if

$$m = \frac{2}{\epsilon^2} \log \frac{|\mathbb{S}||\mathbb{A}|2^{|\mathbb{S}|}}{\delta}, \tag{3.3}$$

then the Π_b-SPIBB algorithm is safe, where ϵ is a bound on the L_1 distance between the estimated transition function and the true transition function, that depends on the precision parameter ζ. We now recall the safe policy improvement guaranteed by the algorithm Π_b-SPIBB from Laroche et al. (2019).

Theorem 1 (Safe policy improvement with baseline bootstrapping by (Laroche et al., 2019))**.** *Let $\pi^{\odot}$ be an optimal policy constrained to Π_b in the MLE-MDP $\widehat{\mathscr{M}}$. Then, $\pi^{\odot}$ is a ζ-approximate safe policy improvement over the baseline π_b with high probability $1-\delta$, where:*

$$\zeta = \frac{4\epsilon V_{max}}{1-\gamma} - V(\pi^{\odot}, \widehat{\mathscr{M}}) + V(\pi_b, \widehat{\mathscr{M}}).$$

Algorithm 3 gives a brief description of the Π_b-SPIBB approach. The $\Pi_{\leq b}$-SPIBB algorithm is a variation of the Π_b-SPIBB algorithm where the constrained space of policies is defined as follows:

$$\Pi_{\leq b} = \{\pi \in \Pi \mid \pi(a \mid s) \leq \pi_b(a \mid s) : \forall (s,a) \in \mathbb{B}_m\}. \tag{3.4}$$

In this case, it is possible to reduce the probability attributed to bootstrapped actions if other actions that have already been sampled enough times have a better performance.

In their experimental analysis, Laroche et al. (2019) used a stochastic baseline policy with softmax exploration over the optimal value function. As expected, the Π_b-SPIBB algorithm displayed a safe behavior. Although the $\Pi_{\leq b}$-SPIBB algorithm has not been proven to be safe (in contrast to the Π_b-SPIBB algorithm), the experimental analysis showed that it could also have a safe behavior.

Since the Π_b-SPIBB and $\Pi_{\leq b}$-SPIBB algorithms can change the policy in only a subset of the state-action pairs, they were demonstrated to be less conservative than other SPI algorithms. Nevertheless, when a set of factors describes the problem, the state space grows exponentially in the number of factors, which implies exponential growth in the amount of data m (Equation 3.3) required to change a policy. The following section shows that, by taking into account the independence between state variables, it is possible to exploit the factored representation of the problem using a minimum number of samples that is only polynomial in the number of parameters of the FMDP.

3.3. FACTORED SPI WITH KNOWN STRUCTURE

This section shows how to adapt the SPIBB methodology to environments with factored dynamics, assuming that the dependence between the factors is known a priori, although the distribution of each factor is unknown. First, we describe how the first two steps of the policy-based SPIBB algorithm can be adapted to this setting. Next, we prove that this algorithm is safe.

3.3.1. FACTORED POLICY-BASED SPIBB

Algorithm 4 presents the Factored Π_b-SPIBB algorithm, an adaptation of Π_b-SPIBB algorithm for factored environments. Note that this algorithm takes an extra input: the dependency function Pa used to determine which transition components must be estimated ($\mathscr{Q}$).

First, the algorithm estimates each transition component according to $\mathscr{D}$ using the same counters as the factored R-max algorithm (Guestrin et al., 2002). The set of state-action pairs to be bootstrapped $\mathbb{B}_{\vec{m}}$ is the complement of the set of known state-action pairs $\mathbb{K}_{\vec{m}}$ (Equation 2.8). In the next section, we show how each value in $\vec{m}$ must be defined to ensure the safety of this algorithm. Given $\mathbb{B}_{\vec{m}}$ and the behavior policy π_b, the constrained policy space Π_b is computed using Equation 3.2. Finally, the algorithm searches for an optimal policy in Π_b, however, in this case the transition function $\hat{P}(\cdot \mid s, a)$ is estimated according to the estimate of each transition component (Equation 2.9).

Replacing Equation 3.2 by Equation 3.4 in Algorithm 4, we obtain the Factored $\Pi_{\leq b}$-SPIBB algorithm, the factored version of the $\Pi_{\leq b}$-SPIBB algorithm. As we mentioned before, Laroche et al. (2019) also proposed a value-based SPIBB algorithm. However, developing an effective factored version of this method would require a factored representation of the value function, which is typically not compactly factorized.

Algorithm 4 Factored Π_b-SPIBB.

```
Input: Previous experiences $\mathscr{D}$
Input: Parameters $\epsilon, \delta$
Input: Behavior policy $\pi_b$
Input: Dependency function Pa
Output: Safe Policy
1: Estimate $\hat{P}(\cdot \mid \mathbf{x}, a), \forall (X, a, \mathbf{x}) \in \mathscr{Q}$    ▷ Equation 2.7
2: Compute $\mathbb{B}_{\vec{m}} = \overline{\mathbb{K}_{\vec{m}}}$    ▷ Equation 2.8
3: Compute $\Pi_b$    ▷ Equation 3.2
4: return $\arg\max_{\pi \in \Pi_b} V(\pi, \widehat{\mathscr{M}})$
```

3.3.2. BENEFITS OF A FACTORED REPRESENTATION

There are two main benefits of exploiting factored representation in the safe RL setting that we are considering: reduced sample complexity and being able to generalize better from deterministic behavior policies. We detail both advantages below.

First, similar to the R-max algorithm (Brafman and Tennenholtz, 2002), these algorithms enumerate the set of states to define $\mathbb{B}_{\vec{m}}$ and compute a new policy. However, we would like to point out that the primary goal of the new factored SPI algorithms is to improve the precision of the estimated transition function, which allows these algorithms to become less conservative. Example 3 illustrates this point.

Example 3. *Consider the problem of controlling the temperature of three rooms, with temperatures T_1, T_2 and T_3. Because the first room only shares a wall with the second room, the next value of T_1 is conditionally independent of T_3 given T_1 and T_2. To estimate the future value of T_1 given $T_1 = v_1$, $T_2 = v_2$ and $T_3 = v_3$, all past experiences where $T_1 = v_1$ and $T_2 = v_2$ can be used, regardless of the temperature of T_3. This way, using a factored representation, a safe RL algorithm would need fewer samples to change how it controls the temperature of the first room since it has a better estimate of the environment dynamics.*

Second, we would like to point out that by using a factored representation, the factored SPI algorithms may choose an action a in a state s even if $\eta(s, a) = 0$, a typical case when the behavior policy is deterministic. In this case, a flat algorithm would never execute a in s. Therefore, the application of flat SPI algorithms is limited since they are not able to increase the probability of a if $\pi_b(a \mid s) = 0$. Example 4 clarify this property, and we demonstrate this feature in an experiment with a deterministic behavior policy.

Example 4. *Consider the dataset of past trajectories was collected following a deterministic behavior policy that always executes $\mathtt{a}' \neq \mathtt{a}$ in the state $\mathtt{s}$. In this case, $\eta(\mathtt{s}, \mathtt{a}) = 0$ and a flat SPI algorithm would always return a policy π where $\pi(\mathtt{a}' \mid \mathtt{s}) = 1$ and $\pi(\mathtt{a} \mid \mathtt{s}) = 0$. The generalization capabilities of a factored representation can deal with this limitation: the agent can estimate the transition components of each state variable $X \in \mathbb{X}$ using past experiences where action $\mathtt{a}$ was executed in other states $\mathtt{s}' \in \mathbb{S} \setminus \{\mathtt{s}\}$ where $\mathtt{s}'[Pa_{\mathtt{a}}(X)] = \mathtt{s}[Pa_{\mathtt{a}}(X)]$. Using the estimate of each component the agent can estimate the distribution over the successor states $\hat{P}(\cdot \mid \mathtt{s}, \mathtt{a})$ and, eventually, choose to execute $\mathtt{a}$ in $\mathtt{s}$.*

Factored representations may also help to tackle the off-policy policy evaluation problem. High Confidence Off-Policy Evaluation (HCOPE) is a model-free method based on importance sampling that estimates a candidate policy's performance with a given confidence level (Thomas et al., 2015a). From a model-based perspective, Hallak et al. (2015) showed that using a factored representation to estimate the model, one can find accurate estimates of the performance of a target policy. Thomas et al. (2015b) proposed a model-free approach for the SPI problem. It divides the dataset in two partitions $\mathscr{D}_{\text{train}}$ and $\mathscr{D}_{\text{test}}$. Using $\mathscr{D}_{\text{train}}$, this algorithm computes a new policy and, to ensure safety, it checks with the HCOPE test if the new policy is safe using $\mathscr{D}_{\text{test}}$, following a cross-validation approach. If the computed policy is not considered safe, it indicates that no improved policy was found, in which case the RL agent could keep executing the behavior policy. Following a similar strategy, the method proposed by Hallak et al. (2015) could be used to exploit the factored dynamics.

3.3.3. THEORETICAL ANALYSIS

In this section, we show that given the admissible error parameter ζ, the Factored Π_b-SPIBB algorithm satisfies the SPIBB safety criterion (Equation 3.1).

First, we show that the error of the transition function is bounded with high probability in all state-action pairs that are not bootstrapped by the Factored Π_b-SPIBB algorithm. With Corollary 1 by Strehl (2007) we can bound the L_1 distance between the estimated transition function computed by the product of a set of components and the actual transition function, given that the error of each component is bounded.

Lemma 1 (Corollary 1 by Strehl (2007)). *Let $\mathscr{M}$ be any FMDP. Suppose that for each transition component $P(\cdot \mid \mathbf{x}, a)$ we have an estimate $\hat{P}(\cdot \mid \mathbf{x}, a)$ such that*

$$\left\| P(\cdot \mid \mathbf{x}, a) - \hat{P}(\cdot \mid \mathbf{x}, a) \right\|_1 \leq \frac{\epsilon}{|\mathbb{X}|}.$$

Then, for all state action pairs

$$\left\| P(\cdot \mid s, a) - \hat{P}(\cdot \mid s, a) \right\|_1 \leq \epsilon.$$

Next, we redefine Proposition 3 by Laroche et al. (2019) for the case where the transition function is estimated according to the estimate of each transition component (Equation 2.9).

Proposition 1. *Consider an environment modeled by a semi-MDP $\mathscr{M}$ (Sutton et al., 1999) and the empirical semi-MDP $\widehat{\mathscr{M}}$ estimated from a dataset $\mathscr{D}$. If in every state s where option o_a*[1] *may be initiated, $s \in \mathscr{I}_a$, we have that for all relevant components $(X, a, \mathbf{x}) \in D_{s,a}$ (Equation 2.3)*

$$\sqrt{\frac{2|\mathbb{X}|^2}{\eta(\mathbf{x}, a)} \log \frac{|\mathscr{Q}| 2^{|dom(X)|}}{\delta}} \leq \epsilon, \tag{3.5}$$

then holds that

$$\Pr\left(\left\| P(\cdot \mid s, a) - \hat{P}(\cdot \mid s, a) \right\|_1 \geq \epsilon\right) \leq \delta, \qquad \forall (s, a) \notin \mathbb{B}_{\vec{m}}.$$

[1]Option o_a is the counterpart of the original action a in the semi-MDP.

Proof. Using Weissman et al. (2003)'s Theorem 2.1, for each transition component $(X, a, \mathbf{x}) \in \mathscr{Q}$ (Equation 2.2), we may write:

$$\begin{aligned}
&\Pr\left(\left\|P(\cdot \mid \mathbf{x}, a) - \hat{P}(\cdot \mid \mathbf{x}, a)\right\|_1 \geq \tilde{\epsilon}\right) \\
&\qquad \leq (2^{|\mathrm{dom}(X)|} - 2)\exp\left(-\frac{\eta(\mathbf{x}, a)\tilde{\epsilon}^2}{2}\right).
\end{aligned} \tag{3.6}$$

This equation bounds the error of each transition function component. To use Lemma 1, we set $\tilde{\epsilon} = \frac{\epsilon}{|\mathbb{X}|}$ and rewrite Equation 3.6 as

$$\begin{aligned}
\Pr&\left(\left\|P(\cdot \mid \mathbf{x}, a) - \hat{P}(\cdot \mid \mathbf{x}, a)\right\|_1 \geq \frac{\epsilon}{|\mathbb{X}|}\right) \\
&\leq (2^{|\mathrm{dom}(X)|} - 2)\exp\left(-\frac{\eta(\mathbf{x}, a)\epsilon^2}{2|\mathbb{X}|^2}\right) \\
&\leq 2^{|\mathrm{dom}(X)|}\exp\left(-\frac{\eta(\mathbf{x}, a)}{2|\mathbb{X}|^2}\frac{2|\mathbb{X}|^2}{\eta(\mathbf{x}, a)}\log\frac{|\mathscr{Q}|2^{|\mathrm{dom}(X)|}}{\delta}\right) \\
&= 2^{|\mathrm{dom}(X)|}\exp\left(-\log\frac{|\mathscr{Q}|2^{|\mathrm{dom}(X)|}}{\delta}\right) \\
&= 2^{|\mathrm{dom}(X)|}\exp\left(\log\frac{\delta}{|\mathscr{Q}|2^{|\mathrm{dom}(X)|}}\right) \\
&= 2^{|\mathrm{dom}(X)|}\frac{\delta}{|\mathscr{Q}|2^{|\mathrm{dom}(X)|}} \\
&= \frac{\delta}{|\mathscr{Q}|}.
\end{aligned} \tag{3.7}$$

Given a bound on the probability of each component being inaccurate, we can now bound the probability that there exists a state-action pair whose transition distribution is inaccurate. In the following derivation (a) comes from Lemma 1, (b) is an application of the union bound over all components in $\mathscr{Q}$ and (c) comes from (Equation 3.7).

$$\begin{aligned}
\Pr&\left(\left\|P(\cdot \mid s, a) - \hat{P}(\cdot \mid s, a)\right\|_1 \geq \epsilon\right) \\
&\overset{(a)}{=} \Pr\left(\bigcup_{(X,a,\mathbf{x})\in\mathscr{Q}} \left\|P(\cdot \mid \mathbf{x}, a) - \hat{P}(\cdot \mid \mathbf{x}, a)\right\|_1 \geq \frac{\epsilon}{|\mathbb{X}|}\right) \\
&\overset{(b)}{\leq} \sum_{i=1}^{|\mathscr{Q}|} \Pr\left(\left\|P(\cdot \mid \mathbf{x}, a) - \hat{P}(\cdot \mid \mathbf{x}, a)\right\|_1 \geq \frac{\epsilon}{|\mathbb{X}|}\right) \\
&\overset{(c)}{\leq} \sum_{i=1}^{|\mathscr{Q}|} \frac{\delta}{|\mathscr{Q}|} \\
&= \delta,
\end{aligned} \tag{3.8}$$

which concludes our proof. □

Now, to ensure that the conditions for Proposition 1 hold, we can set the minimum number of observations for each component to $m_i = \frac{2|\mathbb{X}|^2}{\epsilon^2}\log\frac{|\mathscr{Q}|2^{|\mathrm{dom}(X)|}}{\delta}$, which is derived from Equation 3.5 by isolating $\eta(\mathbf{x}, a)$. This guarantees that all components relevant for non-bootstrapped state-action pairs were experienced enough times, such that the error of the transition function of these state-action pairs is smaller than ϵ. This way, we can use Proposition 1 to replace Proposition 3 in the proof of Theorem 3 (Laroche et al., 2019).

Theorem 2. *(Safe Policy Improvement of the Factored Π_b-SPIBB Algorithm). Let Π_b be the set of policies under the constraint of following π_b in every bootstrapped state-action pair $(s,a) \in \mathbb{B}_{\vec{m}}$. Then, given the dependency function Pa of the underlying FMDP, the policy $\pi^{\odot}$ computed by the Factored Π_b-SPIBB algorithm, is at least a ζ-approximate safe policy improvement over π_b with high probability $1-\delta$, with*

$$\zeta = \frac{4\epsilon V_{\max}}{(1-\gamma)} - V(\pi^{\odot}, \widehat{\mathscr{M}}) + V(\pi_b, \widehat{\mathscr{M}}).$$

Proof. The proof is similar to that of Theorem 3 (Laroche et al., 2019). □

These results show that it is possible to bound the probability that the Factored Π_b-SPIBB algorithm computes a policy worse than the behavior policy. The main difference with the original Π_b-SPIBB algorithm is the way we bound the error of the transition function. Given a desired ϵ, the term $|\mathbb{A}||\mathbb{S}|$ is replaced by $|\mathscr{Q}|$ and $|\mathbb{S}|$ is now reduced to $|\mathrm{dom}(X)|$. This comes at the lower cost of adding a term polynomial in the number of variables $|\mathbb{X}|^2$ (necessary to bound the error of each component distribution). In domains where the features are highly independent of each other, these results can significantly reduce the number of samples necessary to improve the behavior policy, as demonstrated in the empirical analysis.

3.3.4. Empirical Analysis

We evaluate the proposed factored approaches for the SPI problem focusing on their sample efficiency and generalization capability. All algorithms use a flat representation to estimate the transition function, as in the Π_b-SPIBB algorithm, and flat Value Iteration with a discount factor of 0.99 to compute the new policy. We assume that the reward function is known in all algorithms.

We evaluate the Π_b-SPIBB and Factored Π_b-SPIBB algorithms and their respective relaxations $\Pi_{\leq b}$-SPIBB and Factored $\Pi_{\leq b}$-SPIBB. We compare these results with two basic model-based RL algorithms that simply estimate the underlying model and compute a greedy policy according to this estimate. The first, called Basic Flat RL, uses a flat representation and the second, called Basic Factored RL, uses a factored representation.

Experimental Setup

We use two domains with known independence between features:

i. the Taxi domain (Dietterich, 1998) that has 4 conditionally independent features, 500 states, 6 actions, and a horizon of 200 steps; and

ii. the SysAdmin domain with 8 machines in a bidirectional ring topology (Guestrin et al., 2003) that has 256 states, 9 actions, and a 40 steps horizon.

Our analysis uses three metrics:

i. the performance of the policies computed;

ii. the size of the set of bootstrapped state-action pairs; and

iii. the distribution error of $\hat{P}$, defined as the average of the L_1 distance between the estimated transition function and the true transition function.

The first two experiments have a setup similar to the empirical evaluation of the SPIBB methodology (Laroche et al., 2019). A baseline policy π_b is computed using softmax exploration over the optimal value function of each state-action pair (temperature 2 for the Taxi domain and 3 for the SysAdmin domain). Next, a batch of past experiences $\mathscr{D}$ is generated following the behavior policy π_b. Note that $\mathscr{D}$ is composed of a set of trajectories, therefore $|\mathscr{D}|$ denotes the number of trajectories in $\mathscr{D}$. Then, each algorithm uses the historical data $\mathscr{D}$ and the behavior policy π_b to compute a new policy π'. Finally, the policies computed are evaluated by averaging the returns of 1000 simulations.

The third experiment uses the same instance of the SysAdmin domain but with a deterministic behavior policy that always executes the action with the second-highest expected value. This way, there is space for improvement in every state. Note that because the policy is deterministic, this experiment requires the agent to be able to generalize its past experiences to new states.

We performed a parameter search in each domain to choose the minimum value for m and m_i that maintains the safety of the algorithm. For the Taxi domain we set $m = 10$ and $m_i = 20$ for $0 < i < |\mathbb{X}|$. In the case of the SysAdmin problem we set $m = 50$ and $m_i = 10$ for $0 < i < |\mathbb{X}|$.

Experimental Results

Figure 3.1 shows the results obtained. Each column presents a different experiment and each row presents a different metric. The first row shows the average performance of the policies computed over 1000 repetitions and the 1st percentile of these values, which lets us assess the safety of each method. On the second and third rows we omit the results of Basic Flat RL, Basic Factored RL, $\Pi_{\leq b}$-SPIBB, Factored $\Pi_{\leq b}$-SPIBB that are equal to Π_b-SPIBB and Factored Π_b-SPIBB respectively or do not apply. Note that solid (dotted) lines are used for algorithms that use a flat (factored) representation, and dashed lines are used for the baseline policy. We highlight that the experiments with different batch sizes are independent of each other, that is, the trajectories collected where $|\mathscr{D}| = x$ are not related to the trajectories collected where $|\mathscr{D}| < x$.

Experiments with a stochastic policy

In the Taxi experiment (Figure 3.1, first column), we notice that although the unsafe algorithms (Basic Flat RL and Basic Factored RL) can improve their performance quickly, they obtain policies worse than the behavior policy when the batch contains only a few trajectories, precisely what safe reinforcement learning tries to avoid. All the other algorithms are shown to be safe as expected.

The main result of this work is the difference in the number of samples necessary to change the behavior policy of flat SPI algorithms and their factored counterparts. The

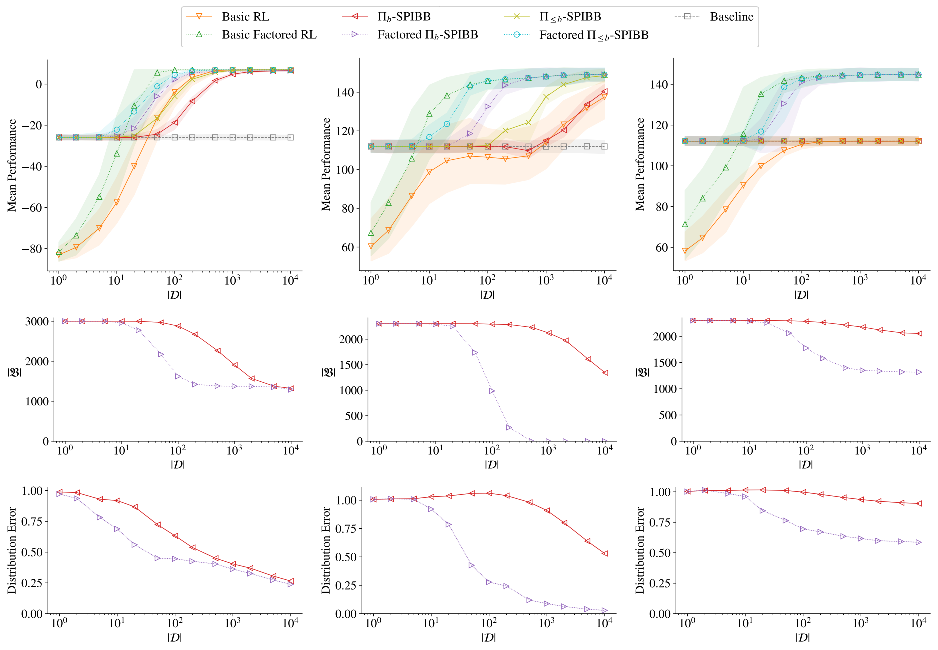


Figure 3.1: In every plot the x-axis shows the number of trajectories in the batch. Each column shows the results of a different experiment: Taxi with softmax policy (left), SysAdmin with softmax policy (middle) and SysAdmin with deterministic policy (right). The rows present: i) the average performance of the computed policy (top), ii) the number of bootstrapped state-action pairs (middle), and iii) the average distribution error of the estimated transition function (bottom).

Factored Π_b-SPIBB algorithm manages to compute policies better than the behavior policy given batches with only 20 trajectories, in contrast to the Π_b-SPIBB algorithm, that only shows improvement when $|\mathcal{D}| \geq 50$. As already demonstrated, the $\Pi_{\leq b}$-SPIBB algorithm can be less conservative (Laroche et al., 2019), and is able to find better improvements when $|\mathcal{D}| = 50$, while its factored version (Factored $\Pi_{\leq b}$-SPIBB) is even less conservative finding improvements when $|\mathcal{D}| = 5$ and already getting close to the optimal policy when $|\mathcal{D}| = 50$. To provide a more precise measure of these differences, Table 3.1 shows the results when $|\mathcal{D}| = 50$.

When comparing the number of bootstrapped state-action pairs by each algorithm (Figure 3.1, second row) and the performance of the policy computed, we see a strong correlation between them. Namely, a smaller number of bootstrapped state-action pairs results in higher performance. The quicker reduction of the distribution error (Figure 3.1, third row) shows why the Factored SPIBB algorithm can bootstrap from fewer state-action pairs; this is a clear result of the generalization capacity of factored representations.

In the SysAdmin with softmax policy experiment (Figure 3.1, second column) the results are similar, although the differences between (Factored) Π_b-SPIBB and (Factored) $\Pi_{\leq b}$-SPIBB algorithms are much larger than in the first experiment. We also notice that for the factored algorithms $|\mathbb{B}|$ drops quickly between $|\mathcal{D}| = 20$ and $|\mathcal{D}| = 200$ when these

Algorithm	**Mean**	**1st percentile**
Baseline Value	-25.91	-27.09
Basic RL	-16.43	-27.24
Basic Factored RL	5.56	-4.71
Π_b-SPIBB	-24.36	-26.54
Factored Π_b-SPIBB	-5.94	-9.72
$\Pi_{\leq b}$-SPIBB	-16.60	-19.70
Factored $\Pi_{\leq b}$-SPIBB	-1.08	-4.32

Table 3.1: Performance of policies computed when $|\mathscr{D}| = 50$.

algorithms stop bootstrapping and achieve the same performance as Basic Factored RL.

EXPERIMENTS WITH A DETERMINISTIC POLICY.
Finally, in the SysAdmin with deterministic behavior policy experiment (Figure 3.1, third column), the factored algorithms are the only ones that manage to improve the behavior policy. As expected, none of the flat algorithms can find a policy better than the behavior policy, given that the behavior policy is deterministic. We notice that the distribution error of the flat representation only drops slightly while for the factored representation it drops to an average of 0.5, which is enough to let the factored algorithms stop bootstrapping some of the state-action pairs.

3.4. STRUCTURE LEARNING FOR SAFE POLICY IMPROVEMENT

In the previous section, we showed that the Π_b-SPIBB framework can be extended to factored environments. Algorithm 4 shows an overview of the Factored Π_b-SPIBB algorithm. Note that this algorithm takes as input the structure of the DBN, represented by the set of parents of each variable-action pair. The main enhancement of this algorithm is a reduction in the number of samples required to stop bootstrapping a state-action pair.

The Factored Π_b-SPIBB algorithm assumes that the structure of the FMDP is known a priori, a strong assumption that frequently is not satisfied. In this section, we propose a more general version of the Factored SPIBB framework for problems where the structure of the problem is unknown.

In this section, we consider a setting where the maximum in-degree d of the underlying FMDP is known. We use the KWIK algorithms designed to learn the structure of the problem as described in Section 2.4.1. We start with an overview of how to couple structure learning algorithms with the SPIBB framework, followed by theoretical and empirical analyses of the new framework.

3.4.1. THE ALGORITHM

Structure Learning Π_b-SPIBB keeps track of the distribution of each transition component using a separate subalgorithm $\mathscr{K}_{X_i,a}$, for each state-variable and action pair $(X_i, a) \in \mathbb{X} \times \mathbb{A}$. The subalgorithms used by this framework can be borrowed from the

Algorithm 5 Structure Learning Π_b-SPIBB

```
Input: Previous experiences $\mathcal{D}$
Input: Parameters $\epsilon, \delta$
Input: Behavior policy $\pi_b$
Input: Subalgorithms $\mathcal{K}$
Output: Safe Policy
 1: for all $(X_i, a) \in \mathbb{X} \times \mathbb{A}$ do
 2:     Initialize $\mathcal{K}_{X_i,a}$ with $\frac{\epsilon}{|\mathbb{X}|}$ and $\frac{\delta}{|\mathbb{X}||\mathbb{A}|}$
 3:     Present $\{(s, a', s') \in \mathcal{D} \mid a' = a\}$ to $\mathcal{K}_{X_i,a}$
 4: end for
 5: $\mathbb{B} = \emptyset$
 6: for all $(s, a) \in \mathbb{S} \times \mathbb{A}$ do
 7:     if $\exists X_i \in \mathbb{X} : \mathcal{K}_{X_i,a}(s) = \oslash$ then
 8:         $\mathbb{B} = \mathbb{B} \cup \{(s, a)\}$
 9:         $\hat{P}(s' \mid s, a) = 0, \forall s' \in \mathbb{S}$
10:     else
11:         $P_i = \mathcal{K}_{X_i,a}(s), \forall X_i \in \mathbb{X}$
12:         $\hat{P}(s' \mid s, a) = \prod_{X_i \in \mathbb{X}} P_i(s'[X_i]), \forall s' \in \mathbb{S}$      ▷ Equation 2.1
13:     end if
14: end for
15: Compute $\Pi_b$ according to $\mathbb{B}$      ▷ Equation 3.2
16: return $\arg\max_{\pi \in \Pi_b} V(\pi, \widehat{\mathcal{M}})$
```

factored RL literature (Section 2.4.1). Algorithm 5 presents an overview of the new framework. Different from the original SPIBB algorithm (Algorithm 3), this framework takes as input a class of subalgorithms $\mathcal{K}$ that must be chosen according to the prior knowledge available. Note that if the given subalgorithm knows the underlying structure, this algorithm would be equal to the Factored Π_b-SPIBB algorithm.

First, Algorithm 5 instantiates one subalgorithm for each variable-action pair and presents the relevant transitions from the batch of previous experiences $\mathcal{D}$ to it (lines 1-4). Next, it estimates the transition function and, at the same time, it builds the set of bootstrapped state-action pairs (lines 5-14). Line 8, shows how the set of bootstrapped state-action pairs $\mathbb{B}$ is constructed. For a given state-action pair (s, a), if one of the subalgorithms related to a returns $\perp$ given s (line 7), then the pair (s, a) is added to $\mathbb{B}$. Note that for these pairs, at least one of the subalgorithms does not return a distribution, therefore we assume that these state-action pairs are absorbing. Finally, the safe policy is computed in the same way as by the Factored Π_b-SPIBB algorithm.

As discussed in Section 2.4, factored RL has been studied in settings where the dependence structure is unknown. These approaches include methods that search for the underlying structure with and without guarantees of sample complexity (Chakraborty and Stone, 2011; Degris et al., 2006; Diuk et al., 2009; Strehl et al., 2007). Although some of these methods do not provide correctness guarantees, we believe they could also be coupled with the factored SPIBB framework to develop more practical algorithms. Another line of research considers a case where a prior distribution over the underlying structures is available. Ross and Pineau (2008) propose a Bayesian RL approach (Duff, 2002; Vlassis et al., 2012) that exploits the DBN structure by adding the belief over the underlying structures to the state space instead of the parameters of the transition function.

3.4.2. THEORETICAL ANALYSIS

For the theoretical analysis of the proposed algorithm, we first show that the transition function of non-bootstrapped state-action pairs is precise with high probability.

Proposition 2. *When the Structure Learning Π_b-SPIBB algorithm is equipped with a subalgorithm $\mathcal{K}$ that is KWIK-admissible, all non-bootstrapped state-action pairs have a transition error smaller than ϵ with high probability $1-\delta$:*

$$\Pr(\forall (s,a) \notin \mathbb{B} : \left\| P(\cdot \mid s,a) - \hat{P}(\cdot \mid s,a) \right\|_1 \leq \epsilon) \geq 1-\delta.$$

Proof. According to Strehl (2007, Corollary 1), if all $|\mathbb{X}|$ relevant subalgorithms return a distribution with error smaller than $\frac{\epsilon}{|\mathbb{X}|}$ then the error of the estimated transition function is smaller than ϵ. Using a union bound, we conclude that the probability that there is a factor with error larger than $\frac{\epsilon}{|\mathbb{X}|}$ is smaller than $\sum_1^{|\mathbb{X}||\mathbb{A}|} \frac{\delta}{|\mathbb{X}||\mathbb{A}|} = \delta$. Given that the subalgorithm used is KWIK-admissible, the above assumptions hold, which concludes our proof[2]. □

We can use Proposition 2 to show that the proposed algorithm is safe.

Theorem 3. *(Safe Policy Improvement of the Structure Learning Π_b-SPIBB Algorithm). Let Π_b be the set of policies under the constraint of following π_b in every bootstrapped state-action pair $(s,a) \in \mathbb{B}$. Then, given the maximum in-degree d of the underlying FMDP, the policy $\pi^{\odot}$ computed by the Structure Learning Π_b-SPIBB algorithm, is at least a ζ-approximate safe policy improvement over π_b with high probability $1-\delta$, with*

$$\zeta = \frac{4\epsilon V_{\max}}{(1-\gamma)} - V(\pi^{\odot}, \widehat{\mathcal{M}}) + V(\pi_b, \widehat{\mathcal{M}}).$$

Proof. We use Proposition 2 to replace Proposition 1 in the proof of Theorem 2 by Laroche et al. (2019). □

In conclusion, the algorithm Structure Learning Π_b-SPIBB is safe when equipped with a KWIK-admissible algorithm.

3.4.3. EMPIRICAL ANALYSIS

We evaluate the Structure Learning Π_b-SPIBB framework combined with the two structure learning algorithms presented before (SL and k-meteorologists, see Section 2.4.1 for details) in three domains. The two baselines for comparison are the Factored Π_b-SPIBB algorithm (Simão and Spaan, 2019a, Section 3.3.1), that knows the structure of the problem, and the (flat) Π_b-SPIBB algorithm (Laroche et al., 2019, Section 3.2.3), that does not

[2]This proof follows the same principle used by (Li et al., 2011, p. 413, Problem 9) to show that the output combination algorithm is KWIK-admissible. However, to prove that FMDPs with unknown structure are KWIK-learnable, Li et al. (2011) use a different algorithm that includes the action as one of the input variables.

		Taxi	SysAdmin	Stock-Trading
Π_b-SPIBB	m	10.00	100.00	10.00
Factored Π_b-SPIBB	m	20.00	10.00	20.00
Π_b-SPIBB SL	m	20.00	10.00	20.00
	ϵ_1	0.01	0.20	0.30
Π_b-SPIBB	m	10.00	10.00	20.00
k-meteorologists	ϵ_1	0.01	0.00	0.01
	c	2000.00	2000.00	300.00

Table 3.2: Parameters used by each algorithm

consider this structure. We also consider an algorithm without safety guarantees, referred to as Factored Basic RL. This algorithm has access to the problem's structure and computes a greedy policy according to an estimate of the factored transition function of the problem.

All algorithms use a flat estimate of the transition function and a flat Value Iteration algorithm with a discount factor of 0.99. We assume that the reward function is known in all algorithms. The problems used are:

i. the Taxi domain with a horizon of 200 steps (Dietterich, 1998),

ii. the SysAdmin domain with 9 machines in a bidirectional ring topology and a horizon of 40 steps (Guestrin et al., 2003), and

iii. the Stock-Trading domain with 3 sectors and 2 stocks per sector with a horizon of 40 steps (Strehl et al., 2007).

We follow a setup similar to the experiments by Laroche et al. (2019) and the experiments from Section 3.3.4, where the behavior policy π_b is defined as a softmax over the optimal value function. The softmax temperature is set to 2 for the Taxi and Stock-Trading domains and 3 for the SysAdmin domain.

In summary, each experiment is executed in three steps, varying the number of episodes in the batch $\mathscr{D}$ of previous experiences:

i. create $\mathscr{D}$ by executing the behavior policy π_b,

ii. present $\mathscr{D}$ and π_b (in the case of the safe algorithms) to each algorithm and compute a new policy π', and

iii. estimate the performance of each new policy by averaging the discounted returns of 1000 trials.

Table 3.2 reports the parameters used by each algorithm. These values were chosen accordingly to reduce the number of samples required to improve the policy while keeping a safe behavior. Figures 3.2 and 3.3 present the results. The x-axis of each plot shows the number of episodes in the batch collected with the behavior policy.

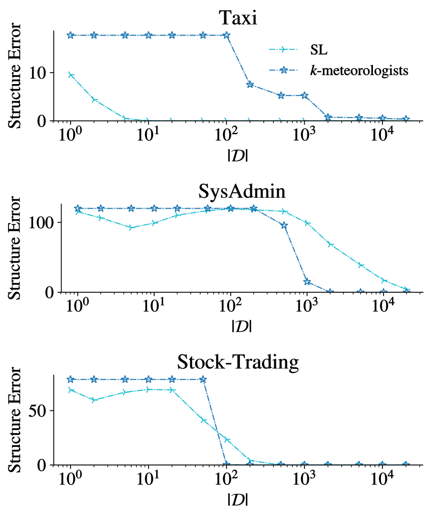


Figure 3.2: Structure learning error of the SL and k-meteorologists algorithms

Searching for the Best Candidate Structure

Figure 3.2 shows how the estimated structure improves as the structure learning algorithms receive more data. For the SL algorithm it shows the number of parents missing in the selected candidate:

$$\sum_{X_i,a\in\mathbb{X}\times\mathbb{A}} \left|\mathrm{Pa}_a(X_i)\setminus\Delta_{X_i,a}\right|,$$

where $\Delta_{X_i,a}$ is the candidate chosen by subalgorithm $\mathscr{K}_{X_i,a}$. For the k-meteorologists algorithm it shows the average number of missing parents between all remaining candidates:

$$\sum_{X_i,a\in\mathbb{X}\times\mathbb{A}} \frac{\sum_{\Delta\in\psi_{X_i,a}}|\mathrm{Pa}_a(X_i)\setminus\Delta|}{\left|\psi_{X_i,a}\right|},$$

where $\psi_{X_i,a}\subseteq\mathbb{X}$ is the set of remaining candidates of $\mathscr{K}_{X_i,a}$.

In the Taxi domain (Figure 3.2 top), the overall number of missing parents is small, which is expected since this domain has only four variables. Nevertheless, we note that the k-meteorologists algorithm needs significantly more samples to discard the candidates that do not contain the true parents. That occurs because the number of mismatches (c) must be large to avoid erroneously discarding a candidate.

The structure learning algorithms exhibit a similar behavior in the SysAdmin domain (Figure 3.2 middle) and in the Stock-Trading domain (Figure 3.2 bottom). Both algorithms select candidates missing many parents in small datasets. The k-meteorologists algorithm, however, has a sudden drop in the number of missing parents. This is explained by the fact that all the subalgorithms crossed the minimum number of mismatches when the dataset is larger than a certain threshold.

We also note that the SL algorithm does not display a monotonic improvement in these domains. This is because to compute the distribution divergence of two sets of

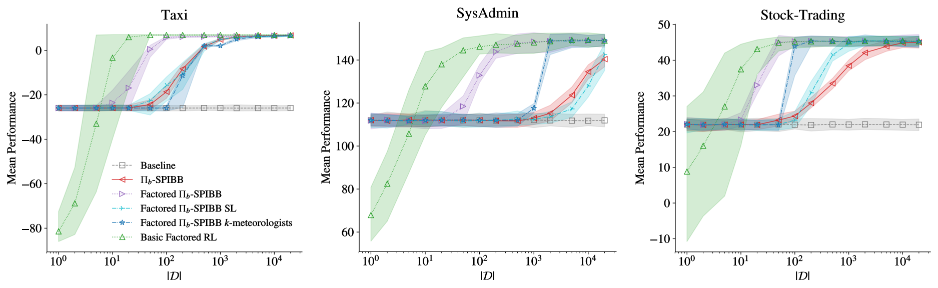


Figure 3.3: Average performance of the computed policy over 50 repetitions, along with the 1st percentile and 99th percentile (shaded area)

candidates (Equation 2.11), we only consider configurations that have been observed at least once. Without this measure, the SL algorithm is not able to learn the structure of the Taxi domain, where some configurations are never observed. Therefore, in the SysAdmin and Stock-Trading domains, when $|\mathscr{D}|$ is small, the SL algorithm ignores some of the configurations and can improve the estimated structure. However, when $\mathscr{D}$ contains more configurations, it becomes more conservative again, returning $\oslash$ until it gets enough data for all configurations.

Policy Improvement

Next, we evaluate the performance of the Factored Π_b-SPIBB with structure learning algorithms (Figure 3.3). We present the average performance of 50 repetitions, plus the 1st percentile and 99th percentile (shaded area) to measure the risk of the algorithms.

First, we observe that in the Taxi domain (Figure 3.3 left), the structure learning approaches are slightly more conservative than the flat Π_b-SPIBB algorithm. This is not surprising since, as pointed out by Strehl et al. (2007), a DBN is not the ideal structure to capture the independence between the variables of this domain. Comparing the methods using structure learning algorithms, we see that the Factored Π_b-SPIBB SL algorithm requires fewer samples to exceed the performance of the behavior policy than the Factored Π_b-SPIBB k-meteorologists algorithm. This is because, as mentioned before, the k-meteorologists algorithm requires a large number of mismatches to discard a candidate, while the SL algorithm can choose the best candidate with fewer samples.

The advantages of using a structure learning algorithm are more prominent in well-factorized environments, where the maximum in-degree d is much smaller than the number of state variables. The experiments with the SysAdmin and Stock-Trading domains illustrate this fact (Figure 3.3 middle and right). We note that in the Stock-Trading domain, the Factored Π_b-SPIBB algorithm needs around 50 trajectories to find the optimal policy while the Factored Π_b-SPIBB k-meteorologists algorithm needs 200 trajectories and the flat Π_b-SPIBB algorithm needs 10000 trajectories. Their relative performance is similar in the SysAdmin domain. In summary, using the k-meteorologists algorithm, Structure Learning Π_b-SPIBB demonstrates a behavior closer to the Factored Π_b-SPIBB algorithm and manages to find an improved policy with an order of magnitude fewer samples than the flat Π_b-SPIBB algorithm.

3.5. A Realistic Case Study

The algorithms proposed in this chapter, along with their flat counterparts, have been tested in a steel melting plant simulation (Kosiorek, 2020). This application is appealing since the process of melting steel consumes large amounts of energy, and even small optimizations can significantly reduce the total energy consumption (Castro et al., 2013).

The steel melting process is executed over a set of batches of metal scrap, called *heats*. Each heat is stored in a ladle that goes through a set of treatment stations, including melting, decarburization, refining, and casting (Castro et al., 2013). Heats with similar characteristics are grouped in a campaign, which must be cast together in a sequence. This imposes time constraints on the problem, which are difficult to handle due to the stochastic duration of each process. In particular, finding the right time to launch each ladle can make the process more efficient, reducing the waiting time of the machines and of the ladles spread over the plant.

In the Steel Plant Model (SPM; Schrama et al., 2015), the launching of each ladle is decided heuristically based on the time passed since the last launch, which might be sub-optimal. Kosiorek (2020) proposed to model this problem as an MDP to compute a ladle launching policy considering the current position of the ladles in the plant. Using the built-in heuristic as the behavior policy for data collection allowed the evaluation of SPI algorithms, such as the proposed in this chapter. In this setting, the SPIBB and factored SPIBB algorithms steadily improved over the behavior policy, finding policies with better production yield and less idle time.

This case study revealed an issue that is not taken into account by the SPI algorithms; the behavior policy might be only partially defined. Meaning it assigns actions only to a subset of the state space, namely, states reachable by following the behavior policy. In practice, the behavior policy may be partially defined when hand-designed since the designer would only put effort into relevant states. This can be an issue for SPI since minor deviations from the behavior policy could lead the agent to states uncovered by the behavior policy, where the expected behavior is simply unknown.

Kosiorek (2020) also handled discrepancies between the real world and the usual RL setting. The SPI task imposes extra constraints on the MDP modeling, which must be compatible with the behavior policy. For instance, the heuristic used on the SPM depends on the time since the last launch, which, when included in the state space, can cause a considerable growth in the state space. While the policy computed by the RL algorithm may not need this information, the SPIBB requires the compatibility between the domain of the behavior policy and the agent's policy.

These findings indicate that practitioners should model their problems as an MDP as soon as possible. This approach would allow the launching heuristic from SPM to be defined on the state space of the MDP, allowing an easier deployment of SPI algorithms later on. Finally, the data collected by the behavior policy might have limited coverage of the problem. Since offline RL algorithms are most effective when the data collected is more diverse, practitioners intending to use such algorithms should consider behavior policies with more extensive support to collect diverse trajectories.

3.6. Conclusions and Future Work

We proposed the Factored Π_b-SPIBB algorithm, an adaptation of the Π_b-SPIBB algorithm (Laroche et al., 2019) to environments with factored dynamics and proved that this algorithm is safe. The Factored Π_b-SPIBB algorithm inherits from factored RL the capability to exploit the independence between features, allowing it to reduce the number of samples necessary to stop bootstrapping the behavior policy. This new method also can improve deterministic policies by generalizing past experiences, a novel feature for SPI algorithms.

Furthermore, we presented an SPI framework for factored environments with unknown structure. Relaxing the assumption that the underlying structure is known a priori makes this method applicable in a broader range of problems. When equipped with an efficient structure learning method, this framework can still exploit the factored structure of the environment and typically requires fewer samples than a flat algorithm to improve the behavior policy.

For simplicity, we assumed that the reward function is known. However, this function can also be succinctly represented in cases with an additive property, as described in Section 2.2. Therefore, it would also be possible to adapt the Factored Π_b-SPIBB algorithm to environments with an unknown reward function.

Studying how to exploit other types of structure such as decision trees and linear dynamics in this setting is a promising line of future work. Finally, we believe it is also possible to extend other model-based SPI algorithms to factored environments, such as the robust approach by Petrik et al. (2016).

4 Safe Policy Improvement with an Estimated Behavior Policy

Previous work has shown the unreliability of existing algorithms in the batch Reinforcement Learning setting, and proposed the theoretically-grounded Safe Policy Improvement with Baseline Bootstrapping (SPIBB) fix: reproduce the baseline policy in the uncertain state-action pairs, in order to control the variance on the trained policy performance. However, in many real world applications such as dialogue systems, pharmaceutical tests or crop management, data is collected under human supervision and the baseline remains unknown. In this chapter, we apply SPIBB algorithms with a baseline estimate built from the data. We formally show safe policy improvement guarantees over the true baseline even without direct access to it. Our empirical experiments on finite and continuous states tasks support the theoretical findings. It shows little loss of performance in comparison with SPIBB when the baseline policy is given, and more importantly, drastically and significantly outperforms competing algorithms both in safe policy improvement, and in average performance.

This chapter is based on a paper presented at AAMAS-20 (Simão et al., 2020).

In Chapter 3, we saw recent advances on offline RL, where algorithms such as SPIBB and Soft-SPIBB have improvement guarantees. Moreover, we saw that exploiting the structure of FMDPs improves the sample complexity of SPI algorithms.

Despite such appealing results, there is a caveat: these algorithms require the behavior policy as input. However, the behavior policy is not always available. Consider, for instance, applications involving human interactions, such as dialogue systems (Serban et al., 2016) and the medical sector. In these situations, it is common to have access to the observations and actions that were taken in a trajectory but not the policy that was followed. To overcome this issue, we investigate the use of SPIBB and Soft-SPIBB algorithms in the setting where the behavior policy is unknown.

In this chapter, we address Research Question 4, which asks whether access to the behavior policy is required or not to obtain improvement guarantees, a very natural question arising from the existing SPIBB analysis. Therefore, the contributions are threefold:

i. We formally prove safety bounds for SPIBB and Soft-SPIBB algorithms with estimated baseline policies in finite MDPs (Section 4.2).

ii. We consolidate the theoretical results with empirical results in finite randomly generated MDPs, unknown baselines, and datasets (Section 4.4.1).

iii. We apply the method on a continuous state task by investigating two types of behavior cloning and show that it outperforms competing algorithms by a large margin, in particular on small datasets (Section 4.4.2).

The code to reproduce all experiments is available online[1]. In summary, our results bring the SPIBB framework a step closer to many real world tasks where the behavior policy is unknown.

4.1. Approximate Safe Policy Improvement

In this section, we review the safe policy improvement problem for the reader's convenience and describe the Soft-SPIBB. For a more comprehensive introduction to the safe policy improvement problem, we refer to Section 3.2.

In the SPI setting, the agent receives as input the dataset $\mathscr{D}$ and must compute a new policy π that approximately improves with high probability the behavior policy π_b. Formally, the safety criterion can be defined as:

$$\Pr\left(V(\pi,\mathscr{M}^*) \geq V(\pi_b,\mathscr{M}^*) - \zeta\right) \geq 1-\delta,$$

where ζ is a hyper-parameter indicating the improvement approximation and $1-\delta$ is the high confidence hyper-parameter. Petrik et al. (2016) demonstrate that the optimization of this objective is NP-hard. To make the problem tractable, Laroche et al. (2019) end up considering an approximate solution by maximizing the policy in the MLE MDP while constraining the policy to be approximately improving in the robust MDPs set Σ. More formally, they seek:

$$\arg\max_{\pi} V(\pi,\widehat{\mathscr{M}}), \text{ s.t. } \forall \mathscr{M} \in \Sigma, V(\pi,\mathscr{M}) \geq V(\pi_b,\mathscr{M}) - \zeta.$$

[1] `https://github.com/RomainLaroche/SPIBB` and `https://github.com/rems75/SPIBB-DQN`

Given a hyper-parameter m, their algorithm Π_b-SPIBB constrains the policy search to the set Π_b of policies that reproduce the baseline probabilities in the state-action pairs that are present less than m times in the dataset $\mathcal{D}$ (Equation 3.2). We restate the definition here for convenience:

$$\Pi_b = \{\pi \in \Pi \mid \pi(a \mid s) = \pi_b(a \mid s) : \forall (s,a) \in \mathbb{B}_m\}.$$

Our work also considers the algorithm Soft-SPIBB (Nadjahi et al., 2019), that constrains the policy search such that the cumulative state-local error never exceeds a fixed hyper-parameter σ. More formally, the policy constraint is expressed as follows:

$$\Pi_\sim = \left\{ \pi \in \Pi \;\middle|\; \sum_{a \in \mathbb{A}} e_\delta(s,a)|\pi(a \mid s) - \pi_b(a \mid s)| \leq \sigma : \forall s \in \mathbb{S} \right\}. \tag{4.1}$$

Under some assumptions, Nadjahi et al. (2019) demonstrate a looser safe policy improvement bound. Nevertheless, the policy search is less constrained, and their empirical evaluation reveals that Soft-SPIBB safely finds better policies than SPIBB.

Both algorithms presented in this section assume the behavior policy π_b is known and can be used during the computation of a new policy. In the next section, we get to the main contribution of this chapter, where we investigate how these algorithms can be applied when π_b is not given.

4.2. BASELINE ESTIMATES

In this section, we consider that the true baseline is unknown and implement a baseline estimate in order for the SPIBB and Soft-SPIBB algorithms to still be applicable. Before we start our analysis, we present an auxiliary lemma.

Let $d^\pi_{\mathcal{M}}(a \mid s)$ be the discounted sum of visits of state-action pair $(s,a) \in \mathbb{S} \times \mathbb{A}$ while following policy π in MDP $\mathcal{M}$ $d_{\mathcal{D}}$ the state-action discounted distribution in the dataset $\mathcal{D}$:

$$d_{\mathcal{D}}(a \mid s) = \sum_{\langle s_t, a_t, r_t, s'_t \rangle \in \mathcal{D}} \gamma^t \mathbb{1}(s_t = s)\mathbb{1}(a_t = a),$$

where $\mathbb{1}$ is the indicator function.

Lemma 2. *Considering that the trajectories in $\mathcal{D}$ are i.i.d. sampled, the L_1 deviation of the empirical discounted sum of visits of state-action pairs is bounded. We have the following concentration bound:*

$$\Pr\left(\left\| d^{\pi_b}_{\mathcal{M}^*} - d_{\mathcal{D}} \right\|_1 (1-\gamma) \geq \epsilon\right) \leq \left(2^{|\mathbb{S}||\mathbb{A}|} - 2\right) \exp\left(-\frac{N\epsilon^2}{2}\right), \tag{4.2}$$

where N is the number of trajectories in $\mathcal{D}$.

Proof. Let $\mathcal{T} = (\mathbb{S} \times \mathbb{A})^*$ denote the set of trajectories and $T = \{T_1, \ldots, T_N\}$ be a set of N $\mathcal{T}$-valued random variables. For a given set of state-action pairs $E \subset \mathbb{S} \times \mathbb{A}$, we define the function f_E on $\mathcal{T}$ as:

$$f_E(T) = f_E(T_1, \ldots, T_N) := (1-\gamma) \sum_{i=1}^{N} \sum_{t \geq 0} \gamma^t \mathbb{1}(T_i^t \in E),$$

where T_i^t is the state-action pair on trajectory i at time t. In particular, we have that

$$f_E(\mathscr{D}) = N(1-\gamma)d_{\mathscr{D}}(E) \text{ and} \tag{4.3}$$

$$\mathbb{E}\left[f_E(T)\right] = N(1-\gamma)d^{\pi_b}_{\mathscr{M}^*}(E), \tag{4.4}$$

where $d_{\mathscr{D}}(E)$ and $d^{\pi_b}_{\mathscr{M}^*}(E)$ denote the mass of set E under $d_{\mathscr{D}}$ and $d^{\pi_b}_{\mathscr{M}^*}$ respectively.

For two sets T and T' differing only on one trajectory, say the k-th, we have:

$$\left|f_E(T) - f_E(T')\right| = \left|(1-\gamma)\sum_{t\geq 0}\gamma^t\left(\mathbb{1}(T_k^t \in E) - \mathbb{1}({T'}_k^t \in E)\right)\right| \leq 1.$$

This allows us to apply the independent bounded difference inequality by McDiarmid (1998, Theorem 3.1), which gives us:

$$\Pr\left(f_E(T) - \mathbb{E}[f_E(T)] \geq \bar{\varepsilon}\right) \leq \exp\left(-2\frac{\bar{\varepsilon}^2}{N}\right). \tag{4.5}$$

We know that

$$\left\|d^{\pi_b}_{\mathscr{M}^*} - d_{\mathscr{D}}\right\|_1(1-\gamma) = \max_{E\subset\mathbb{S}\times\mathbb{A}} 2(1-\gamma)(d_{\mathscr{D}}(E) - d^{\pi_b}_{\mathscr{M}^*}(E)).$$

This guarantees from a coarse union bound and Equations 4.3 to 4.5 that:

$$\begin{aligned}
\Pr\left(\left\|d^{\pi_b}_{\mathscr{M}^*} - d_{\mathscr{D}}\right\|_1(1-\gamma) \geq \varepsilon\right)& \\
&\leq \sum_{E\subset\mathbb{S}\times\mathbb{A}} \Pr\left((1-\gamma)(d_{\mathscr{D}}(E) - d^{\pi_b}_{\mathscr{M}^*}(E)) \geq \frac{\varepsilon}{2}\right)\\
&= \sum_{E\subset\mathbb{S}\times\mathbb{A}} \Pr\left((1-\gamma)\left(\frac{f_E(\mathscr{D})}{N(1-\gamma)} - \frac{\mathbb{E}[f_E(\mathscr{D})]}{N(1-\gamma)}\right) \geq \frac{\varepsilon}{2}\right)\\
&= \sum_{E\subset\mathbb{S}\times\mathbb{A}} \Pr\left(f_E(\mathscr{D}) - \mathbb{E}[f_E(\mathscr{D})] \geq \frac{N\varepsilon}{2}\right)\\
&\leq \sum_{E\subset\mathbb{S}\times\mathbb{A}} \exp\left(-2\frac{(\frac{N\varepsilon}{2})^2}{N}\right)\\
&\leq \left(2^{|\mathbb{S}||\mathbb{A}|} - 2\right)\exp\left(-\frac{N\varepsilon^2}{2}\right),
\end{aligned}$$

where in the sum over subsets, we ignored the empty and full sets for which the probability is trivially 0. □

4.3. Algorithm and analysis

We construct the MLE baseline $\widehat{\pi_b}$ as follows:

$$\widehat{\pi_b}(a \mid s) = \begin{cases} \frac{\eta_{\mathscr{D}}(s,a)}{\eta_{\mathscr{D}}(s)} & \text{if } \eta_{\mathscr{D}}(s) > 0,\\ \frac{1}{|\mathbb{A}|} & \text{otherwise,}\end{cases} \qquad \forall s, a \in \mathbb{S}\times\mathbb{A}, \tag{4.6}$$

where $\eta_{\mathscr{D}}(s)$ is the number of transitions starting from state s in dataset $\mathscr{D}$. Using this MLE policy, we may prove approximate safe policy improvement:

Theorem 4 (Safe policy improvement with a baseline estimate)**.** *Given an algorithm α relying on the baseline π_b to train a ζ-approximate safe policy improvement $\pi^\odot$ over π_b with high probability $1-\delta$. Then, α with an MLE baseline $\widehat{\pi_b}$ allows to train a $\widehat{\zeta}$-approximate safe policy improvement $\widehat{\pi}^\odot$ over π_b with high probability $1-\widehat{\delta}$:*

$$\widehat{\delta} = \delta + 2\delta',$$
$$\widehat{\zeta} = \zeta + \frac{2R^\top}{1-\gamma}\sqrt{\frac{3|\mathbb{S}||\mathbb{A}| + 4\log\frac{1}{\delta'}}{2N}},$$

where N is the number of trajectories in the dataset $\mathscr{D}$ and $1-\delta'$ controls the uncertainty stemming from the baseline estimation.

Proof. We are ultimately interested in the performance improvement of $\widehat{\pi}^\odot$ with respect to the true baseline π_b in the true environment $\mathscr{M}^*$. To do so, we decompose the difference into two parts:

$$V(\widehat{\pi}^\odot,\mathscr{M}^*) - V(\pi_b,\mathscr{M}^*) = \underbrace{V(\widehat{\pi}^\odot,\mathscr{M}^*) - V(\widehat{\pi_b},\mathscr{M}^*)}_{\alpha\text{-SPI guarantee}} + \underbrace{V(\widehat{\pi_b},\mathscr{M}^*) - V(\pi_b,\mathscr{M}^*)}_{\text{baseline estimate approximation}}.$$

Regarding the first term, note that, while $\widehat{\pi_b}$ is not the true baseline, it is the MLE baseline, meaning in particular that it was more likely to generate the dataset $\mathscr{D}$ than the true one. Hence, we may consider it as a potential behavior policy and apply the safe policy improvement guarantee provided by algorithm α to bound the difference.

Regarding the second term, we need to use the distributional formulation of the performance of any policy π:

$$V(\pi,\mathscr{M}) = \sum_{s\in\mathbb{S}}\sum_{a\in\mathbb{A}} d^\pi_{\mathscr{M}}(a \mid s)\,\mathbb{E}[R(s,a)].$$

Then, we may rewrite the second term in Equation 4.7 and upper bound it using Hölder's inequality as follows:

$$\begin{aligned}\sum_{s\in\mathbb{S}}\sum_{a\in\mathbb{A}} &\left(d^{\widehat{\pi_b}}_{\mathscr{M}^*}(s,a) - d^{\pi_b}_{\mathscr{M}^*}(s,a)\right)\mathbb{E}[R^*(s,a)] \\ &\leq \left\| d^{\widehat{\pi_b}}_{\mathscr{M}^*} - d^{\pi_b}_{\mathscr{M}^*}\right\|_1 R^\top.\end{aligned} \tag{4.7}$$

Next, we decompose the state-action discounted visits divergence as follows:

$$\left\| d^{\widehat{\pi_b}}_{\mathscr{M}^*} - d^{\pi_b}_{\mathscr{M}^*}\right\|_1 \leq \underbrace{\left\| d^{\pi_b}_{\mathscr{M}^*} - d_{\mathscr{D}}\right\|_1}_{\text{Lemma 2}} + \underbrace{\left\| d^{\widehat{\pi_b}}_{\mathscr{M}^*} - d_{\mathscr{D}}\right\|_1}_{\text{positive correlation}}. \tag{4.8}$$

For the first term, we can use the concentration inequality from Lemma 2[2]. With a little calculus and by setting the right value to ε, we obtain with high probability $1-\delta'$:

$$\left\| d^{\pi_b}_{\mathscr{M}^*} - d_{\mathscr{D}}\right\|_1 \leq \frac{1}{1-\gamma}\sqrt{\frac{3|\mathbb{S}||\mathbb{A}| + 4\log\frac{1}{\delta'}}{2N}}.$$

[2]We need to rescale the state-action discounted visits with $(1-\gamma)$ to make it sum to 1 since the original bound applies to probability distributions.

Regarding the second term of Equation 4.8, we may observe that there is a correlation between $\widehat{\pi_b}$ and $d_{\mathscr{D}}$ through $\mathscr{D}$, but it is a positive correlation, meaning that the divergence between the distributions is smaller than the one with an independently drawn dataset of the same size. As a consequence, we are also able to upper bound it by assuming independence and using the same development as for the first term. This finally gives us from Equation 4.8 and with high probability $1-2\delta'$:

$$\left\| d^{\widehat{\pi_b}}_{\mathscr{M}^*} - d^{\pi_b}_{\mathscr{M}^*} \right\|_1 \leq \frac{2}{1-\gamma}\sqrt{\frac{3|\mathbb{S}||\mathbb{A}|+4\log\frac{1}{\delta'}}{2N}}, \tag{4.9}$$

which allows us to conclude the proof using union bounds. □

4.3.1. Theorem 4 discussion

SPIBB and Soft-SPIBB safe policy improvement guarantees exhibit a trade-off (controlled with their respective hyper-parameters $\frac{1}{\sqrt{m}}$ and σ) between upper bounding the true policy improvement error (first term in Theorem 1) and allowing maximal policy improvement in the MLE MDP (next terms). When the hyper-parameters are set to 0, the true policy improvement error is null because, trivially, no policy improvement is allowed: the algorithm is forced to reproduce the baseline. When the hyper-parameters grow, larger improvements are permitted, but the error upper bound term also grows. When the hyper-parameters tend to $+\infty$, the algorithms are not constrained anymore and find the optimal policy in the MLE MDP. In that case, the error is no longer upper bounded, resulting in poor safety performance.

When using the MLE baseline instead of the true baseline, Theorem 4 introduces another error to the upper bound term accounting for the accurateness of the baseline estimate that cannot be reduced by hyper-parameter settings. That fact is entirely expected, as otherwise, we could consider an empty dataset, pretend it was generated with an optimal policy, and expect a safe policy improvement over it. Another interesting point is that the bound depends on the number of trajectories, not the number of state-action visits, nor the total number of samples. Indeed, even with a huge number of samples, if there were collected only from a few trajectories, the variance may still be high since future states visited on the trajectory depend on the previous transitions.

Regarding the MDP parameters dependency, the upper bound grows as the square root of the state set size, as for standard SPIBB, but also grows as the square root of the action set size contrarily to SPIBB that has a logarithmic dependency, which may cause issues in some RL problems. The direct horizon dependency is the same (linear). But one could argue that it is actually lower. The maximal value V_{max} in the SPIBB bounds can reach $\frac{R^\top}{1-\gamma}$, making the dependency in H quadratic, while the N in our denominator may be regarded as a hidden horizon (since $N \approx \frac{|\mathscr{D}|}{H}$), making the total dependency $\approx H^{3/2}$. In both cases, those are better than the Soft-SPIBB cubic dependency.

One may consider other baseline estimates than the MLE, using Bayesian priors, for instance, and infer new bounds. This should work as long as the baseline estimate remains a policy that could have generated the dataset.

4.4. Empirical Analysis

Our empirical analysis reproduces the most challenging experiments found in Laroche et al. (2019) and Nadjahi et al. (2019). We split it into two parts: the first considers random MDPs with finite state spaces, and the second considers MDPs with continuous state spaces.

4.4.1. Random finite MDPs

The objective of this experiment is to empirically analyze the consistency between the theoretical findings and the practice. The experiment is run on finite MDPs that are randomly generated, with randomly generated baseline policies from which trajectories are obtained.

Setup

The true environment is a randomly generated MDP with 50 states, 4 actions, and a transition connectivity of 4: a given state-action pair may transit to 4 different states at most. The reward function is 0 everywhere, except for transitions entering the goal state, in which case the trajectory terminates with a reward of 1. The goal state is the hardest state to reach from the initial one.

The baselines are also randomly generated with a predefined level of performance specified by a ratio χ between the performance of the optimal policy π^* and the performance of the uniform policy $\tilde{\pi}$: $V(\pi_b, \mathcal{M}) = \chi V(\pi^*, \mathcal{M}) + (1-\chi) V(\tilde{\pi}, \mathcal{M})$. For more details on the process, we refer the interested reader to Laroche et al. (2019, Section B.1.4). Our experiments consider two values for χ: 0.1 and 0.9. We also study the influence of the dataset size $|\mathscr{D}| \in [10, 20, 50, 100, 200, 500, 1000, 2000]$.

Competing Algorithms

Our plots display 9 curves:

i. π^*: the optimal policy,

ii. π_b: the true baseline,

iii. $\widehat{\pi_b}$: the MLE baseline,

iv. $\Pi_b/\widehat{\Pi}_b$-SPIBB: SPIBB with their respective baselines,

v. $\Pi_b/\widehat{\Pi}_b$-Soft: Soft-SPIBB with their respective baselines,

vi. RaMDP: Reward-adjusted MDP,

vii. and Basic RL: dynamic programming on the MLE MDP.

All the algorithms are compared using their optimal hyper-parameter according to previous work. Our hyper-parameter search with the MLE baselines did not show significant differences, and we opted to report results with the same hyper-parameter values. Soft-SPIBB algorithms are the ones coined as Approx. Soft SPIBB by Nadjahi et al. (2019).

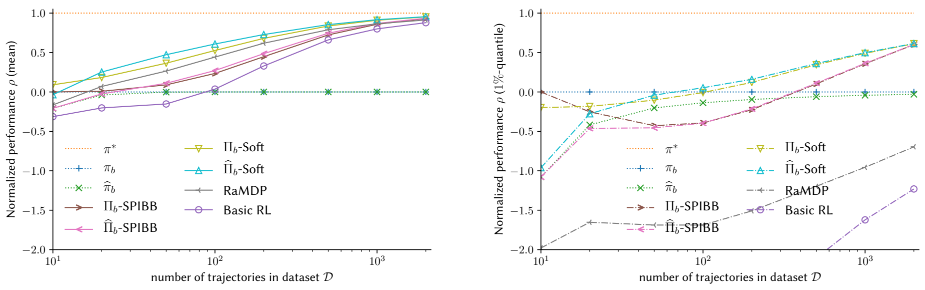


Figure 4.1: Finite MDPs with $\boldsymbol{\chi} = \mathbf{0.9}$, $\boldsymbol{m} = \mathbf{7}$ and $\boldsymbol{\sigma} = \mathbf{0.5}$. On the left, the mean curves, on the right, the 1st percentile curves.

Performance Indicators

Given the random nature of the MDP and baseline generations, we normalize the performance assuming $V(\pi_b, \mathcal{M}^*) < V(\pi^*, \mathcal{M}^*)$ to allow inter-experiment comparison:

$$\rho = \frac{V(\pi, \mathcal{M}^*) - V(\pi_b, \mathcal{M}^*)}{V(\pi^*, \mathcal{M}^*) - V(\pi_b, \mathcal{M}^*)}. \tag{4.10}$$

Thus, the optimal policy always has a normalized performance of 1, and the true baseline a normalized performance of 0. A positive normalized performance means a policy improvement, and a negative normalized performance means an infringement of the policy improvement objective. Figures either report the average normalized performance of the algorithms or its 1st percentile[3]. Each setting is processed on 250K seeds to ensure that every performance gap visible to the naked eye is significant.

Empirical Results

We start our analysis considering Figure 4.1, which shows the results for $\chi = 0.9$, the hard setting where the behavior baseline is almost optimal, and therefore difficult to improve.

Performance of the estimated policy. First, we notice that the mean performance of the MLE baseline $\widehat{\pi}_b$ is slightly lower than the true baseline policy π_b for small datasets. As $|\mathcal{D}|$ increases, the performance of $\widehat{\pi}_b$ quickly increases to reach the same level. The 1st percentile is significantly lower when the number of trajectories is reduced.

Soft-SPIBB with true and estimated baselines. Comparing the results of Π_b-Soft and $\widehat{\Pi}_b$-Soft curves, it is surprising that the policy computed using an estimated policy as a baseline yields better results than the one computed with the true policy. Notice that the estimated baseline $\widehat{\pi}_b$ has a higher variance than the true baseline π_b. If we consider the impact of this variance in a given state, it means that sometimes the best (resp. worst) action will be taken more often (resp. less). When it is the case, the trained policy will be better than what could have been done with the true baseline. Sometimes, the opposite

[3]Note the difference with previously reported results in SPIBB papers, which focused on the conditional value at risk indicator.

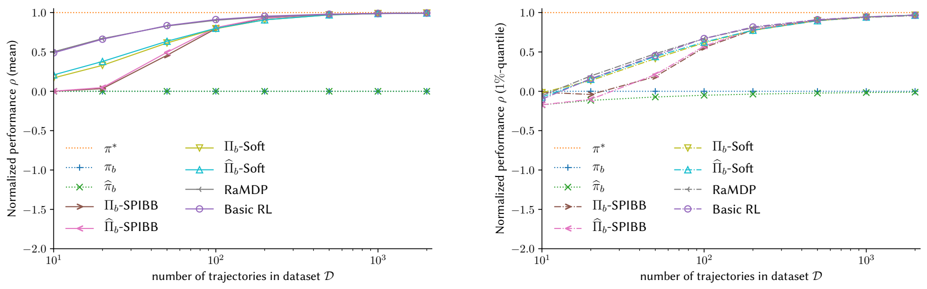


Figure 4.2: Finite MDPs with χ = **0.1**, m = **7** and σ = **0.5**. On the left, the mean curves, on the right, the 1st percentile curves.

will happen, but in this case, the algorithm will try to avoid reaching this state and choose an alternative path. This means that in expectation, this does not average out, and the variance in the baseline estimation might be beneficial.

SPIBB with true and estimated baselines. Analyzing the performance of the $\widehat{\Pi}_b$-SPIBB algorithm, we notice that it also slightly improves over Π_b-SPIBB on the mean normalized performance. As far as safety is concerned, we see that the 1st percentile of policies computed with $\widehat{\Pi}_b$-SPIBB falls close to the 1st percentile of the estimated baseline $\widehat{\pi}_b$ for small datasets and close to the 1st percentile of the policies Π_b-SPIBB for datasets with around 100 trajectories. It is expected as $\widehat{\Pi}_b$-SPIBB tends to reproduce the baseline for very small datasets and improves over it for larger ones. That statement is also true of $\widehat{\Pi}_b$-Soft.

RaMDP and Basic RL. Finally, it is interesting to observe that although RaMDP and Basic RL can compute policies with high mean performance, these algorithms often return policies performing much worse than the MLE policy $\widehat{\pi}_b$ (as seen in their 1st percentile).

The easy setting. Figure 4.2 shows the results for χ = 0.1, so the performance of the baseline policy is low, and it is easy to find an improved policy. We notice that all algorithms show a reliable performance improvement over the baseline. The 1st percentile of the estimated policy is slightly below the true behavior policy with small datasets and, naturally, the $\widehat{\Pi}_b$-SPIBB shows similar behavior.

4.4.2. CONTINUOUS MDPS

In this section we consider MDPs with a continuous state space $\mathbb{S} \subseteq \mathbb{R}^n$, in this case a state is a vector $\mathbf{s} \in \mathbb{S}$, nevertheless, for simplicity we maintain the notation as $s \in \mathbb{S}$. A challenge to apply the SPIBB algorithms in this setting is the requirement of counting how often a state-action was observed.

HELICOPTER DOMAIN

For MDPs with continuous state space, we focus on the helicopter environment (Laroche et al., 2019, Figure 4.3). In this stochastic domain, the state is defined by the position

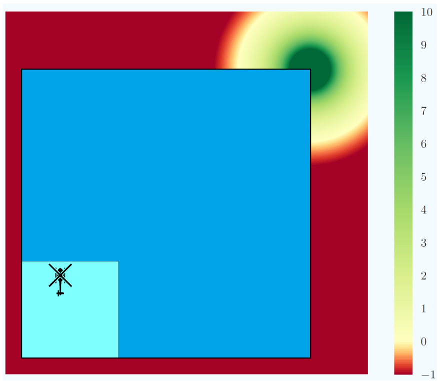


Figure 4.3: The helicopter environment (Laroche et al., 2019).

and velocity of the helicopter. The agent has a discrete set of 9 actions to control the thrust applied in each dimension. The helicopter begins in a random position of the bottom-left corner with a random initial velocity. The episode ends if the helicopter's speed exceeds some threshold, giving a reward of -1, or if it leaves the valid region, in which case the agent gets a reward between -1 and 10 depending on how close it is to the top-right corner.

Behavior Cloning

In infinite MDPs, there is no MLE baseline definition. We have to lean on behavior cloning techniques. We compare here two straightforward ones in addition to the true behavior policy π_b: a baseline estimate $\widehat{\pi}_c$ based on the same pseudo-counts used by the algorithms, and a neural-based baseline estimate $\widehat{\pi}_n$ that uses a standard probabilistic classifier.

The *count-based policy* follows a principle similar to the MLE policy. It uses a pseudo-count for state-action pairs $\tilde{\eta}(s,a)$ defined according to the sum of the euclidean distance $\|s-s'\|_2$ from the state s and all states of transitions in the dataset where the action a was executed (Laroche et al., 2019, Section 3.4):

$$\tilde{\eta}_{\mathcal{D}}(s,a) = \sum_{\left\{\left\langle s_j,a_j,r_j,s'_j\right\rangle\in\mathcal{D}\,\middle|\,a_j=a\right\}} \max\left(0, 1-\frac{\|s-s_j\|_2}{d_0}\right), \qquad \forall s,a\in\mathbb{S}\times\mathbb{A},$$

where d_0 is a hyper-parameter to impose a minimum similarity before increasing the counter of a certain state. We also compute the state pseudo-count using this principle:

$$\tilde{\eta}_{\mathcal{D}}(s) = \sum_{a\in\mathbb{A}} \tilde{\eta}_{\mathcal{D}}(s,a), \qquad \forall s\in\mathbb{S}.$$

This way, we can define the count-based baseline estimate replacing the count in Equation 4.6 by its pseudo-count counterpart:

$$\widehat{\pi}_c(a\mid s) = \begin{cases} \frac{\tilde{\eta}_{\mathcal{D}}(s,a)}{\tilde{\eta}_{\mathcal{D}}(s)} & \text{if } \tilde{\eta}_{\mathcal{D}}(s)>0, \\ \frac{1}{|\mathbb{A}|} & \text{otherwise,} \end{cases} \qquad \forall s,a\in\mathbb{S}\times\mathbb{A}. \tag{4.11}$$

The *neural-based policy* $\widehat{\pi}_n(a \mid s)$ is estimated using a supervised learning approach. We train a probabilistic classifier using a neural network to minimize the negative log-likelihood with respect to the actions in the dataset.

We use the same architecture as the one used to train the Double-DQN models, which is shared among all the algorithms in the helicopter domain experiments: a fully connected neural network with 3 hidden layers of 32, 128, and 28 neurons, respectively, and 9 outputs corresponding to the 9 actions.

To avoid overfitting, we split the dataset into two parts: 80% for training and 20% for validation. During training, we evaluate the classifier on the validation dataset at the end of every epoch and keep the network with the smallest validation loss.

COMPETING ALGORITHMS

We consider results for 11 algorithms and baselines:

- π_b: the true baseline,
- $\widehat{\pi}_c$: the pseudo-count-based estimate of the baseline,
- $\widehat{\pi}_n$: the neural-based estimate of the baseline,
- $\Pi_b/\widehat{\Pi}_c/\widehat{\Pi}_n$-SPIBB: SPIBB with their respective baselines,
- $\Pi_b/\widehat{\Pi}_c/\widehat{\Pi}_n$-Soft: Soft-SPIBB with their respective baselines,
- RaMDP: Double-DQN with Reward-adjusted MDP, and
- Double-DQN: basic deep RL algorithm.

HYPER-PARAMETERS

Building on the results presented by Nadjahi et al. (2019), we set the hyper-parameters for the experiments with $|\mathscr{D}| = 10,000$ ($|\mathscr{D}| = 3,000$) as follows: Π_b-SPIBB with $m = 3$ ($m = 1$), Π_b-Soft with $\sigma = 0.6$ ($\sigma = 0.8$), and RaMDP with $\kappa = 1$ ($\kappa = 1.75$). For the algorithms using an estimated baseline we run a parameter search considering $m \in [2,3,4,5]$ ($m \in [0.5,1,2,3]$) for SPIBB and $\sigma \in [0.4,0.6,0.8,1]$ ($\sigma \in [0.6,0.8,1,1.2,1.5,1.8,2]$) for Soft-SPIBB and set the parameters for the main experiments as follows: $\widehat{\Pi}_n$-SPIBB and $\widehat{\Pi}_c$-SPIBB with $m = 3.0$ ($m = 1.0$), and $\widehat{\Pi}_n$-Soft and $\widehat{\Pi}_c$-Soft with $\sigma = 0.6$ ($\sigma = 0.8$).

PERFORMANCE INDICATORS

The plots represent for each algorithm a modified box-plot where the caps show the 10th percentile and 90th percentile, the upper and lower limits of the box are the 25th and 75th percentiles, and the middle line in black shows the median. We also show the average of each algorithm (dashed lines in green) and finally add a swarm-plot to enhance the distribution visualization. The table provides additional details, including the percentage of policies that showed a performance above the average performance of the true baseline policy.

THE BASELINE POLICY

To compute the baseline policy, we execute an online DQN algorithm on the underlying problem but stop before convergence. Then, we apply a softmax function on the Q-network to obtain the behavior policy π_b used on the experiments.

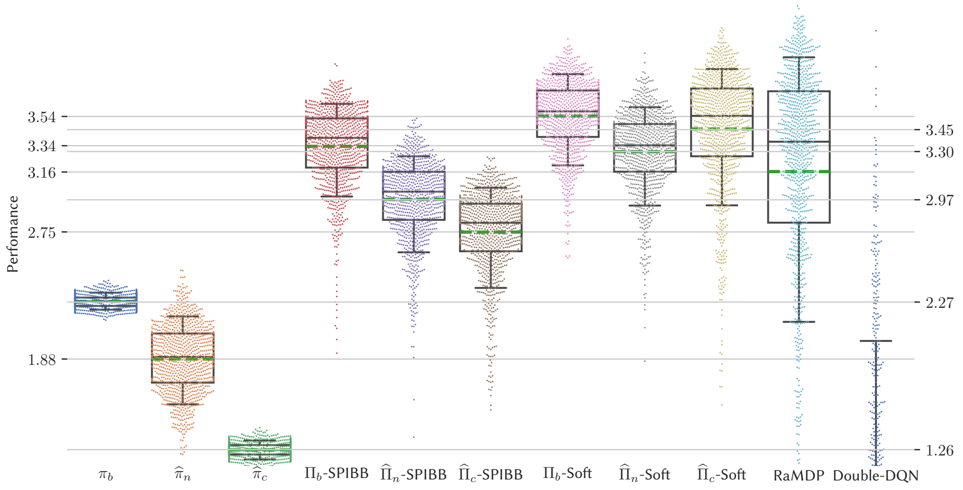


Figure 4.4: $|\mathscr{D}| = 10,000$. The green dashed line shows the average and the caps show the 10% and 90% percentile. Each dot on the swarm plots displays the evaluation of a seed.

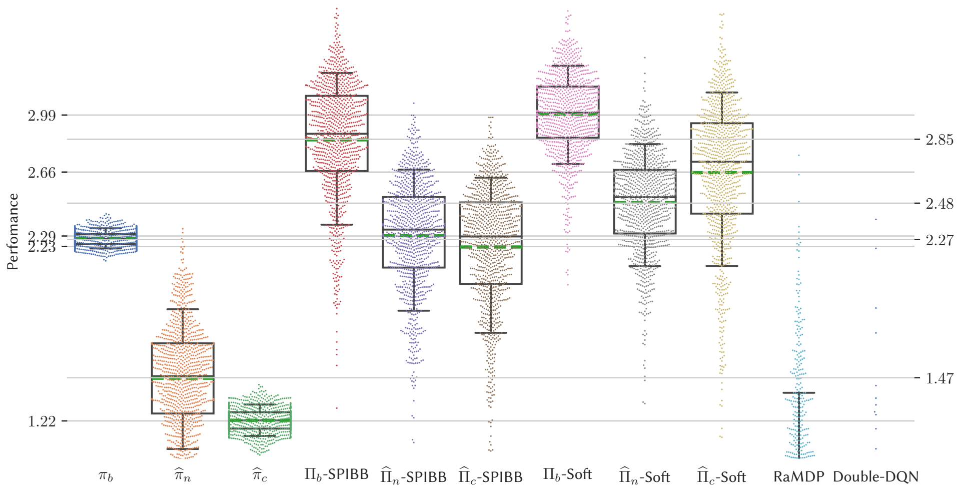


Figure 4.5: $|\mathscr{D}| = 3,000$. The green dashed line shows the average and the caps show the 10% and 90% percentile. Each dot on the swarm plots displays the evaluation of a seed.

Results

Using a fixed behavior policy π_b we generate 1,000 datasets for each algorithm. We report results for two dataset sizes: 3,000 and 10,000 transitions. The results are reported numerically in Table 4.1 and graphically on Figure 4.4 for $|\mathscr{D}| = 10,000$ and Figure 4.5 for $|\mathscr{D}| = 3,000$.

		$\lvert\mathscr{D}\rvert = 3,000$				$\lvert\mathscr{D}\rvert = 10,000$			
Algorithm	π	$\Pr(V(\pi) > V(\pi_b))$	avg perf	10%-qtl	1%-qtl	$\Pr(V(\pi) > V(\pi_b))$	avg perf	10%-qtl	1%-qtl
	π_b	0.499	2.27	2.22	2.18	0.499	2.27	2.22	2.18
Baseline	$\widehat{\pi}_n$	0.002	1.47	1.06	0.75	0.032	1.88	1.57	1.34
	$\widehat{\pi}_c$	0.000	1.22	1.13	1.05	0.000	1.26	1.19	1.14
	π_b	0.928	2.85	2.36	1.90	0.992	3.34	2.99	2.39
SPIBB	$\widehat{\pi}_n$	0.582	2.29	1.86	1.43	0.973	2.97	2.61	2.15
	$\widehat{\pi}_c$	0.514	2.23	1.73	1.21	0.930	2.75	2.37	1.75
	π_b	0.990	2.99	2.71	2.31	1.000	3.54	3.21	2.82
Soft-SPIBB	$\widehat{\pi}_n$	0.760	2.48	**2.12**	**1.71**	**0.996**	3.30	**2.93**	**2.47**
	$\widehat{\pi}_c$	**0.785**	**2.66**	**2.11**	1.51	0.980	**3.45**	**2.93**	2.09
RaMDP		0.006	0.37	-0.75	-0.99	0.876	3.16	2.13	0.23
Double-DQN		0.001	-0.77	-1.00	-1.00	0.076	0.25	-0.97	-1.00

Table 4.1: Numerical results for the two size of datasets. The key performance indicators are respectively the percentage of policy improvement over the true baseline, the average performance of the trained policies, the 10th-percentile, and the 1st percentile. For each column, we bold the best performing algorithm that is not using the true baseline π_b.

Empiric baseline policies. On Figure 4.4, we observe that the baseline policies $\widehat{\pi}_c$ and $\widehat{\pi}_n$ have a performance poorer than the true behavior policy π_b. On the one hand, the neural-based baseline estimate $\widehat{\pi}_n$ can get values close to the performance of the true behavior policy; however, it has a high variance, and even the 90th percentile is below the mean of the true policy. On the other hand, the count-based policy $\widehat{\pi}_c$ has a low variance, but it has a much lower mean performance. In general, we observe a larger performance loss than in finite MDPs between the true baseline and the estimated baseline.

SPIBB. With SPIBB, the neural-based baseline estimate leads to better results for all indicators. The loss in average performance makes it worse than RaMDP in the $\lvert\mathscr{D}\rvert = 10,000$ datasets, but it is more reliable and yields more consistently to policy improvements. On the $\lvert\mathscr{D}\rvert = 3,000$ datasets (Figure 4.5), it demonstrates a higher robustness with respect to the small datasets, still compared to RaMDP.

Soft-SPIBB. The Soft-SPIBB results with baseline estimates are impressive. The loss of performance with respect to Soft-SPIBB with the true baseline is minor. We highlight that, although the policy based on pseudo-counts has a lower performance than the true one (1 point difference), it still achieves a strong performance when used with Soft-SPIBB (less than 0.1 point difference). This indicates that the proposed method is robust with respect to the performance of the estimated policy. It seems that Soft-SPIBB changes are much more forgiving than the baseline approximations.

Small dataset. The experiment with a small dataset $\lvert\mathscr{D}\rvert = 3,000$ (Figure 4.5) aims to evaluate the robustness of these algorithms. We observe that the estimated policies have a performance even lower than in the experiment with $\lvert\mathscr{D}\rvert = 10,000$. While RaMDP's performance indicators dramatically plummet, even largely lower than the behavior cloning policies, the algorithm SPIBB using the estimated policies usually returns policies with a performance similar to the true baseline π_b. Most exciting, the algorithm Soft-SPIBB

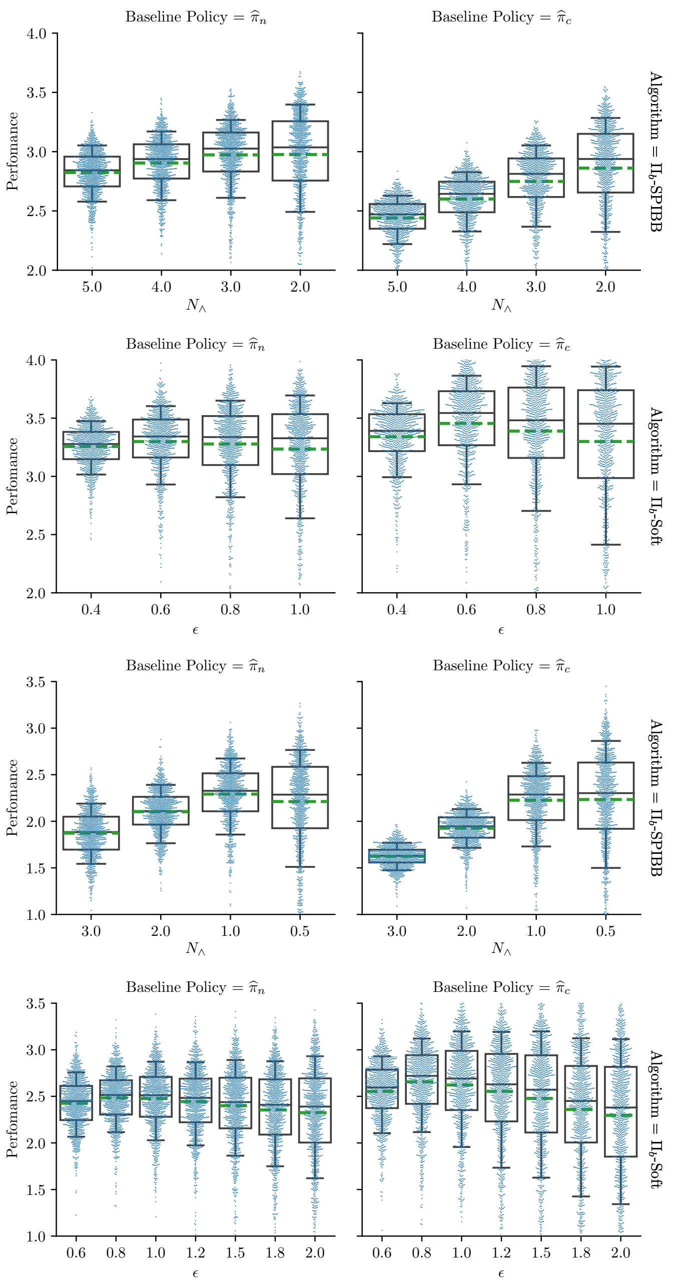


Figure 4.6: $|\mathscr{D}| = 10{,}000$ in the two first rows and $|\mathscr{D}| = 3{,}000$ in the two last rows. The green dashed line shows the average and the caps show the 10th and 90th percentile.

manages to improve upon π_b with all the baselines policies, obtaining a mean performance above the average performance of π_b, and a 10th-percentile slightly lower than that of the true baseline when using the estimated policies.

Hyper-parameter sensitivity. The hyper-parameter search (Figure 4.6) gave us extra insights on the behavior of the algorithms SPIBB and Soft-SPIBB using estimated baselines. We noticed that these algorithms do not have a high sensitivity to their hyper-parameters since the performance is stable in a wide range of values, especially the Soft-SPIBB variations. We sometimes notice a trade-off that has to be made between variance reduction and expectation maximization.

4.5. CONCLUSIONS

This work addresses the problem of performing safe policy improvement in batch RL without direct access to the baseline, *i.e.*, the behavior policy of the dataset. We provide the first theoretical guarantees for safe policy improvement in this setting and show on finite and continuous MDPs that the algorithm is tractable and significantly outperforms all competing algorithms that do not have access to the baseline. We also empirically confirm the limits of the approach when the number of trajectories in the dataset is low.

Currently, the limitation of SPIBB methods is the lack of algorithms to estimate the parametric uncertainty of the estimated model. Bellemare et al. (2016); Burda et al. (2019); Fox et al. (2018) investigated some methods for optimism-based exploration, which proved to not be robust enough for pessimism-based purposes where there is a requirement for exhaustiveness. Further research on this issue is warranted, but also on the multi-batch setting where there are several sequential updates (Laroche and Tachet des Combes, 2019) and on problems with continuous action space (Kumar et al., 2019).

5 Safe Reinforcement Learning During Training

Deploying reinforcement learning (RL) involves major concerns around safety. Engineering a reward signal that allows the agent to maximize its performance while remaining safe is not trivial. Safe RL studies how to mitigate such problems. For instance, we can decouple safety from reward using constrained Markov decision processes (CMDPs), where an independent signal models the safety aspects. In this setting, an RL agent can autonomously find trade-offs between performance and safety. Unfortunately, most RL agents designed for CMDPs only guarantee safety after the learning phase, which might prevent their direct deployment. In this chapter, we investigate settings where a concise abstract model of the safety aspects is given, a reasonable assumption since a thorough understanding of safety-related matters is a prerequisite for deploying RL in typical applications. Factored CMDPs provide such compact models when a small subset of features describe the dynamics relevant for the safety constraints. We propose an RL algorithm that uses this abstract model to learn policies for CMDPs safely. During the training process, this algorithm can seamlessly switch from a conservative policy to a greedy policy without violating the safety constraints. We prove that this algorithm is safe under the given assumptions. Empirically, we show that even if safety and reward signals are contradictory, this algorithm always operates safely and, when they are aligned, this approach also improves the agent's performance. Finally, we study how to reduce the performance regret of this algorithm, without sacrificing the safety guarantees.

This chapter is based on a paper presented at AAMAS-21 (Simão et al., 2021).

As we discussed in Chapter 1, despite all the astonishing successes in Reinforcement Learning (RL; Sutton and Barto, 2018), safe exploration is still a major concern preventing the deployment of RL in real world tasks (Amodei et al., 2016). This problem has motivated the study of Constrained Reinforcement Learning (CRL) to ensure safety (Dulac-Arnold et al., 2021). In this framework, an agent interacts with an environment modeled as a Constrained Markov Decision Process (CMDP; Altman, 1999)[1] without knowing the transition, reward, and cost functions. In SRL, the cost function is used as a proxy to distinguish between safe and unsafe behaviors (García and Fernández, 2015). Therefore, the agent must find a policy with maximum expected reward among the safe (feasible) policies, namely those with expected cost smaller than a safety threshold.

We would like to distinguish between two CRL settings. The first is the *common* setting, where the agent trains in an assumed perfect simulator and only cares about constraint violations later, when deployed in the real environment. In this case, safe exploration is not a major issue since the agent is free to explore during the learning period. The second, which is the focus of this chapter, is what we may call the *true* setting, where the agent interacts directly with the environment and is not allowed to violate the constraints while learning.

Following the *optimism in the face of uncertainty* framework to trade-off exploration and exploitation, Efroni et al. (2020) proposed different algorithms for CRL with bounded regret in terms of performance and in terms of constraint violation. However, these algorithms may still violate the constraints since they encourage the agent to explore unknown parts of the environment, making them unsuitable for the true RL setting. We aim to develop RL algorithms that can learn without violating the constraints, that is, with no regret in terms of constraint violation.

This leads us to Research Question 5. We observe that often most of the state description is only relevant to the reward signal and does not influence the safety of the agent. In this setting, it can be easy for an expert to define the dynamics relevant for safety. Consider, for instance, imposing a limit on the consecutive movements of a robot arm to avoid overheating or indicating unsafe areas such as stairs on a mobile robot's map where the target location is not relevant for safety. Such constraints may be represented in a compact model and are a prerequisite for deploying RL in practice. Without such knowledge, typical RL algorithms would need to perform random exploration, which is not an option in safety-critical applications. Hence, we assume that this compact model is known and is represented by an abstract CMDP $\overline{\mathcal{M}}$.

Notice that by assuming access to the abstract model $\overline{\mathcal{M}}$, we are effectively restricting our method to a subset of problems, therefore dealing with Research Question 1, regarding which classes of problems are suitable for SRL. This assumption allows us to safely trade-off exploration and exploitation, so the agent has an incentive to explore, but *always within a set of safe policies.* Figure 5.1 shows a CMDP where this kind of abstraction can be found. In this example, we observe that the variable y does not influence the cost function. We will use this problem as a running example henceforth.

Constraining the agent to always execute safe policies changes the exploration process, which raises Research Question 6 concerned to how the safe exploration affects the performance of an agent. Therefore, we also investigate the exploration capabilities of

[1]See Section 2.3 for a formal definition.

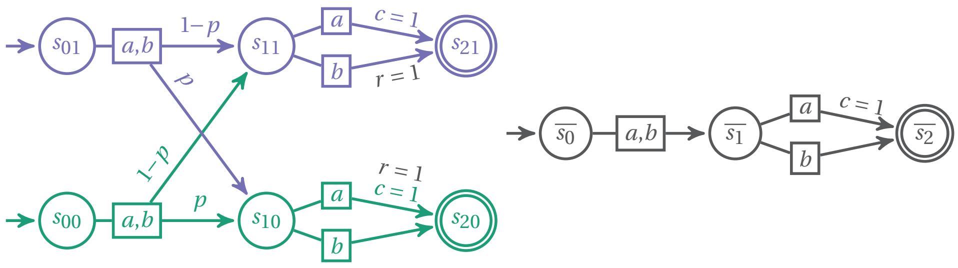


Figure 5.1: A factored CMDP with 2 features (x, y) and 6 states (left) and the corresponding abstract CMDP built with a cost-model-irrelevant abstraction that ignores the feature y (right). Costs and rewards with value 0 are omitted as well as the probability of deterministic transitions.

the proposed algorithm and how it balances between exploration and safety.

This work uses abstractions in a novel way. The literature usually focuses on building a policy for $\overline{\mathcal{M}}$ that will later be executed in the ground CMDP $\mathcal{M}$. Our approach, however, focuses on computing a policy in the ground CMDP $\mathcal{M}$ and uses the abstract model $\overline{\mathcal{M}}$ to guarantee safety, which decouples the safety concerns from the reward.

The contributions of this chapter are four-fold:

i. we study the kind of abstraction sufficient to concisely describe and distill safety dynamics;

ii. we devise an example of such an abstract model using Factored Markov Decision Processes (Boutilier et al., 1995);

iii. assuming the abstract model of the safety dynamics is given, we propose a safe algorithm that learns an optimal policy for the CMDP without violating the constraints; and

iv. we show that this algorithm is always safe and has no regrets in terms of constraint violation.

The empirical analysis showcases the overall capabilities of the proposed algorithm:

i. as expected, it respects the constraints during training;

ii. given enough budget for exploration, it eventually achieves optimal performance; and

iii. when the cost function is aligned with the reward function, it reduces the performance regret.

The code to reproduce all experiments is available online[2].

[2]`https://github.com/AlgTUDelft/AlwaysSafe`.

5.1. CONSTRAINED REINFORCEMENT LEARNING

In Reinforcement Learning (RL; Sutton and Barto, 2018), the agent does not have access to the description of the underlying MDP, so it must find a balance between *exploration*, to learn about the environment potentially increasing its performance, and *exploitation*, taking advantage of its current knowledge to collect rewards, as discussed in Section 2.4. The difference between the value of the policy executed and the value of the optimal policy, called *regret*, is one measure of the efficiency of RL algorithms. Intuitively, an agent with bounded regret can make a good trade-off between exploration and exploitation.

In CRL, the agent interacts with a CMDP instead of an MDP. This changes the efficiency metrics used to evaluate the agent, as well as the way the agent solves the problem, as we discuss in this section.

5.1.1. EFFICIENCY METRICS

To evaluate the efficiency of constrained RL algorithms we may consider two types of regret: *performance regret* and *constraint violation regret* (Efroni et al., 2020). The performance regret is similar to the one in traditional episodic RL settings (Auer and Ortner, 2006; Jaksch et al., 2010; Neustroev and de Weerdt, 2020):

$$\mathrm{Reg}(K,R) = \sum_{k\in\mathbb{N}_K} \left[V_R^{\pi^*}(\mu) - V_R^{\pi_k}(\mu) \right]_+,$$

where $[x]_+ = \max\{x,0\}$ and K is the number of episodes. Note that this definition ignores values larger than the value of the optimal policy (this is only possible if the constraint is violated). The constraint violation regret is the cumulative cost violation:

$$\mathrm{Reg}(K,C) = \sum_{k\in\mathbb{N}_K} \left[V_C^{\pi_k}(\mu) - \hat{c} \right]_+.$$

In this case, we note that there is only regret when the expected cost is higher than the given bound, so a policy has no regret for having an expected cost lower or equal to the bound. In the next section, we describe an algorithm for CRL with bounded regret.

5.1.2. SOLVING CMDPS WITH OPTIMISM

OptCMDP (Efroni et al., 2020, Algorithm 6) extends the UCRL2 algorithm (Jaksch et al., 2010) to the CMDP setting. This algorithm requires no knowledge about the components of the CMDP and bounds the performance regret with respect to the optimal policy as well as the constraint violation regret.

Intuitively, at the beginning of each episode, OptCMDP defines a set of CMDPs Σ that contains the true CMDP $\mathcal{M}$ with high probability $1-\delta$. It computes an optimistic policy, assuming it can also choose the best CMDP in Σ:

$$\operatorname*{arg\,max}_{P',R',C'\in\Sigma,\pi\in\Sigma} \quad V_{R'}^{\pi_k}(\mu) \quad \text{s.t. } V_{C'}^{\pi_k}(\mu) \le \hat{c}. \tag{5.1}$$

Then, the algorithm executes the optimistic policy for one episode, collecting data to update Σ. Over time the set Σ shrinks, and the optimistic policy approaches the optimal policy.

Algorithm 6 OptCMDP

```
Input: $\delta \in (0,1)$ : confidence level.
1: for $k \in [1, \cdots, K]$ do
2:     Update the empirical model.
3:     Compute $\pi_k$ with LP2.
4:     Execute policy $\pi_k$ for one episode.
5: end for
```

Efroni et al. (2020) show that, under an optimistic perspective, we can simplify the problem in Equation 5.1 by choosing the upper bound of the reward function and the lower bound of the cost function. Therefore, using $\hat{f}$ to denote the maximum likelihood estimate of the function f, we only consider a set of transition functions when defining Σ:

$$\Sigma = \left\{ P', R', C' \;\middle|\; \begin{array}{l} P' \in \left[\hat{P}(s' \mid s,a) - e^P_\delta(s,a,s'), \hat{P}(s' \mid s,a) + e^P_\delta(s,a,s')\right], \\ R' = \hat{R}(s,a) + e^R_\delta(s,a), \\ C' = \hat{C}(s,a) - e^C_\delta(s,a), \forall s,a,s' \in \mathbb{S}, \mathbb{A}, \mathbb{S} \end{array} \right\},$$

where the confidence intervals e^P_δ, e^R_δ and e^C_δ are based on the Bernstein inequality (for the transition function) and Hoeffding's inequality (for the cost and reward function) (Hoeffding, 1963). We refer to Efroni et al. (2020, Equation 20) for a detailed derivation of the confidence intervals such that $\Pr(\mathcal{M} \in \Sigma) \geq 1 - \delta$. Now we can solve the problem in Equation 5.1 with the following optimistic LP (Efroni et al., 2020; HasanzadeZonuzy et al., 2021; Rosenberg and Mansour, 2019):

$$\max \sum_{s,a,t \in \mathbb{S} \times \mathbb{A} \times \mathbb{N}_H} y_t(s,a)(\hat{R}(s,a) + e^R_\delta(s,a)) \quad \text{s.t. } \underbrace{\text{C3–C5}}_{\text{LP1}}, \text{C7–C9}. \tag{LP2}$$

$$\sum_{s,a,t \in \mathbb{S} \times \mathbb{A} \times \mathbb{N}_H} y_t(s,a)(\hat{C}(s,a) - e^C_\delta(s,a)) \leq \hat{c}. \tag{C7}$$

$$x_t(s,a,s') \leq (\hat{P}(s' \mid s,a) + e^P_\delta(s,a,s'))y_t(s,a) \qquad \forall (s,a,s',t) \in \mathbb{S} \times \mathbb{A} \times \mathbb{S} \times \mathbb{N}_H. \tag{C8}$$

$$x_t(s,a,s') \geq (\hat{P}(s' \mid s,a) - e^P_\delta(s,a,s'))y_t(s,a) \qquad \forall (s,a,s',t) \in \mathbb{S} \times \mathbb{A} \times \mathbb{S} \times \mathbb{N}_H. \tag{C9}$$

Compared to LP1, LP2 replaces the equality C6 by two inequalities, which ensures that the chosen transition function is close to the true transition function with high probability. Besides an optimistic policy, computed by Equation 2.4, a solution for this LP also gives us the optimistic transition function picked for problem in Equation 5.1:

$$P'_t(s' \mid s,a) = \frac{x_t(s,a,s')}{y_t(s,a)}, \qquad \forall t \in \mathbb{N}_H.$$

Over the episodes, as the agent collects more experiences, the estimate $\hat{P}$ improves as the confidence interval e^P_δ narrows. This way, P' approaches P and the policy computed by LP2 gets closer to an optimal one.

Even though the OptCMDP algorithm has bounded constraint violation regret, in safety-critical applications, even a small regret would not be acceptable, so this algorithm could not be directly deployed in real world tasks. In the next section, we present a model that allows us to compactly represent a CMDP. Later, we will use this model to define a compact abstraction of the dynamics that are relevant for the safety constraints and use it to build a CRL algorithm with no constraint violation regret.

5.2. Abstraction for Expected Cost

Before describing our method, let us consider the robot with an arm that can overheat again. In this example, we could keep track of how often the motor was activated in the last hour and constraint the RL agent's policies to avoid excessive consecutive movements. This is a simple example of an abstraction that lets the robot act safely. In this section, we will formalize an abstract version of the problem that captures the knowledge required to ensure safety in the CRL setting. As an example, we explore factored CMDPs where the cost function is independent of some variables, namely, only a subset of the variables is relevant for the constraints.

Following the definitions by Li et al. (2006), we denote a state abstraction function by $\phi : \mathbb{S} \to \overline{\mathbb{S}}$, where $\overline{\mathbb{S}}$ is the finite abstract state space. The inverse of ϕ is denoted by $\phi^{-1} : \overline{\mathbb{S}} \to 2^{\mathbb{S}}$, that is $\phi^{-1}(\overline{s})$ is the set of ground states whose abstract state is $\overline{s}$ according to ϕ. Given a CMDP $\mathcal{M} = \langle \mathbb{S}, \mathbb{A}, P, R, \gamma, \mu, H, C, \hat{c} \rangle$ and an abstraction function ϕ, the respective abstract CMDP is $\overline{\mathcal{M}}_\phi = \langle \overline{\mathbb{S}}, \mathbb{A}, \overline{P}, \overline{R}, \overline{\mu}, \overline{C}, \hat{c} \rangle$, where

$$\begin{aligned}
\overline{P}(\overline{s'} \mid \overline{s}, a) &= \sum_{s \in \phi^{-1}(\overline{s})} \sum_{s' \in \phi^{-1}(\overline{s'})} w(s) P(s' \mid s, a), \\
\overline{R}(\overline{s}, a) &= \sum_{s \in \phi^{-1}(\overline{s})} w(s) R(s, a), \\
\overline{C}(\overline{s}, a) &= \sum_{s \in \phi^{-1}(\overline{s})} w(s) C(s, a), \\
\overline{\mu}(\overline{s}) &= \sum_{s \in \phi^{-1}(\overline{s})} \mu(s)
\end{aligned}$$

and $w(s)$ indicates the contribution of each state $s \in \phi^{-1}(\overline{s})$ to the abstract state $\overline{s}$, with the constraint that $\sum_{s \in \phi^{-1}(\overline{s})} w(s) = 1$.

5.2.1. Cost-model Irrelevance

Li et al. (2006, Definition 3) use the above formalism to define different types of abstractions. For instance, Q_R^π-irrelevant abstractions preserve the Q_R^π function. This may be useful when solving an MDP, as we could compute Q_R^π over the abstract state space, which can speed up the convergence of an MDP solver (Gopalan et al., 2017) or an RL agent (van Seijen et al., 2014). Following this idea, we define an abstraction related to the model necessary to compute the expected cost V_C^π.

Definition 1. *Given a CMDP $\mathcal{M} = \langle \mathbb{S}, \mathbb{A}, P, R, \gamma, \mu, H, C, \hat{c} \rangle$, we say that an abstraction* *function ϕ is* cost-model-irrelevant *when*

$$\phi(s_1) = \phi(s_2) \Rightarrow \sum_{s' \in \phi^{-1}(\bar{s})} P(s' \mid s_1, a) = \sum_{s' \in \phi^{-1}(\bar{s})} P(s' \mid s_2, a) \text{ and}$$
$$C(s_1, a) = C(s_2, a) \quad \forall a, s_1, s_2, \bar{s} \in \mathbb{A} \times \mathbb{S} \times \mathbb{S} \times \bar{\mathbb{S}}. \tag{5.2}$$

Definition 1 is similar to model-irrelevance (Li et al., 2006), considering the cost function instead of the reward function. It says that if a cost-model-irrelevant abstraction maps two states to the same abstract state, then the cost of executing action $a \in \mathbb{A}$ and the distribution over the next abstract state is the same in both states.

We would like to use prior knowledge of a model of the abstract CMDP to guarantee that a policy for the ground CMDP will not violate the cost constraints. So, while most literature on abstraction for RL is interested in deploying the policy computed in the abstract CMDP to the ground CMDP, we use the abstract CMDP to test the safety of a policy defined in the ground CMDP.

5.2.2. A COST-MODEL-IRRELEVANT ABSTRACTION

Similar to FMDPs (Section 2.2), a factored CMDP compactly represents a CMDP using a DBN. The only addition is the cost function that similarly to the reward function can be succinctly represented by the sum of C local functions, respectively:

$$C(s, a) = \sum_{i \in \mathbb{N}_\mathsf{C}} C_i(s[\Delta_i^C], a),$$

where C_i is the i-th local cost function that only depends on the subset of variables $\Delta_i^C \subseteq \mathbb{X}$.

In factored CMDPs, we can define a cost-model-irrelevant abstraction by considering only the subset of state variables that influence the cost function. Given a set of variables $\Delta \subseteq X$, we define their parents as $\text{Pa}(\Delta) = \bigcup_{X_i, a \in \Delta \times \mathbb{A}} \text{Pa}_a(X_i)$ and their ancestors as:

$$\text{Anc}(\Delta) = \begin{cases} \Delta & \text{if } \text{Pa}(\Delta) \subseteq \Delta, \\ \text{Anc}(\text{Pa}(\Delta) \cup \Delta) & \text{otherwise.} \end{cases}$$

Intuitively, the set $\text{Anc}(\Delta)$ might influence Δ over multiple time steps, while the set $\text{Pa}(\Delta)$ are only variables that have an immediate influence on Δ. Let $\text{Pa}(C) = \cup_{i \in \mathbb{N}_\mathsf{C}} \Delta_i^C$ be the set of variable that directly influences the cost function, we define a cost-model-irrelevant abstraction ϕ_C based on their ancestors $\text{Anc}(\text{Pa}(C))$:

$$\phi_C(s[X]) = s[\text{Anc}(C)]. \tag{5.3}$$

Our running example (Figure 5.1) shows an instance of such abstraction. The efficiency of this abstraction, related to the size reduction from the original CMDP to the abstract CMDP, corresponds to the size of the set of ancestors of the cost function: $|\text{Anc}(C)|$. For instance, if $\text{Anc}(C) = \mathbb{X}$, this abstraction would be the identity function and the abstract CMDP would be the same as the original. If, however, $\text{Anc}(C) = \emptyset$, the abstract CMDP would contain a single state since the cost function is independent of the state variables.

Theorem 5. *ϕ_C is a* cost-model-irrelevant *abstraction.*

Proof. Given $a, s_1, s_2, \overline{s} \in \mathbb{A} \times \mathbb{S} \times \mathbb{S} \times \overline{\mathbb{S}}$. If $\phi_C(s_1) = \phi_C(s_2)$, then we have

$$\begin{aligned} C(s_1, a) &= \sum_{i \in \mathbb{N}_C} C_i(s_1[\Delta_i^C], a) \\ &= \sum_{i \in \mathbb{N}_C} C_i(s_2[\Delta_i^C], a) \\ &= C(s_2, a). \end{aligned}$$

The first and last derivations come from the definition of the factored cost function. The middle derivation comes from the fact that both states were mapped together, so from Equation 5.3 we conclude that $s_1[\Delta_i^C] = s_2[\Delta_i^C] : \forall i \in \mathbb{N}_C$.

In the following derivation, we use $P(s'[\Delta] \mid s, a) = \prod_{X \in \Delta} P(s'[X] \mid s, a)$ where $\Delta \subseteq \mathbb{X}$, $s, a, s' \in \mathbb{S} \times \mathbb{S} \times \mathbb{A}$ and $\overline{\text{Anc}(C)} = \mathbb{X} \setminus \text{Anc}(C)$.

If $\phi_C(s_1) = \phi_C(s_2)$, then we have

5

$$\begin{aligned} \sum_{s' \in \phi^{-1}(\overline{s})} P(s' \mid s_1, a) &\underset{(a)}{=} \sum_{s' \in \phi^{-1}(\overline{s})} P(s'[\mathbb{X}] \mid s_1, a), \\ &\underset{(b)}{=} \sum_{s' \in \phi^{-1}(\overline{s})} P(s'[\text{Anc}(C)] \mid s_1, a) P(s'[\overline{\text{Anc}(C)}] \mid s_1, a), \\ &\underset{(c)}{=} \sum_{s' \in \phi^{-1}(\overline{s})} P(\overline{s}[\text{Anc}(C)] \mid s_1, a) P(s'[\overline{\text{Anc}(C)}] \mid s_1, a), \\ &\underset{(d)}{=} P(\overline{s}[\text{Anc}(C)] \mid s_1, a) \underbrace{\sum_{s' \in \phi^{-1}(\overline{s})} P(s'[\overline{\text{Anc}(C)}] \mid s_1, a)}_{=1(\text{ sum over all values of } \overline{\text{Anc}(C)})}, \\ &\underset{(e)}{=} P(\overline{s}[\text{Anc}(C)] \mid s_1, a) \\ &\underset{(f)}{=} \prod_{X_i \in \text{Anc}(C)} P(\overline{s}[X_i] \mid s_1[\text{Pa}_a(X_i)], a) \\ &\underset{(g)}{=} \prod_{X_i \in \text{Anc}(C)} P(\overline{s}[X_i] \mid s_2[\text{Pa}_a(X_i)], a) \\ &\underset{(h)}{=} \sum_{s' \in \phi^{-1}(\overline{s})} P(s' \mid s_2, a). \end{aligned}$$

In this derivation,

(a) shows we are considering the values of each variable in state s';

(b) decouples ancestors from non-ancestors;

(c) removes the dependence on the state s', since $\forall s' \in \phi_C^{-1}(\overline{s})$, the values of variables in $\text{Anc}(C)$ are the same, by the definition of ϕ_C (Equation 5.3);

(d) factors out the probability term, since it is now independent of s';

(e) removes the summation that results in 1;

(f) uses the conditional independence from the factored CMDP;

(g) swaps s_1 and s_2, since they were mapped to the same abstract state, the values of their parents are the same;

(h) uses the same reasoning from (f) to (a).

□

While ϕ_C is a convenient and natural *cost-model-irrelevant* abstraction, it is not necessarily the most compact. We refer to Givan et al. (2003) for a discussion on how to find more compact abstract models in factored MDPs.

5.2.3. PLANNING WITH THE ABSTRACT CMDP

Given a cost-model-irrelevant abstraction, we extend LP1 to take this knowledge into consideration by adding variables z that represents the occupancy of each pair of abstract states and action for each time step. Our goal is to decouple the expected cost from the full transition function to ensure that the policy computed still respects the cost constraints.

$$\max \sum_{s,a,t \in \mathbb{S}\times\mathbb{A}\times\mathbb{N}_H} y_t(s,a) R(s,a) \quad \text{s.t. } \underbrace{\texttt{C3–C6}}_{\texttt{LP1}}, \texttt{C10–C12}. \tag{LP3}$$

$$\sum_{\bar{s},a,t \in \bar{\mathbb{S}}\times\mathbb{A}\times\mathbb{N}_H} z_t(\bar{s},a)\bar{C}(\bar{s},a) \leq \hat{c}. \tag{C10}$$

$$z_t(\bar{s},a) = \sum_{s\in\phi^{-1}(\bar{s})} y_t(s,a) \qquad \forall \bar{s},a,t \in \bar{\mathbb{S}}\times\mathbb{A}\times\mathbb{N}_H. \tag{C11}$$

$$\sum_{a\in\mathbb{A}} z_t(\bar{s},a) = \sum_{\bar{s}^\circ,a^\circ \in \bar{\mathbb{S}}\times\mathbb{A}} \bar{P}(\bar{s} \mid \bar{s}^\circ, a^\circ) z_{t-1}(\bar{s}^\circ, a^\circ) \qquad \forall \bar{s},t \in \bar{\mathbb{S}}\times\mathbb{N}_H \setminus \{1\}. \tag{C12}$$

In LP3, constraint C11 helps us to connect the flow from the ground CMDP and the abstract CMDP by ensuring that the flow leaving an abstract state is the sum of the flow that leaves the respective ground states. Constraint C10 replaces constraint C1, notice that it uses the abstract cost function and the expected cost is computed according to the occupancy of the abstract CMDP. Finally, constraint C12 ensures that the flow of the abstract CMDP respects the abstract transition function. Although this last constraint is redundant since this flow is already specified in the ground CMDP, it will be important for our method later when we do not have access to the underlying transition function.

Since LP3 keeps variables for the ground CMDP and the abstract CMDP, the policy it computes in the ground CMDP might be different in states that were merged. In other words, the policy that is induced by the abstract variables z is different from the policy that is induced by the ground variables x and y.

In the next section, we show how to use this formulation to devise an RL algorithm compliant with the safety constraints.

Algorithm 7 AbsOptCMDP-π_G

```
Input: $\delta \in (0,1)$ : confidence level.
Input: $\overline{\mathscr{M}}$: cost-model-irrelevant CMDP.
1: for $k \in [1, \cdots, K]$ do
2:     Update the empirical model.
3:     Compute $\pi_k$ using $\overline{\mathscr{M}}$ according to LP4 and Equation 5.5.
4:     Execute policy $\pi_k$ for one episode.
5: end for
```

5.3. Always Safe

Now we consider the setting where the agent has access to the abstract CMDP generated from a cost-model irrelevance abstraction but does not have access to the full transition function or the reward function. We propose an algorithm that computes a policy for the underlying CMDP without incurring any constraint violation regret using an optimistic approach.

5.3.1. The Linear Program

The idea is to combine the abstract CMDP created with a cost-model-irrelevant abstraction (Section 5.2.1) with an optimistic approach for exploration (Section 5.1.2). LP4 puts all the pieces together:

$$\max \sum_{s,a,t \in \mathbb{S}\times\mathbb{A}\times\mathbb{N}_H} y_t(s,a)(\hat{R}(s,a) + e^R_\delta(s,a)) \text{ s.t. } \underbrace{\texttt{C3–C5}}_{\texttt{LP1}}, \underbrace{\texttt{C8–C9}}_{\texttt{LP2}}, \underbrace{\texttt{C10–C12}}_{\texttt{LP3}}. \tag{LP4}$$

The main difference between LP4 and LP2 is the use of the extra variables z that represent the flow in the abstract CMDP. Therefore we replace C7 that constrains the expected cost on the ground CMDP by C10 that constrains the expected cost in the abstract CMDP. The constraints C10–C12 control the flow in the abstract CMDP.

We can compile two basic policies using a solution for LP4. A *ground policy* π_G using y and an *abstract policy* π_A using z:

$$\pi_A(a \mid s,t) = \frac{z_t(\phi(s),a)}{\sum_{a' \in \mathbb{A}} z_t(\phi(s),a')} \qquad \forall s,a,t \in \mathbb{S}\times\mathbb{A}\times\mathbb{N}_H, \tag{5.4}$$

$$\pi_G(a \mid s,t) = \frac{y_t(s,a)}{\sum_{a' \in \mathbb{A}} y_t(s,a')} \qquad \forall s,a,t \in \mathbb{S}\times\mathbb{A}\times\mathbb{N}_H. \tag{5.5}$$

The algorithm AbsOptCMDP-π_G (Algorithm 7) follows policy π_G in each episode. We conjecture that it has a bounded performance regret, inherited from the OptCMDP algorithm, since it makes the same assumptions. However, it can still exhibit some safety violation stemming from the unknown transition function (see Example 5).

Example 5 (π_G might be unsafe.)**.** *Let us consider the CMDP from Figure 5.1 from an optimistic perspective. We may assume that our estimate of transition function is perfect, $\hat{p} = p$, but we still have some uncertainty about it, represented by $e(p)$. This way, we know $p \in [\hat{p} - e(p), \hat{p} + e(p)]$. In the optimistic CMDP, we are also picking the parameters of the* *transition function. This means the agent chooses the value of p, which in this case would be the lower bound $\hat{p} - e(p)$ since it minimizes the chance of reaching state s_{10}. Following the same reasoning as in Example 2, we find a greedy policy $\pi_G(a \mid s_{10}) = \frac{\hat{c}}{p-e(p)}$ which is unsafe, since it is larger than the maximum value we found in Example 2 ($\pi^*(a \mid s_{10}) = \frac{\hat{c}}{p}$).*

Theorem 6. *Given an uncertainty set Σ containing the underlying CMDP $\mathcal{M}$, and the abstract CMDP $\overline{\mathcal{M}}_\phi = \langle \overline{\mathbb{S}}, \mathbb{A}, \overline{P}, \overline{R}, \overline{\mu}, \overline{C}, \hat{c} \rangle$ built according to a cost-model-irrelevant abstraction ϕ, the policy π_A computed according to LP4 and Equation 5.4 does not violate the constraints when applied in $\mathcal{M}$: $V_C^{\pi_A}(\mu) \leq \hat{c}$.*

Proof sketch. Li et al. (2006) show that a model-irrelevant abstraction preserves the expected value. In the same way, a cost-model-irrelevant abstraction preserves the expected cost. So, the policy computed in the abstract state has the same expected cost in the ground state, $V_C^{\pi_A}(\mu) = V_{\overline{C}}^{\pi_A}(\overline{\mu})$. From the constraint C10 in LP4, we have $V_{\overline{C}}^{\pi_A}(\overline{\mu}) \leq \hat{c}$. Therefore, $V_C^{\pi_A}(\mu) \leq \hat{c}$. □

Theorem 6 essentially shows that the policy π_A is *safe*, independent of the uncertainty over the transition function. However, this policy is not expressive enough to describe an optimal policy for the underlying CMDP; for instance, its domain might ignore variables that influence the reward function (see Example 6).

Example 6 (π_A might be sub-optimal.). *Considering the CMDP from Figure 5.1 again, we may notice that a policy defined in the abstract MDP would assign at most probability $\hat{c}$ to action a in the abstract state $\overline{s_1}$, while a policy defined on the ground state can distinguish between s_{10} and s_{11}, as we saw in Example 2.*

5.3.2. POLICIES

In this section, we study how to switch between π_A and π_G, to find an RL algorithm that has no constraint regret and can still find an optimal policy for the underlying CMDP.

POLICY π_T

To devise an algorithm that can eventually find an optimal policy for the underlying CMDP, we propose to use the ground policy based on a test:

$$\pi_T = \begin{cases} \pi_G & \text{if } \max_{P' \in \Sigma} V_C^{\pi_G}(\mu) \leq \hat{c} \\ \pi_A & \text{otherwise.} \end{cases} \tag{5.6}$$

To test if we can deploy π_G, we compute the maximum expected cost within the uncertainty set, fixing the policy π_G:

$$\begin{aligned} \max_{x,y,z} \quad & \sum_{\overline{s},a,t \in \overline{\mathbb{S}} \times \mathbb{A} \times \mathbb{N}_H} z_t(\overline{s},a)\overline{C}(\overline{s},a) \\ \text{s.t.} \quad & \underbrace{\text{C3–C5}}_{\text{LP1}}, \underbrace{\text{C8–C9}}_{\text{LP2}}, \underbrace{\text{C11}}_{\text{LP3}}, \text{C13}. \end{aligned} \tag{LP5}$$

$$y_t(s,a) = \pi_G(a \mid s,t) \sum_{a' \in \mathbb{A}} y_t(s,a') \qquad \forall s,a,t \in \mathbb{S} \times \mathbb{A} \times \mathbb{N}_H. \tag{C13}$$

Algorithm 8 Dynamic Constraint Tightening (π_α)

```
Input: $\Sigma$: uncertainty set
Input: $\overline{\mathcal{M}}$: abstract model
Input: $\alpha$: learning rate
Input: $\beta$: coefficient lower bound (default: 0)
 1: $\beta \leftarrow \overline{1}$
 2: repeat
 3:     $y, z, status \leftarrow$ solve LP4 with $\beta\hat{c}$
 4:     if ($status$ is infeasible) or ($\beta \leq \underline{\beta}$) then
 5:         $\pi_\alpha \leftarrow \pi_A$                                   ▷ Equation 5.4 based on z
 6:         return $\pi_\alpha$
 7:     end if
 8:     $\pi_\alpha \leftarrow \pi_G$                                       ▷ Equation 5.5 based on y
 9:     $maxV \leftarrow \max_{P' \in \Sigma} V_C^{\pi_\alpha}(\mu)$            ▷ LP5 based on $\pi_\alpha$
10:     $\beta \leftarrow \beta - \alpha \frac{\max\{maxV - \hat{c}, 0\}}{\hat{c}}$
11: until $maxV \leq \hat{c}$
12: return $\pi_\alpha$
```

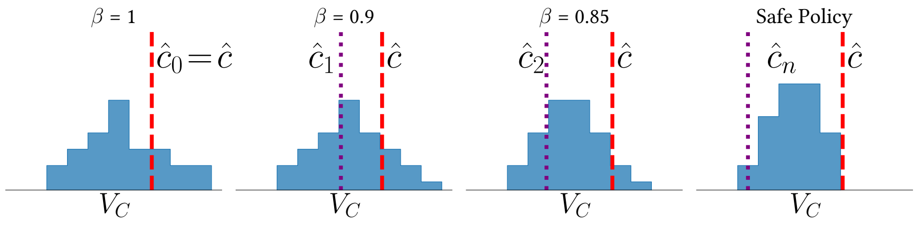


Figure 5.2: Search for a ground policy that respects the constraints in all CMDPs from the uncertainty set Σ. The x-axis indicates the expected cost, and the y-axis indicates the frequency we can find a CMDP in Σ for which the policy computed has that expected cost.

In this LP, the constraint C13 is derived from Equation 2.4 to ensure that the policy π_G is applied in every ground state. The value of $\pi_G(a \mid s, t)$ are constants computed by Equation 5.5 according to the solution of LP4, for every state, action and time step. Intuitively, LP5 chooses the transition function in the uncertainty set Σ with the highest expected cost.

Although this approach could be more efficient, for instance if we tested whether $\exists P' \in \Sigma : V_C^{\pi_G}(\mu) > \hat{c}$, we opt to compute the maximum expected cost, because it gives us an indication of how much the constraint might be violated. This can help us find a more conservative policy in the ground CMDP, as we describe next.

Policy π_α

The previous solution may never choose the ground policy π_G, in particular when the optimal policy has an expected cost close to the bound $\hat{c}$. In this case, even a small confidence interval could put the maximum expected cost above the given cost bound. Inspired by de Nijs et al. (2017), we propose to compute a policy that is more conservative

Algorithm 9 AlwaysSafe

```
Input: $\delta \in (0,1)$ : confidence level.
Input: $\overline{\mathcal{M}}$: cost-model-irrelevant CMDP.
1: for $k \in [1, \cdots, K]$ do
2:     Update the empirical model.
3:     Compute $\pi_k$ using $\overline{\mathcal{M}}$ and Equation 5.4, Equation 5.6 or Algorithm 8.
4:     Execute policy $\pi_k$ for one episode.
5: end for
```

such that it passes the test from Equation 5.6. Our strategy is to compute policies for a tighter bound $\hat{c}$[3].

Algorithm 8 describes one way to compute such a policy. The algorithm initializes the coefficient β with value 1. Then, in each iteration, the algorithm solves LP4 using an adjusted bound $\beta\hat{c}$. If the algorithm does not meet any of its stopping criteria, it lowers the coefficient β and repeats. The algorithm can terminate in three ways: (*i*) by finding a policy that respects the constraints in all CMDPs in the uncertainty set; (*ii*) by setting a cost bound too low, such that none of the CMDPs can satisfy the constraints; or (*iii*) by setting the coefficient β below the lower bound $\underline{\beta}$.

Figure 5.2 demonstrates a successful search for a safe ground policy with Algorithm 8. Each plot shows the distribution of expected cost (according to the CMDPs in Σ) for policies computed with a certain bound $\hat{c}_i$. The first three plots show how the cost bound $\hat{c}_i$ changes over the iterations, and the last plot shows one of the stopping conditions of the algorithm: when the policy computed according to $\hat{c}_n$ respect the original constraints in all CMDPs in Σ.

Using the coefficient lower bound $\underline{\beta}$, we can limit how tight the safety constraint can get, which ultimately allows the dynamic constraint tightening algorithm to return the abstract policies instead of the ground policy. This can help us prevent the execution of a ground policy that is overly conservative, which might be of interest when the abstract policy already has a reasonable performance.

5.3.3. THE ALGORITHM

The AlwaysSafe algorithm (Algorithm 9) can be equipped with any of the policies described in the previous section or the abstract policy π_A. We name the variations of the AlwaysSafe algorithm according to the policy used:

- AlwaysSafe-π_A refers to Algorithm 9 using LP4 to compute the occupancy of each state-action pair and Equation 5.4 to extract the policy
- AlwaysSafe-π_T refers to Algorithm 9 using LP4 to compute the occupancy of each state-action pair and Equation 5.4 or Equation 5.5 to extract the policy depending on the test in Equation 5.6.
- AlwaysSafe-π_α refers to Algorithm 9 using Algorithm 8 to compute the policy.

[3]Another option would be to change the test in Equation 5.6 allowing a small error ($\hat{c} + \epsilon$). This would give us an approximately safe algorithm.

Observe that varying the parameter $\underline{\beta}$ on Algorithm 8 can change the behavior of the AlwaysSafe-π_α algorithm. Setting $\underline{\beta} = 0$ give us the standard AlwaysSafe-π_α algorithm, while setting $\underline{\beta} = 1$ recovers the AlwaysSafe-π_A algorithm. For $\underline{\beta} \in (0,1)$, we expect that AlwaysSafe-π_α can still reach the optimal policy, however with a potentially smaller performance regret. Notice that none of these configurations should return an unsafe policy.

5.3.4. THEORY

We would like to show that when the AlwaysSafe algorithm is equipped with a cost-model-irrelevant abstract CMDP $\overline{\mathscr{M}}$ and one of the safe policies π_A, π_T, or π_α, it has no constraint violations with high probability, as stated in Theorem 7. In summary, the algorithm AlwaysSafe relies on the fact that the underlying CMDP $\mathscr{M}$ belongs to Σ with high probability, so it can test if the proposed ground policy is safe for all CMDPs in Σ, and when this cannot be guaranteed, it executes π_A which is guaranteed to be safe to collect more data.

Theorem 7. *Given an abstract CMDP built according to a cost-model irrelevance abstraction and a fixed $\delta \in (0,1)$, the algorithm AlwaysSafe equipped with policies π_A, π_T or π_α has no constraint violation regret with probability $1-\delta$.*

Proof. We split the proof in three parts related to the three policies considered.

AlwaysSafe-π_A. From Theorem 6 we know that

$$V_C^{\pi_A}(\mu) \leq \hat{c}.$$

This is enough to conclude that AlwaysSafe with π_A does not violate the safety constraints.

AlwaysSafe-π_T. First let us define the maximum expected cost of executing the policy π_G in a CMDP of the uncertainty set:

$$maxC = \max_{P' \in \Sigma} V_C^{\pi_G}(\mu, P').$$

From Equation 5.6 we must show that π_T is safe in both cases.

- **Case 1 ($maxC \leq \hat{c}$)**: in this case the policy executed is π_G. This way, we have that the expected cost for executing π_G in any of the CMDP of the uncertainty set is smaller than $maxC$:

  $$V_C^{\pi_G}(\mu, P') \leq maxC : \forall P' \in \Sigma.$$

  Therefore, if the true CMDP is in the uncertainty set, then the expected cost of the policy π_G is less or equal to the cost bound:

  $$P \in \Sigma \Longrightarrow V_C^{\pi_G}(\mu) \leq maxC \leq \hat{c}, \tag{5.7}$$

where the last inequality comes from the condition of this case. By construction, the transition function of the true CMDP belongs to the uncertainty set Σ with high probability $1-\delta$:

$$\Pr\left(P \in \Sigma\right) \geq 1-\delta. \tag{5.8}$$

Therefore, we have from Equation 5.7 and Equation 5.8 that:

$$\begin{aligned}\Pr\left(V_C^{\pi_G}(\mu) \leq \hat{c}\right) &\geq \Pr\left(P \in \Sigma\right)\\ &\geq 1-\delta.\end{aligned}$$

- **Case 2 ($maxC > \hat{c}$):** in this case the policy executed is π_A, which is safe (Theorem 6).

AlwaysSafe-π_α. This proof is similar to the proof for AlwaysSafe-π_T. In this case, the only difference is that π_G might be computed with a bound $\beta\hat{c}$ that is lower than the original bound $\hat{c}$.

□

Theorem 6 is enough to show that AlwaysSafe with π_A will not violate the constraints. By definition the transition of the true CMDP belongs to the uncertainty set Σ with probability $1-\delta$. Since the expected cost of the policies π_T and π_α is less or equal to $\hat{c}$ in all CMDPs in Σ, these policies are safe with probability $1-\delta$.

5.4. EMPIRICAL RESULTS

In this section, we empirically assess the proposed algorithms. We start by describing the experiments methodology, baseline algorithms, parameters, and the environments considered (Section 5.4.1). Next, we present a series of experiments focusing on different questions regarding the algorithms considered, including if they are able to ensure safety during the learning phase (Section 5.4.2), how the safety guarantees affect the performance of the algorithms (Section 5.4.3), which problems can lead to a sub-optimal policy (Section 5.4.4), and what is the impact of safety on the exploration capabilities (Section 5.4.5). The code to reproduce the experiments is available online[4].

5.4.1. SETUP

We evaluate the variations of the AlwaysSafe algorithm equipped with different policies from Section 5.3 (π_A, π_T and π_α with $\alpha = 0.5$), plus an instance using π_T with an adjusted cost bound $0.9\hat{c}$. We use the following algorithms as baselines:

- OptCMDP: Algorithm 6.
- AbsOptCMDP-π_G: Algorithm 7.

[4] `https://github.com/AlgTUDelft/AlwaysSafe`

- MLE CMDP with Bonus: a model-based reinforcement learning algorithm that uses `LP1` based on the maximum likelihood estimate of the CMDP and the optimistic estimate of the reward function as in OptCMDP. In particular, this method is computationally cheaper than OptCMDP since it does not handle the uncertainty of the transition function.

- Q-Learning: the classic off-policy model-free algorithm with a linearly decaying exploration rate (Watkins, 1989).

- Q-Learning Optimistic: Q-Learning with an optimistic initialization of the state-action values.

Although Q-Learning and Q-Learning Optimistic are not designed for CMDPs, they can give us some perspective with respect to the performance of the algorithms.

We use the following confidence intervals in the experiments:

$$\begin{aligned} e^P(s,a,s') &= \frac{1}{\max\{\eta(s,a),1\}} + \sqrt{\frac{\mathrm{Var}(\hat{P}(s' \mid s,a))}{\max\{\eta(s,a),1\}}}, \\ e^R(s,a) &= \frac{R^\top - R_\perp}{\max\{\eta(s,a),1\}}, \text{ and} \\ e^C(s,a) &= \frac{C^\top - C_\perp}{\max\{\eta(s,a),1\}}, \end{aligned}$$

where $\eta(s,a)$ is the number of times action $a \in \mathbb{A}$ has been executed in the state $s \in \mathbb{S}$, $R_\perp$ and $R^\top$ ($C_\perp$ and $C^\top$) are the minimum and maximum value of the reward (cost) function, and $\mathrm{Var}(x) = x * (1-x)$. We removed the subscript δ since these bounds are tighter than the theoretical bounds and do not depend on δ. Since the reward and cost functions in these environments are not in the interval $[0,1]$, we normalize the confidence intervals according to their spans. We also make them tighter to handle the large magnitude difference in the rewards of the cliff environment and taxi environment.

Finally, we follow a doubling epoch schedule (Jin et al., 2020a), where a new policy is computed only when one of the state-action counters doubles. To reduce the number of constraints and state variables in the linear programs we also assume the successors of each state is known:

$$\mathrm{suc}(s) = \left\{ s' \in \mathbb{S} \mid \exists a \in \mathbb{A} : P(s' \mid s,a) > 0 \right\}, \qquad \forall s \in \mathbb{S}.$$

Next, we describe the five environments used in the experiments to showcase different features of the AlwaysSafe algorithm.

SIMPLE CMDP

The **simple CMDP**, depicted in Figure 5.3 (left), was adapted from a problem proposed by Zheng and Ratliff (2020), it has 3 states ($\mathbb{S} = \mathbb{N}_3$) and 2 actions: *stay* in the current state, which does not incur any cost or reward; and *move* to the next state, which incurs a cost of 1 and a reward equals to the current state index. The agent has to balance between the actions move and stay to get the maximum reward without violating the

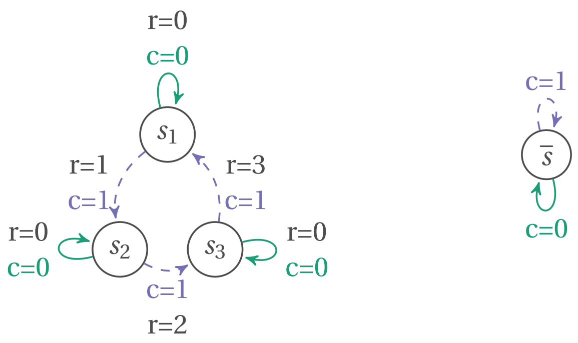


Figure 5.3: A CMDP with 3 states (left) and the corresponding abstract CMDP built with a cost-model-irrelevant abstraction (right).

safety constraints. We set $\hat{c} = 3$, $H = 6$, $K = 100$, and the abstraction ignores the state since the cost depends only on the action, see Figure 5.3 (right). In this way, the cost-model-irrelevance abstraction maps all states to a single state. Similar to Example 6, in this environment, the reward depends on the ground state, so the optimal policy cannot be computed in the abstract state space. Therefore, this environment serves to check whether the algorithms based on the safety abstraction can compute an optimal policy.

Factored CMDP

The **factored CMDP** from Figure 5.1 with $p = 0.9$. We set $\hat{c} = 0.1$, $H = 2$, $K = 5000$. We use the state abstraction that ignores the state variable y, see Figure 5.1 (right). This is a particularly challenging environment from a safety perspective because an optimistic algorithm may underestimate the value of p, as discussed in Example 5, leading to unsafe behaviors.

Cliff walking

The **cliff walking** problem is a 4×6 grid-world where the agent must get from a starting location (S) to a goal location (G) without falling off a cliff (Sutton and Barto, 2018, Example 6.6). We used the augmented version with a cost for walking close to the cliff (Lee et al., 2017), as shown in Figure 5.4 (left). The agent gets a reward of -1 for each movement. If the agent falls from the cliff (stepping in one of the grey areas), it is sent back to state S and gets a reward of -100. The agent gets a cost of 2 for walking in cells adjacent to the cliff (second row) and a cost of 1 for walking 2 cells away from the cliff (third row). We set $\hat{c} = 2$, $H = 15$, $K = 5000$, and we do not ignore any variables. In this example, the cost function depends on both variables, so we need to use an identity function to define the abstraction. We set $\hat{c} = 2$ and thereby ensure that an optimal policy is stochastic, as it needs to randomize between two longer paths. Figure 5.4 (right) shows the optimal paths (dashed lines) for a cost bound $\hat{c} = 2$. The agent needs to randomize between these two paths, taking each path 50% of the time, which gives an expected cost of 2 and an expected value of 10.

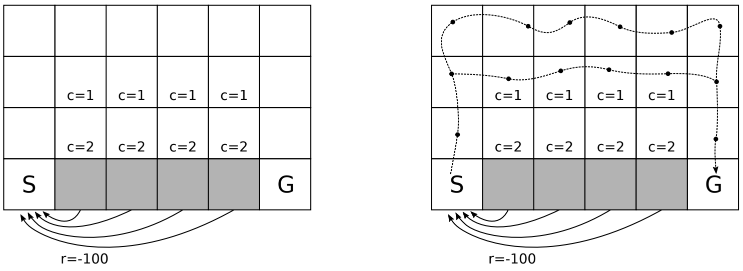


Figure 5.4: Cliff walking.

Figure 5.5: Taxi with fuel.

Taxi

The **taxi** with fuel problem (Dietterich, 1998) has 5 features describing a taxi's location, a passenger's location and destination, and the amount of fuel. It has 7 actions, including: move north, south, west, or east; pick up the passenger; drop off the passenger; and refuel. We considered a small version with a 2×2 grid and fuel capacity of 5, as shown in Figure 5.5 (left). We augment the problem by giving a cost signal of 1 if the taxi runs out of fuel and 0 otherwise. We set $\hat{c} = 0$, $H = 6$, $K = 5000$, and ignore the passenger's location and destination. Setting $\hat{c} = 0$ ensures the taxi never runs out of fuel.

Cost Chain

The **cost chain** environment is inspired by the chain environment (Osband et al., 2016), commonly used to evaluate the exploration capability of RL algorithms, where the agent can collect a small reward in the first state and a large reward in the last state. The reward and cost structure of this environment is also similar to the factored CMDP. It has 2 features (x and y) where $x \in [1, n]$ and $y \in [1, 2]$. The agent has three actions: a, b, and reset.

In this environment, when the agent takes action A in the state (x, y), we observe the following dynamics. If $x = n - 1$, the agent gets a reward of 1 and moves to the absorbing state (n, y). If $A = reset$, the agent moves to state $(0, y)$, with no cost and a reward of $\frac{1}{n}$. For states where x is even and $A \in \{a, b\}$ the agent receives no reward or cost and transitions to state $(x+1, 0)$ with probability p and to state $(x+1, 1)$ with probability $1-p$. Finally, for states where the x is odd, the agent moves to state $(x+1, y)$, receives a cost of 1 if $A = a$ and 0 otherwise, and receives a reward equals to $\frac{1}{2n}$ if $y = 0$ and $A = a$, or if $y = 1$ and $A = b$, otherwise it gets no reward. We set $n = 20$, $\hat{c} = 0.5$, $H = 21$, $K = 10000$, similar to the factored CMDP, a cost-model-irrelevant abstraction ignores the feature y.

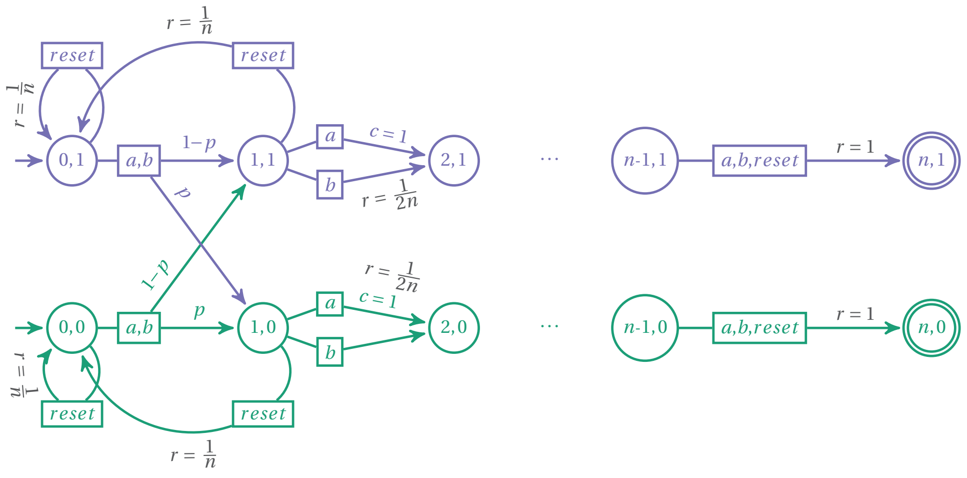


Figure 5.6: The cost chain environment CMDP. Costs and rewards with value 0 are omitted as well as the probability of deterministic transitions.

5.4.2. Safety evaluation

The first experiment aims to evaluate the safety aspects of the algorithms. Figure 5.7 shows the expected cost (left column) and expected value (right column) of the policy in each episode on the environments simple CMDP, factored CMDP, and cliff walking. These environments are used because they require the agent to sacrifice some return to meet the safety requirements.

We start the analysis with the simple CMDP (top row). We can observe that the algorithms OptCMDP, MLE CMDP with Bonus, and Q-Learning obtain policies with an expected value larger than the optimal constrained policy (top left). However, to do so, they have to violate the cost constraints (top right). Although the algorithm AbsOptCMDP π_G has no safety guarantees, in this domain, it converges to the optimal policy without violating the constraints.

As expected, all instances of the AlwaysSafe algorithm respect the cost constraint. However, only π_α converges to the optimal policy, while the others converged to a sub-optimal policy, indicating that the ground policy did not pass the safety test and the abstract policy was used. We notice that in the first episodes, even though the confidence interval was loose, the final $\hat{c}$ was low enough to make the ground policy safe in the whole uncertainty set. Analyzing the expected value AlwaysSafe π_α uses a conservative policy. In Section 5.4.3, we investigate how to make this algorithm less conservative.

The experiments with the factored CMDP (middle row) show that AbsOptCMDP π_G is not safe. Only AlwaysSafe-π_α safely reaches the optimal performance. We can also see that AlwaysSafe π_T with $0.9\hat{c}$ changes from policy π_A to policy π_G after ~ 500 episodes but still does not reach the optimal performance, while the algorithm AlwaysSafe-π_T always executes π_A. This issue is investigated further in Section 5.4.4.

We conclude this analysis with the cliff environment (bottom row). For the cliff environment, we only consider the algorithm AlwaysSafe-π_A since $\mathbb{S} = \overline{\mathbb{S}}$ which implies

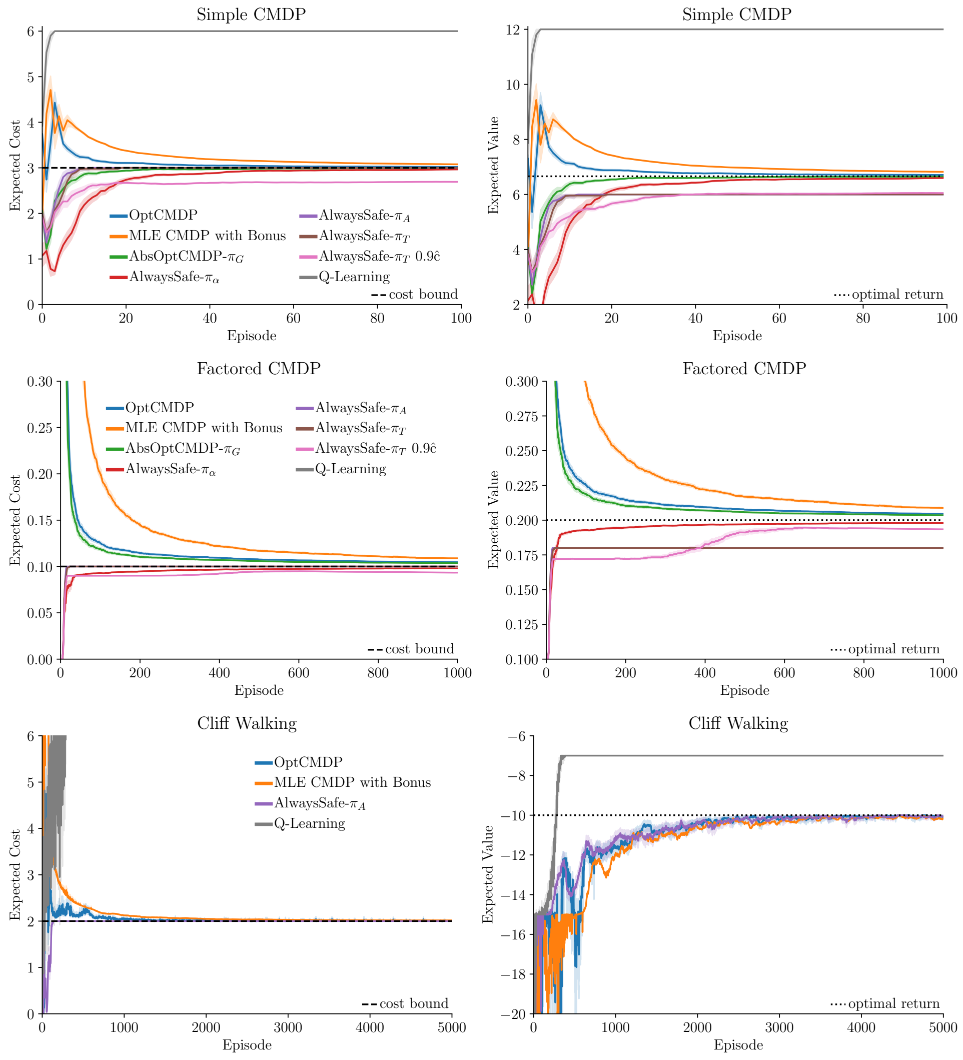


Figure 5.7: Results for the simple CMDP (top), factored CMDP (middle), and cliff environment (bottom), all averaged over 100 runs with a 95% confidence interval. The left column shows the expected cost of the policy executed in each episode while the right column shows the expected value of the policy. A dashed line depicts the bound on the expected cost $\hat{c}$ and a dotted line depicts the optimal expected value.

$\pi_G = \pi_A$, making all the algorithms equipped with the abstract safety dynamics virtually the same. We observe, to no surprise, that the AlwaysSafe-π_A algorithm is able to always execute policies with an expected cost lower than the given bound. The OptCMDP and MLE CMDP with Bonus algorithms, on the other hand, violate the cost constraints for hundreds of episodes. Analyzing the expected value of the policies executed (bottom right), we have an indication that, in the cliff environment, the constraints on the expected cost are actually beneficial for the AlwaysSafe-π_A algorithm that accumulates a smaller regret in terms of performance as well. Section 5.4.4 provides more intuition on this aspect.

5.4.3. Dynamic constraint tightening evaluation

Analyzing the expected value on the simple CMDP at Figure 5.7 (top right), we may notice that in the early episodes, the algorithm AlwaysSafe π_A shows a higher expected value than the algorithm AlwaysSafe-π_α. This raises the question if it is possible to reduce the performance regret of the AlwaysSafe-π_α algorithm by executing the abstract policy in the early episodes. In other words, can we avoid policies too conservative in the early episodes, when the agent only finds a ground policy that satisfies the safety constraints using a very low β. To examine this hypothesis, we evaluate the AlwaysSafe-π_α algorithm varying the coefficient lower bound $\underline{\beta}$.

We use the same configurations of the simple CMDP described before, but we let $K = 300$ to observe the performance of the variants that take more time to converge. We consider $\underline{\beta} \in \{0, 0.1, \cdots, 1\} \cup \{0.91, 0.92 \cdots, 0.99\}$.

Figure 5.8 (top) shows the value of the policies executed in the first 300 episodes. On the left, we notice that, as expected, all variations have an expected cost lower than the given safety threshold, demonstrating that the value of $\underline{\beta}$ does not affect the safety of the algorithm. On the right, we observe significant differences in terms of expected value with respect to $\underline{\beta}$. In particular, when $\underline{\beta}$ is closer to 0, the agent might deploy extremely conservative policies in the early episodes. We also notice that when $\underline{\beta}$ is close to 1, the performance is closer to the abstract policy (when $\underline{\beta} = 1$) during more episodes, and the agent needs significantly more time to find the optimal policy. For instance, for $\underline{\beta} = 0.99$ the agent only starts approaching the optimal performance after 100 episodes.

On Figure 5.8 (bottom), we see the accumulative performance regret over the 300 episodes on the left and after 300 episodes on the right. We can observe that, in the early episodes, the agent with high values for $\underline{\beta}$ accumulate less regret than the agents with low $\underline{\beta}$. However, over time we notice that agents with $\underline{\beta}$ too high can accumulate more regret than the original algorithm ($\underline{\beta} = 0$), in particular when $\underline{\beta} = 0.99$. Therefore, setting $\underline{\beta}$ too high can make the agent use the abstract policy for too many episodes, which might prevent the exploration of specific ground states leading to a larger performance regret.

Figure 5.8 (bottom right) shows that the values that accumulate less regret are close to $\underline{\beta} = 0.9$, indicating that a balanced $\underline{\beta}$ is ideal. It also shows that the algorithm is robust to the choice of $\underline{\beta}$, obtaining almost the smallest regret with values between 0.7 and 0.9.

Figure 5.9 shows the results of the same experiment on the factored CMDP environment. The plots follows the same structure from Figure 5.8. We notice that, in this example, the low values of $\underline{\beta}$ perform better than values closer to 1. This can be explained by the lower difference in performance between the ground policy and abstract policy, as

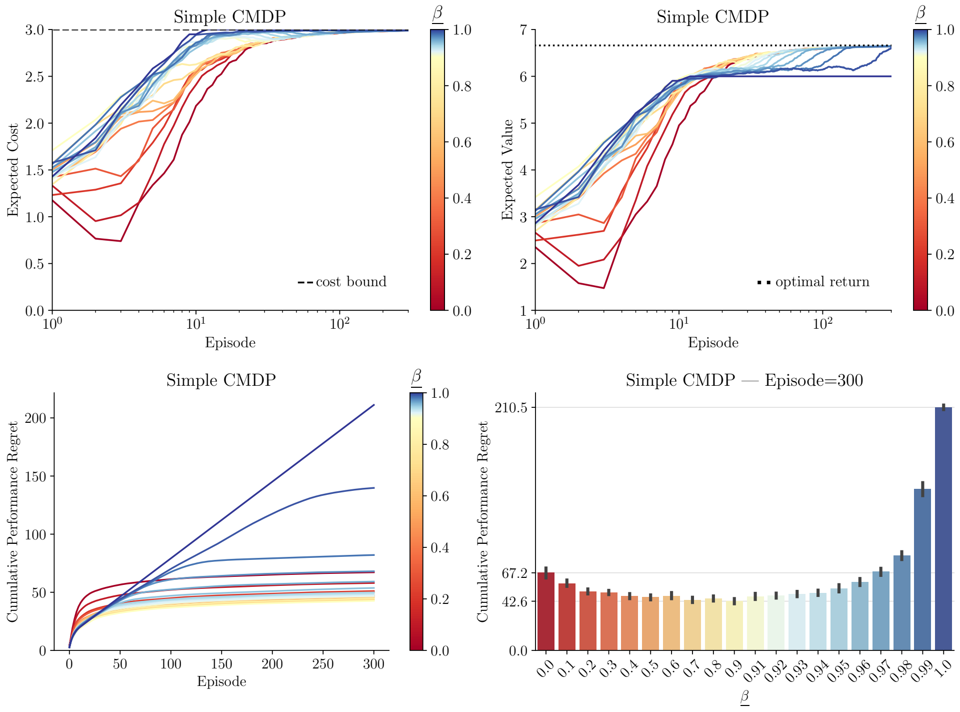


Figure 5.8: AlwaysSafe-π_α on the simple CMDP environment varying the coefficient lower bound $\underline{\beta}$, averaged over 100 runs: (top left) the episodic expected cost, (top right) the episodic expected value, (bottom left) the cumulative performance regret over episodes and (bottom right) the accumulated performance regret after 300 episodes (with a 95% confidence interval). A dashed line depicts the bound on the expected cost $\hat{c}$ and a dotted line depicts the optimal expected value.

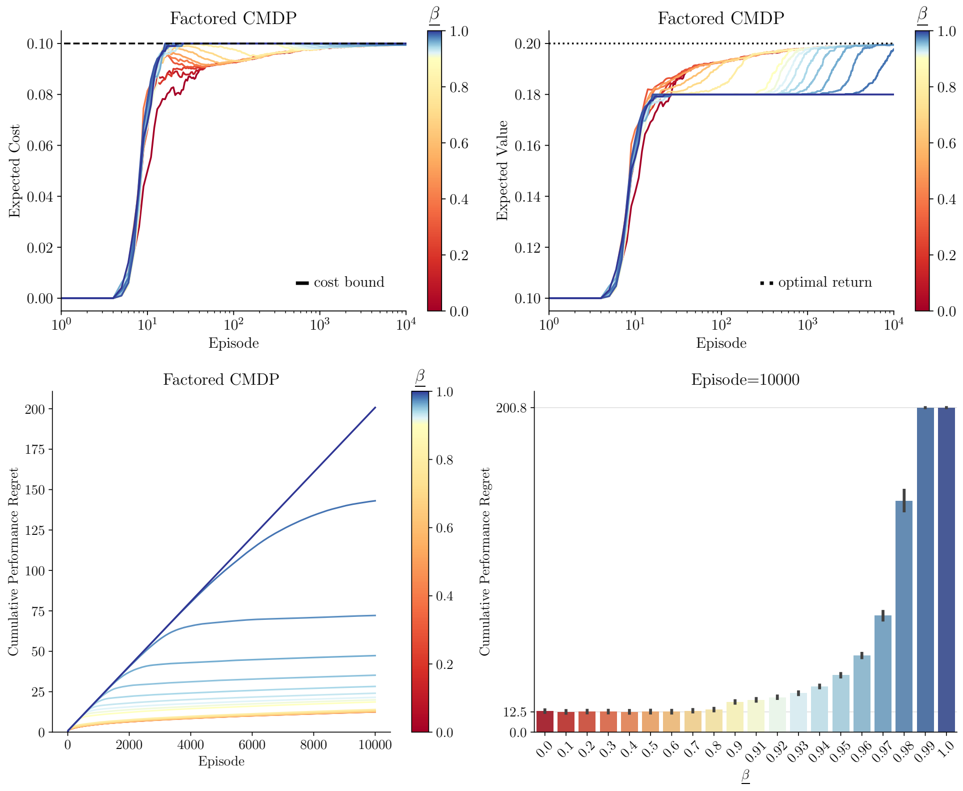


Figure 5.9: AlwaysSafe-π_α on the factored CMDP environment varying the coefficient lower bound $\underline{\beta}$ over 10000 episodes, averaged over 100 runs: (top left) the episodic expected cost, (top right) the episodic expected value, (bottom left) the cumulative performance regret and (bottom right) the accumulated performance regret (with a 95% confidence interval). A dashed line depicts the bound on the expected cost $\hat{c}$ and a dotted line depicts the optimal expected value.

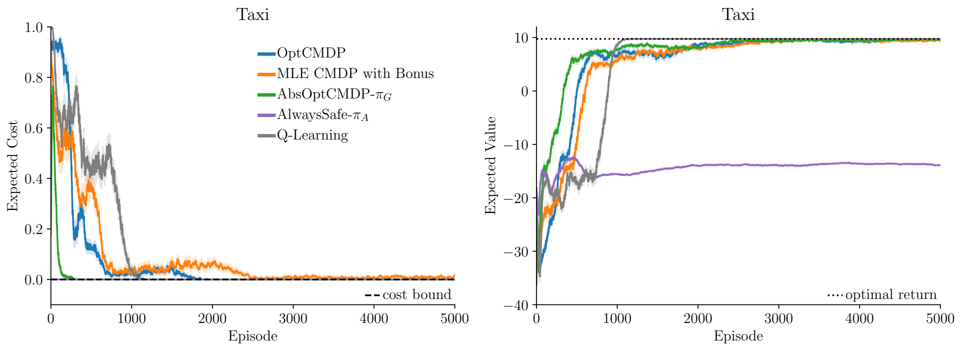


Figure 5.10: Results for the taxi with fuel environment averaged over 100 runs with a 95% confidence interval. The left column shows the expected cost of the policy executed in each episode while the right column shows the expected value of the policy. A dashed line depicts the bound on the expected cost $\hat{c}$ and a dotted line depicts the optimal expected value.

observed on Figure 5.7 (middle right). Nevertheless, the range of $\underline{\beta}$ that has better performance is also wide, varying from 0 and 0.8, which indicates again that the algorithm is robust to the choice of $\underline{\beta}$.

Overall, finding a principled way to decide whether or not to stop the dynamic constraint tightening and return a conservative policy is still an open question.

5.4.4. Tight safety bounds

Figure 5.10 shows the results on the taxi environment. In this environment, the cost bound is set to 0. Therefore, the dynamic constraint tightening strategy cannot reduce the cost bound $\hat{c}$ without making the problem infeasible. Therefore, we only consider the algorithms AlwaysSafe-π_A and AbsOptCMDP-π_G.

Similar to the results on simple CMDP and factored CMDP, the AlwaysSafe algorithm equipped with the abstract policy π_A is able to respect the safety constraints but is not able to find the optimal policy. The remaining algorithms tested are able to solve the task, however, they violate the cost constraints in early episodes. In particular, we notice that AlwaysSafe-π_G can learn how to remain safe faster than the remaining algorithms; consequently, AlwaysSafe-π_G also finds policies with high performance faster. This shows that when the safety constraints are aligned with the reward function, the performance of safer agents also improves.

5.4.5. Exploration efficiency

In this last experiment, we consider how the safe behavior of the agent impacts its exploration efficiency, using the cost chain environment with horizon 20. Figure 5.11 shows the results.

First, we notice that both Q-Learning and Optimistic Q-Learning are not able to find the optimal policy for this environment, which shows that this task requires a reasonable exploration strategy. Otherwise, the remaining algorithms have similar behavior as in the factored CMDP environment, where OptCMDP and MLE CMDP with Bonus can

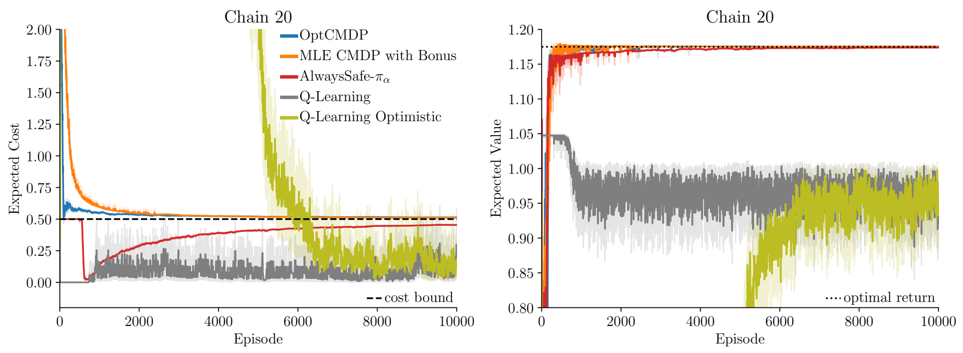


Figure 5.11: Results for the cost chain environment averaged over 100 runs with a 95% confidence interval. The left column shows the expected cost of the policy executed in each episode while the right column shows the expected value of the policy. A dashed line depicts the bound on the expected cost $\hat{c}$ and a dotted line depicts the optimal expected value.

find the optimal policy but violate the cost constraints in the early episodes, while the AlwaysSafe-π_α can find the optimal policy without violating the safety constraints. This lets us conclude that AlwaysSafe-π_α is still able to explore the environment optimistically with respect to the reward.

5.4.6. Results discussion

Following Ray et al. (2019), we may conclude that the algorithm AlwaysSafe equipped with the safe policies π_A, π_T, π_α or π_T with $0.9\hat{c}$ dominates the algorithms OptCMDP and AbsOptCMDP π_G since the former does not violate the constraints, while the latter does. Then, AlwaysSafe-π_α dominates AlwaysSafe-π_A, AlwaysSafe-π_T and AlwaysSafe-π_T with $0.9\hat{c}$ since in general it achieves higher performance. Nevertheless, these results come with the requirement that the abstract model relevant for safety is known. We believe that in cases where this model is only partially known, these algorithms would still be useful to make the most of the knowledge available.

In general, we notice that, on the one hand, the optimistic algorithms OptCMDP and AbsOptCMDP π_G eventually approach the safety bound, but might use unsafe policies during this process, which is a clear consequence of their optimism with respect to the transition and cost function. On the other hand, the AlwaysSafe algorithm equipped with safe policies sacrifices performance to ensure safety. However, when the cost function is well aligned with the reward function, it can also have a smaller performance regret.

5.5. Related Work

As we mentioned in Chapter 1, there are two popular directions in safe reinforcement learning (García and Fernández, 2015): (i) shaping the optimization criterion towards risk sensitivity (Chow et al., 2018a) and (ii) changing the exploration process by, for instance, assuming the existence of a safe policy (Zheng and Ratliff, 2020). The method proposed in this chapter is at the intersection of these directions since we employ external knowledge to modify the exploration process using a different optimization criterion.

Zheng and Ratliff (2020) adapt the UCRL2 algorithm to CMDPs and assume that the full transition model of the MDP is known and the agent has access to a safe (baseline) policy. In contrast, we only require an abstraction of the transition model that is relevant for the cost function. Recently, HasanzadeZonuzy et al. (2021) showed that the sample complexity of learning in CMDPs increases only logarithmically in comparison to unconstrained problems. However, their probably-approximately-correct (PAC) scheme does not provide any safety guarantees during the learning phase.

Furthermore, RL algorithms that provide guarantees of not violating the constraints during the learning phase include methods that model the environment dynamics using Gaussian processes (Berkenkamp et al., 2017; Turchetta et al., 2019; Wachi and Sui, 2020; Wachi et al., 2018), design Lyapunov functions to guarantee the global constraints (Chow et al., 2018b) or use analogies (Roderick et al., 2021). In general, these methods assume an initial safe policy to begin exploring, allowing the agents to slowly expand the set of known safe policies/states.

For problems with high-dimensional input spaces, different policy search algorithms have been proposed that provide certain (though not hard) guarantees of not violating the constraints (Achiam et al., 2017; Tessler et al., 2019; Yang et al., 2020b) or find safe policies only at the end of the training process (Ray et al., 2019). In this setting, the safety constraint has also been generalized to consider the tail of the distribution of accumulated expected costs instead of the mean (Yang et al., 2021).

FMDPs have been explored in different RL settings, developing algorithms with near-optimal regret bounds in FMDPs without constraints (Osband and Van Roy, 2014) and with constraints (Chen et al., 2021b), improving sample efficiency (Chakraborty and Stone, 2011; Diuk et al., 2009; Strehl et al., 2007), and tackling the off-policy policy evaluation problem (Hallak et al., 2015), In the safety literature, FMDPs have been used to reduce the sample complexity of batch RL algorithms with safety guarantees with respect to the performance of a baseline policy (see Chapter 3) and to allow an agent to query a supervisor about the features of the FMDP to avoid side effects (Zhang et al., 2018).

Reachability constraints enforce policies to avoid catastrophic states. Fatemi et al. (2019) avoid such states with high probability, and Taleghan and Dietterich (2018) look into deterministic policies that are easier to perceive than the usual randomized policies. Similarly, works from the formal methods community use reachability constraints and their extension, temporal logic constraints, to argue about safety during exploration using prior knowledge about the transition model (Alshiekh et al., 2018; Hasanbeig et al., 2020; Jansen et al., 2020; Junges et al., 2016). Finally, a control-theoretic simplex architecture has been employed to switch between safe and high-performance controllers (Phan et al., 2020).

5.6. Conclusions

This chapter investigates settings where safety-relevant dynamics are given. We proposed the AlwaysSafe algorithm that can be optimistic with respect to the reward while ensuring safety during exploration.

In particular, we used an abstract model of the safety-relevant dynamics to compute an abstract policy that is always safe and a ground policy that can achieve high performance. We showed how to switch between these two policies to build an algorithm that is safe and eventually converges to the optimal policy. This method not only enforces the agent to always act safely but can also prune under-performing actions, improving the training efficiency when the cost function is aligned with the reward function.

Further studies might investigate alternative methods to devise the abstractions of the safety dynamics, for instance, using core states (Shariff and Szepesvári, 2020); how the AlwaysSafe copes with an approximation of the abstract CMDP (Abel et al., 2016); and algorithms that can aggregate states online (Ortner, 2013) without violating the constraints.

In summary, the proposed algorithm is always safe during the learning process, eventually reaches the optimal policy, and decouples exploration from safety issues in RL.

6
Concluding Remarks

Before closing this thesis, we review its main contributions, reflect on the results obtained and consider possible directions for further research.

6.1. Contributions

As mentioned in Chapter 1, there are multiple ways to make RL algorithms more reliable. We considered an offline setting, where the goal is to provide guarantees on the performance of the policy computed. In particular, in Chapter 3 we explore settings where the behavior policy is known, while in Chapter 4 we explore settings where only the set of trajectories is available. In Chapter 5 we took a step towards safe exploration, considering situations where prior knowledge about the safety dynamics of the environment is available, but not about the entire task of the agent.

Overall, RL only assumes the environment is an MDP, which is a weaker assumption than the assumption in probabilistic planning that the full model of the environment is provided. However, to provide safety guarantees, we must make stronger assumptions, putting SRL closer to probabilistic planning. Keeping this in mind, we may analyze the contributions in this thesis according to how we change the strength of such assumptions. On the one hand, we investigated settings with stronger assumptions to make problems more tractable. On the other hand, we investigated problems with weaker assumptions to relax their constraints and make the solution more general. We analyze our contributions on this spectrum.

6.1.1. Increasing Tractability

Addressing Research Question 1, we observed multiple benefits of narrowing the scope of the problem by relying on some domain knowledge. Focusing on FMDPs allowed us to reduce the sample complexity of offline RL algorithms, in particular of SPI algorithms, and to provide some generalization capabilities to these algorithms (Chapter 3). Following a similar strategy, we also developed mechanisms that allow safe exploration of the environment using a description of the safety dynamics (Chapter 5).

Considering FMDPs, we managed to reduce significantly the sample complexity of SPI algorithms, which addresses Research Question 2. We investigated two classes that make different assumptions regarding prior knowledge about the problem. In the first, the structure of the problem is provided, so the agent only needs to estimate the parameters of the local transition components representing the DBN. In the second, only a bound on the number of parents each state variable is given, so the agent needs to estimate the structure of the problem. When such assumptions are met, these algorithms can compute policies with higher performance than algorithms that handle unrestricted MDPs when given the same dataset.

A compelling benefit of focusing on a restricted class of problems, such as FMDPs, is that the resulting SPI algorithms gain some generalization capabilities, which addresses our Research Question 3. These algorithms can make inferences about unvisited states by combining samples from similar states. This can be particularly helpful when the data available have only partial coverage of the problem. While SPI algorithms designed for unrestricted classes of MDPs become very conservative and always rely on the behavior policy, we showed that by exploiting the factored structure of the problem, the agent might infer the dynamics of states that are not present in the dataset allowing it to choose actions that had not been executed during the data collection phase.

Incorporating domain knowledge also increases transparency and hence the user's trust in the system. To address Research Question 5, we considered safe exploration in CMDPs, where the user provides the safety dynamics of the environment (Chapter 5). Besides allowing the agent to learn to optimize its main task without violating the safety constraints, this provides assurance to the user that the agent will not spend resources to learn about information already known. Ultimately, this approach also showed a regret reduction in terms of performance when the safety is aligned which the agent's task, addressing Research Question 6.

Constraints are another way of exploiting an expert's prior knowledge. It allows practitioners to directly specify the behaviors expected from the agent, avoiding the issues with reward engineering (Roy et al., 2021). This increases the user's autonomy, allowing it to prevent some behaviors of the RL agent. We studied constraints on the total expected cost; nevertheless, other types of constraints could also be considered, as discussed later.

As a final note about narrowing the class of problems, we would like to point out that investigating other types of structure may allow us to incorporate more domain knowledge in the agent's learning process. For example, the exploitation of Gaussian (Turchetta et al., 2016) and linear (Jin et al., 2020c) dynamics have a large potential to expand the reach of online and offline SRL agents.

6.1.2. STRIVING FOR SIMPLICITY

Moving towards the opposite direction from the previous section, we investigated how to relax some of the SPI algorithms' assumptions to make them more general.

In Chapter 3, we first proposed a factored SPI algorithm that assumes the structure of the problem is known, meaning we know the dependencies between the features of the problem. If that is not known beforehand, we show that it is also possible to learn the structure of an FMDP, in which case we make the weaker assumption that the user provides a bound on the number of state variables that can influence another state variable.

Addressing Research Question 4, we showed in Chapter 4 that offline RL algorithms can have improvement guarantees even when the behavior policy is unknown. This revealed the possibility of developing reliable offline RL algorithms with minimum requirements, which is essential to make these algorithms more accessible. Previously, if we had collected some data with a particular policy but were unable to reconstruct this policy, we would not have any guarantees regarding the performance of the policy returned by an offline RL algorithm. The approach we proposed ensures that, even in this situation, the practitioner could still recover the behavior policy and use SPI algorithms with improvement guarantees.

6.1.3. On the Gap between the Lab and the Real World

Our motivation has been to increase the reliability of RL agents, which should facilitate their deployment in real world scenarios. Unfortunately, we did not perform experiments directly on real world applications since we may require an excessive number of trials to have statistically significant results for safe policy improvement methods. Nevertheless, some applications have high fidelity simulators that can be used to test these methods. We mentioned an example of such a case study in Section 3.5, where SPI algorithms were tested on a steel melting plant simulation (Kosiorek, 2020).

This case study also showed the benefits of incorporating safety constraints explicitly. In the steel plant, some actions certainly have bad outcomes, such as launching two ladles from the same group simultaneously, since they might arrive at the caster at similar times, causing unnecessary delays. Preventing the agent from using such actions made the algorithm's final policies more reliable overall. This is an example of the need for means to define the safety constraints explicitly, which we studied in Chapter 5.

6.2. Reflections

While safety and reliability were the main focus of this thesis, we also touched on other issues preventing the deployment of reinforcement learning to real world tasks, such as data efficiency and learning with a fixed batch of historical data. Nevertheless, we must recall that multiple other challenges still require attention, such as partial observability, explainability and others (Figure 1.1).

In the offline setting, we evaluated different ways to improve sample efficiency. This is important considering the amount of data available in many applications is limited. We developed algorithms that, by exploiting the factored structure of the problem, require fewer data to compute improved policies. This approach also provides some generalization capabilities, which helps offline agents handle cases where the data only provides partial support over the state-action space. Finally, we investigated the setting where the behavior policy on which these algorithms rely is not available. We found that using an estimate of the behavior policy can be a reasonable solution.

In the online setting, we considered how to prevent constraint violations during the learning process. We investigated settings where the user provides prior knowledge of the safety dynamics. While we focused on constraining the expected safety cost, we believe similar approaches can be developed for problems with hard constraints or with chance constraints.

6.3. DIRECTIONS FOR FUTURE WORK

In this section, we discuss potential future work. We start by revisiting our two core problems (Figure 1.1), as well as their interplay. We also present some alternative ways to provide safety to reinforcement learning agents and reflections about the future of RL.

6.3.1. REVISITING THE OFFLINE REINFORCEMENT LEARNING

Since the beginning of this work, offline RL has gained considerable attention due to its use of historical data in the hope of following the trends of supervised learning (Levine et al., 2020). Nevertheless, there are still numerous open challenges that require further research.

Closer to this thesis, we may mention that there are still multiple ways of exploiting the factored structure of the problem. First, we observe that we could also easily exploit the factored structure for the soft SPIBB algorithm (Nadjahi et al., 2019), which was only used in Chapter 4. Another intriguing question is whether we can exploit the factored structure of the problem when the baseline policy is unknown. This is challenging since the behavior policy might not have a factored structure, even if used in an FMDP. Nevertheless, if we can assume that the behavior policy has some structure, for instance, it is defined by a decision tree with a limited depth or a limited number of nodes, we could also estimate the policy efficiently.

We may also exploit other types of structures to reduce the problem class. Some examples include: ignoring exogenous variables (Dietterich et al., 2018), using influence abstraction to reduce the size of the problem (Oliehoek et al., 2012), or separating private and environmental variables (Liu et al., 2021a). These approaches have the potential to present benefits similar to exploiting the factored structure, meaning they might improve the data efficiency of the algorithms and may help with generalization.

An important question to be explored in the offline setting is the source of the historical data. In some cases where the system designer envisions the future use of offline RL algorithms, collecting the data with good coverage of the problem can be critical to find high-performing policies later on. In this case, computing policies to effectively explore the environment would be ideal (Jin et al., 2020b; Wang et al., 2020).

Although the pure offline RL setting has received close attention over the last years, recent results show that this problem might be considerably more complex than its online counterpart (Xiao et al., 2021a; Zanette, 2021). This indicates that using a single batch of past trajectories has limited potential. Therefore, the development of intermediate algorithms that can interleave between data collection and policy optimization is a promising direction for future research, following the growing batch methodology (Lange et al., 2012; Riedmiller et al., 2021).

Revisiting the setting where a baseline policy is available but the agent can still interact with the environment, the reliability question would be how to minimize the regret with respect to the baseline policy (Dey et al., 2021; Pirotta et al., 2013). An interesting compromise would be to compute a policy with maximum entropy while ensuring a minimum performance (Savas et al., 2018). This would create a new trade-off between reliability and exploration. One potential solution for this problem is to use a set of policies that allows the agent to collect diverse trajectories during exploration (Ghasemi et al., 2021; Kumar et al., 2020). In this case, constraining each policy to guarantee a minimum performance could make the data collection more reliable.

6.3.2. NEW CONSTRAINTS FOR REINFORCEMENT LEARNING

To take safety into account more explicitly, we considered the constraints with respect to the expected accumulated safety cost, in other words, a soft budget constraint. However, constrained reinforcement learning encompasses an extensive set of problems stemming from its diverse types of constraints (Liu et al., 2021b). Therefore, finding ways to learn while satisfying other types of constraints would expand the reach of safe RL. For instance, considering a similar setting with instantaneous hard constraints, when the safety cost in each time step is bounded .

Section 2.2 mentioned the descriptive power of planning tools that allow experts to specify their problems, such as using accessible and intuitive languages (Sanner, 2010). We may use it to define the abstraction of the safety dynamics as a partial description of the problem, where we only have access to a subset of the CPTs of the DBN. In this scenario, it would also be interesting to learn the remaining relational description of the problem (Ng and Petrick, 2019).

In Chapter 5, we proposed an RL method that computes a policy in the dual space using a linear program. While solving a linear program is computationally efficient, the state space might be too large even to instantiate the linear program, which would render the proposed algorithm ineffective. Nevertheless, in the literature, we can find multiple strategies that could improve the scalability of this method: exploring the structure of the problem further using an approximate linear programming approach can reduce the size of LP (Lee et al., 2017; Poupart et al., 2015), restricting the computational effort to more relevant parts of the environment using online algorithms (Brázdil et al., 2020; Rostov and Kaisers, 2021), or generating constraint or variables incrementally to avoid having to instantiate the whole problem at once (Hansen and Bowman, 2020; Walraven and Spaan, 2018).

We can also consider other types of constraints. When some demonstrations from a teacher are available, it could be interesting to investigate how to ensure the new policy induces a state occupancy close to the demonstrations while guaranteeing a minimum performance (Wang et al., 2021). Instead of bounding the expected cost, one might prefer to bound the probability of reaching an accumulated cost higher than the given budget in a trajectory (Moreira et al., 2021), or, taking a more general perspective, bounding the visitation density of specific states (Qin et al., 2021). Finally, integrating constraints in the offline RL setting could be attractive since it considers constraints more explicitly while allowing us to use offline data (Le et al., 2019). A natural direction for this idea would be the constrained safe policy improvement problem, which is already getting some attention (Satija et al., 2021).

6.3.3. ALTERNATIVE SOURCES OF SAFETY GUARANTEES

In this thesis, we often considered some restricted class of problems which allowed us to make strong statements regarding the safety of the proposed algorithms. Naturally, if these assumptions are not met, such statements might become invalid. For situations where one cannot make such assumptions, there are other ways to deal with the safety issues in RL. We will discuss two of them next.

Teacher Intervention. One option to ensure safety is to have a teacher responsible for overseeing the actions chosen by the agent. This teacher could be a piece of software (Jansen et al., 2020) or even a human (Saunders et al., 2018). The goal of the teacher might be to provide feedback for the behavior of the agent (Thomaz and Breazeal, 2008; Zhang et al., 2019), demonstrate the expected behavior in the form of trajectories (Osa et al., 2018), or make corrections on the actions chosen by the agent (Dalal et al., 2018). On the one hand, teacher interventions can also speed up the training process (Celemin et al., 2019; Silva et al., 2020). On the one hand, they also raises new challenges, such as how to choose what the agent should learn first (Turchetta et al., 2020), how to communicate the user's preferences to the agent (Aytemiz et al., 2021), and how to minimize the workload of the teacher (Andrés et al., 2018; Ha et al., 2020; Prakash et al., 2019).

Initial Safe Policy. Another option is to consider a potentially sub-optimal safe policy is available, similar to how we can exploit the baseline policy on the offline setting (see Chapters 3 and 4). In the online setting, the agent can collect experiences using this policy until it builds enough confidence to execute a different policy. A considerable body of literature on safe exploration assumes such policy is available (Berkenkamp et al., 2017; Chow et al., 2018b; Luo and Ma, 2021; Zheng and Ratliff, 2020). However, the origin of this policy is not always clear. Considering so many methods need such safe policy to start exploring safely, it would be interesting to investigate how to compute a policy effective for such tasks. An option is to train the RL agent in a controlled environment, for instance, in a laboratory, to later transfer the policy obtained to the real world. In this controlled environment, the agent may violate the safety constraints since it is possible to reset the episode at any time.

6.4. FINAL REMARKS

As a final note, we discussed briefly in Chapter 2 how the trade-off between exploration and exploitation in RL has been studied extensively. However, with the quick dissemination of RL, new issues and requirements emerge, such as safety, fairness, privacy, transparency, security, and others. While this thesis focused on increasing the reliability of RL algorithms, overall considering safety as a primary issue, the question of how to trade-off between these new requirements is still essentially an open question (Pineau, 2021). Such multiplicity of objectives indicates the importance of the multi-objective setting given the challenges of compiling them in a single reward function.

To conclude, we expect that the deployment of RL agents will require an interdisciplinary approach, combining the domain knowledge from experts and RL researchers, that together can investigate what assumptions are reasonable for the problem at hand.

A

Acronyms

AI	Artificial Intelligence.
CMDP	Constrained Markov Decision Process.
CPT	Conditional Probability Table.
CRL	Constrained Reinforcement Learning.
DBN	Dynamic Bayesian Network.
FMDP	Factored Markov Decision Process.
HCOPE	High Confidence Off-Policy Evaluation.
HCPI	High Confidence Policy Improvement.
KWIK	Knows What It Knows.
LP	Linear Program.
MDP	Markov Decision Process.
MLE	Maximum Likelihood Estimate.
RaMDP	Reward-adjusted MDP.
RL	Reinforcement Learning.
Soft-SPIBB	Safe Policy Improvement with Soft Baseline Bootstrapping.
SPI	Safe Policy Improvement.
SPIBB	Safe Policy Improvement with Baseline Bootstrapping.
SPM	Steel Plant Model.
SRL	Safe Reinforcement Learning.

B

NOTATION

Π	Set of policies.
Σ	The set of probable MDPs given the past trajectories.
δ	A parameter related to the *confidence*.
ϵ	A parameter related to the *precision*.
η	Visitation counter.
γ	Discount factor.
μ	Initial state distribution over states.
$\oslash$	Symbol used by a KWIK algorithm to indicate "I do not know".
π	Policy.
π_b	Behavior policy.
σ	Hyper-parameter of the Soft SPIBB algorithm.
ζ	An admissible error for SPI algorithms.
$\mathbb{1}$	Indicator function.
$\mathbb{A}$	Set of actions.
Anc	Ancestors: the state variables with eventual influence on some state variable(s).
$\mathbb{B}$	Set of states-action pairs *bootstrapped* by an SPIBB algorithm.
C	$\mathrm{C}_k^{\mathbb{G}}$: the set of all *combinations* of size k of a set $\mathbb{G}$.
C	Cost function, indicates the cost to take an action on a given state.
$\hat{c}$	Upper bound on the expected accumulated cost.
$\mathscr{D}$	Collection of past interactions with the environment.
d	Maximum in-degree.
dom	Domain of a state variable, set of values a variable can assume.

H	Horizon.
K	Number of episodes.
$\mathscr{K}$	A KWIK algorithm.
$\mathbb{K}$	Set of states-action pairs considered *known* by a KWIK algorithm.
$\mathscr{M}$	A specific MDP.
m	Hyper-parameter related to number of samples required by the respective learning algorithm..
$\mathscr{P}$	Probability simplex: set of probability distributions over a finite set.
P	Transition function, describes how the environment changes.
Pa	Parents: the state variables with immediate influence on some state variable(s).
$\mathscr{Q}$	Domain of transition components on a factored MDP.
R	Reward function, indicates how good it is to take an action on a given state.
$\mathbb{S}$	Set of states.
$\mathbb{X}$	Set of state variables that describe the state of the environment.

Bibliography

David Abel, D Ellis Hershkowitz, and Michael L Littman. Near Optimal Behavior via Approximate State Abstraction. In *Proceedings of the 33rd International Conference on Machine Learning*, pages 2915–2923. PMLR, 2016.

Joshua Achiam, David Held, Aviv Tamar, and Pieter Abbeel. Constrained Policy Optimization. In *Proceedings of the 34th International Conference on Machine Learning*, pages 22–31. PMLR, 2017.

Mohammed Alshiekh, Roderick Bloem, Rüdiger Ehlers, Bettina Könighofer, Scott Niekum, and Ufuk Topcu. Safe Reinforcement Learning via Shielding. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence*, pages 2669–2678. AAAI Press, 2018.

Ibrahim Althamary, Chih-Wei Huang, and Phone Lin. A Survey on Multi-Agent Reinforcement Learning Methods for Vehicular Networks. In *15th International Wireless Communications & Mobile Computing Conference*, pages 1154–1159. IEEE, 2019.

Eitan Altman. *Constrained Markov decision processes*, volume 7. CRC Press, 1999.

Susan Amin, Maziar Gomrokchi, Harsh Satija, Herke van Hoof, and Doina Precup. A Survey of Exploration Methods in Reinforcement Learning. *arXiv preprint arXiv:2109.00157*, 2021.

Dario Amodei, Chris Olah, Jacob Steinhardt, Paul F Christiano, John Schulman, and Dan Mané. Concrete Problems in AI Safety. *arXiv preprint arXiv:1606.06565*, 2016.

Ignasi Andrés, Leliane N de Barros, Denis D Mauá, and Thiago D Simão. When a Robot Reaches Out for Human Help. In *Advances in Artificial Intelligence - 16th Ibero-American Conference on AI*, pages 277–289. Springer, 2018.

Szilard Aradi. Survey of Deep Reinforcement Learning for Motion Planning of Autonomous Vehicles. *IEEE Transactions on Intelligent Transportation Systems*, pages 1–20, 2020.

Brenna D Argall, Sonia Chernova, Manuela M Veloso, and Brett Browning. A survey of robot learning from demonstration. *Robotics Auton. Syst.*, 57(5):469–483, 2009.

Christopher G Atkeson and Juan Carlos Santamaría. A Comparison of Direct and Model-Based Reinforcement Learning. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 3557–3564. IEEE, 1997.

Peter Auer and Ronald Ortner. Logarithmic Online Regret Bounds for Undiscounted Reinforcement Learning. In *Advances in Neural Information Processing Systems 19*, pages 49–56. MIT Press, 2006.

Batu Aytemiz, Mikhail Jacob, and Sam Devlin. Acting with Style: Towards Designer-centred Reinforcement Learning for the Video Games Industry. Reinforcement Learning for Humans, Computer, and Interaction Workshop at ACM CHI, 2021.

Bowen Baker, Ingmar Kanitscheider, Todor M Markov, Yi Wu, Glenn Powell, Bob McGrew, and Igor Mordatch. Emergent Tool Use From Multi-Agent Autocurricula. In *Proceedings of the 8th International Conference on Learning Representations*, pages 1–15. OpenReview.net, 2020.

Sumeet Batra, Zhehui Huang, Aleksei Petrenko, Tushar Kumar, Artem Molchanov, and Gaurav Sukhatme. Decentralized Control of Quadrotor Swarms with End-to-end Deep Reinforcement Learning. *arXiv preprint arXiv:2109.07735*, 2021.

Marc G Bellemare, Sriram Srinivasan, Georg Ostrovski, Tom Schaul, David Saxton, and Rémi Munos. Unifying Count-based Exploration and Intrinsic Motivation. In *Advances in Neural Information Processing Systems 29*, pages 1471–1479. Curran Associates, Inc., 2016.

Felix Berkenkamp, Matteo Turchetta, Angela P Schoellig, and Andreas Krause. Safe Model-based Reinforcement Learning with Stability Guarantees. In *Advances in Neural Information Processing Systems 30*, pages 908–918. Curran Associates, Inc., 2017.

J D Biersdorfer. How to Talk to the World Through Free Translation Apps. The New York Times, 2021. URL `https://www.nytimes.com/2021/10/20/technology/personaltech/google-apple-translate-language.html`.

Craig Boutilier, Richard Dearden, and Moisés Goldszmidt. Exploiting Structure in Policy Construction. In *Proc. Int. Joint Conf. on Artificial Intelligence*, pages 1104–1113. Morgan Kaufmann, 1995.

Craig Boutilier, Thomas Dean, and Steve Hanks. Decision-Theoretic Planning: Structural Assumptions and Computational Leverage. *Journal of Artificial Intelligence Research*, 11:1–94, 1999.

Ronen I Brafman and Moshe Tennenholtz. R-max — A General Polynomial Time Algorithm for Near-Optimal Reinforcement Learning. *Journal of Machine Learning Research*, 3:213–231, 2002.

Tomáš Brázdil, Krishnendu Chatterjee, Petr Novotný, and Jiří Vahala. Reinforcement Learning of Risk-Constrained Policies in Markov Decision Processes. In *Proceedings of the Thirty-Fourth AAAI Conference on Artificial Intelligence*, pages 9794–9801. AAAI Press, 2020.

Jacob Buckman, Carles Gelada, and Marc G Bellemare. The Importance of Pessimism in Fixed-Dataset Policy Optimization. In *Proceedings of the 9th International Conference on Learning Representations*, pages 1–11. OpenReview.net, 2021.

Thiago P Bueno, Leliane N de Barros, Denis D Mauá, and Scott Sanner. Deep Reactive Policies for Planning in Stochastic Nonlinear Domains. In *Proceedings of the Thirty-Third AAAI Conference on Artificial Intelligence*, pages 7530–7537. AAAI Press, 2019.

Yuri Burda, Harrison Edwards, Amos Storkey, and Oleg Klimov. Exploration by Random Network Distillation. In *Proceedings of the 7th International Conference on Learning Representations*, pages 1–13. OpenReview.net, 2019.

Miguel Calvo-Fullana, Luiz F O Chamon, and Santiago Paternain. Towards Safe Continuing Task Reinforcement Learning. In *American Control Conference*, pages 902–908. IEEE, 2021.

Pedro M Castro, Lige Sun, and Iiro Harjunkoski. Resource-task network formulations for industrial demand side management of a steel plant. *Industrial and Engineering Chemistry Research*, 52:13046–13058, 2013.

Carlos Celemin, Javier Ruiz-del-Solar, and Jens Kober. A fast hybrid reinforcement learning framework with human corrective feedback. *Autonomous Robots*, 43:1173–1186, 2019.

Doran Chakraborty and Peter Stone. Structure Learning in Ergodic Factored MDPs without Knowledge of the Transition Function's In-Degree. In *Proceedings of the 28th International Conference on Machine Learning*, pages 737–744. Omnipress, 2011.

Minmin Chen, Alex Beutel, Paul Covington, Sagar Jain, Francois Belletti, and Ed H Chi. Top-K Off-Policy Correction for a REINFORCE Recommender System. In *Proceedings of the Twelfth ACM International Conference on Web Search and Data Mining*, pages 456–464. ACM, 2019.

Wuhui Chen, Xiaoyu Qiu, Ting Cai, Hong-Ning Dai, Zibin Zheng, and Yan Zhang. Deep Reinforcement Learning for Internet of Things: A Comprehensive Survey. *IEEE Commun. Surv. Tutorials*, 23(3):1659–1692, 2021a.

Xiaoyu Chen, Jiachen Hu, Lihong Li, and Liwei Wang. Efficient Reinforcement Learning in Factored MDPs with Application to Constrained RL. In *Proceedings of the 9th International Conference on Learning Representations*, pages 1–10. OpenReview.net, 2021b.

Rohan Chitnis, Tom Silver, Beomjoon Kim, Leslie P Kaelbling, and Tomás Lozano-Perez. CAMPs: Learning Context-Specific Abstractions for Efficient Planning in Factored MDPs. In *4th Conference on Robot Learning*, pages 64–79. PMLR, 2020.

Yinlam Chow, Mohammad Ghavamzadeh, Lucas Janson, and Marco Pavone. Risk-Constrained Reinforcement Learning with Percentile Risk Criteria. *Journal of Machine Learning Research*, 18(167):1–51, 2018a.

Yinlam Chow, Ofir Nachum, Edgar A Duéñez-Guzmán, and Mohammad Ghavamzadeh. A Lyapunov-based Approach to Safe Reinforcement Learning. In *Advances in Neural Information Processing Systems 31*, pages 8103–8112. Curran Associates, Inc., 2018b.

William R Clements, Benoît-Marie Robaglia, Bastien van Delft, Reda Bahi Slaoui, and Sébastien Toth. Estimating Risk and Uncertainty in Deep Reinforcement Learning. *arXiv preprint arXiv:1905.09638*, presented at the ICML 2020 Workshop on Uncertainty and Robustness in Deep Learning, 2019.

Andrew Cohen, Lei Yu, and Robert Wright. Diverse Exploration for Fast and Safe Policy Improvement. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence*, pages 2876–2883. AAAI Press, 2018.

Antonio Coronato, Muddasar Naeem, Giuseppe De Pietro, and Giovanni Paragliola. Reinforcement learning for intelligent healthcare applications: A survey. *Artif. Intell. Medicine*, 109(101964):1–16, 2020.

Gal Dalal, Krishnamurthy Dvijotham, Matej Vecerík, Todd Hester, Cosmin Paduraru, and Yuval Tassa. Safe Exploration in Continuous Action Spaces. *arXiv preprint arXiv:1801.08757*, 2018.

Frits de Nijs, Erwin Walraven, Mathijs M de Weerdt, and Matthijs T J Spaan. Bounding the Probability of Resource Constraint Violations in Multi-Agent MDPs. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence*, pages 3562–3568. AAAI Press, 2017.

Frits de Nijs, Erwin Walraven, Mathijs M de Weerdt, and Matthijs T J Spaan. Constrained Multiagent Markov Decision Processes: a Taxonomy of Problems and Algorithms. *Journal of Artificial Intelligence Research*, 70:955–1001, 2021.

Thomas Dean and Keiji Kanazawa. A model for reasoning about persistence and causation. *Computational Intelligence*, 5(3):142–150, 1989.

Richard Dearden and Craig Boutilier. Abstraction and approximate decision-theoretic planning. *Artificial Intelligence*, 89(1):219–283, 1997.

Thomas Degris, Olivier Sigaud, and Pierre-Henri Wuillemin. Learning the Structure of Factored Markov Decision Processes in Reinforcement Learning Problems. In *Proceedings of the 23rd International Conference on Machine Learning*, pages 257–264. ACM, 2006.

Thomas Degris, Olivier Sigaud, and Pierre-Henri Wuillemin. Exploiting Additive Structure in Factored MDPs for Reinforcement Learning. In *8th European Workshop on Recent Advances in Reinforcement Learning, Revised and Selected Papers*, volume 5323 of *Lecture Notes in Computer Science*, pages 15–26. Springer, 2008.

Sheelabhadra Dey, Sumedh Pendurkar, Guni Sharon, and Josiah P Hanna. A Joint Imitation-Reinforcement Learning Framework for Reduced Baseline Regret. In *Proc. of International Conference on Intelligent Robots and Systems*, pages 1–7. IEEE, 2021.

Thomas G Dietterich. The MAXQ Method for Hierarchical Reinforcement Learning. In *Proceedings of the Fifteenth International Conference on Machine Learning*, pages 118–126. Morgan Kaufmann, 1998.

Thomas G Dietterich, George Trimponias, and Zhitang Chen. Discovering and Removing Exogenous State Variables and Rewards for Reinforcement Learning. In *Proceedings of the 35th International Conference on Machine Learning*, pages 1262–1270. PMLR, 2018.

Carlos Diuk, Lihong Li, and Bethany R Leffler. The Adaptive k-meteorologists Problem and Its Application to Structure Learning and Feature Selection in Reinforcement Learning. In *Proceedings of the 26th International Conference on Machine Learning*, pages 249–256. ACM, 2009.

Michael Duff. *Optimal Learning: Computational procedures for Bayes-adaptive Markov decision processes.* PhD thesis, University of Massachusetts, Amherst, United States, 2002.

Gabriel Dulac-Arnold, Nir Levine, Daniel J Mankowitz, Jerry Li, Cosmin Paduraru, Sven Gowal, and Todd Hester. Challenges of real-world reinforcement learning: definitions, benchmarks and analysis. *Machine Learning*, 110(9):2419–2468, 2021.

Yonathan Efroni, Shie Mannor, and Matteo Pirotta. Exploration-Exploitation in Constrained MDPs. *arXiv preprint arXiv:2003.02189*, presented at the ICML 2020 Workshop on Theoretical Foundations of Reinforcement Learning, 2020.

Damien Ernst, Pierre Geurts, and Louis Wehenkel. Tree-based batch mode reinforcement learning. *Journal of Machine Learning Research*, 6(18):503–556, 2005.

Mehdi Fatemi, Shikhar Sharma, Harm van Seijen, and Samira Ebrahimi Kahou. Dead-ends and Secure Exploration in Reinforcement Learning. In *Proceedings of the 36th International Conference on Machine Learning*, pages 1873–1881. PMLR, 2019.

Eugene A Feinberg and Adam Shwartz. *Handbook of Markov Decision Processes: Methods and Applications.* Springer US, 2002.

Zhengzhu Feng and Eric A Hansen. Symbolic Heuristic Search for Factored Markov Decision Processes. In *Proceedings of the Eighteenth National Conference on Artificial Intelligence*, pages 455–460. AAAI Press, 2002.

Raphaël Féraud, Reda Alami, and Romain Laroche. Decentralized Exploration in Multi-Armed Bandits. In *Proceedings of the 36th International Conference on Machine Learning*, pages 1901–1909. PMLR, 2019.

Lior Fox, Leshem Choshen, and Yonatan Loewenstein. DORA The Explorer: Directed Outreaching Reinforcement Action-Selection. In *Proceedings of the 6th International Conference on Learning Representations*, pages 1–11. OpenReview.net, 2018.

Scott Fujimoto, David Meger, and Doina Precup. Off-Policy Deep Reinforcement Learning without Exploration. In *Proceedings of the 36th International Conference on Machine Learning*, pages 2052–2062. PMLR, 2019.

Javier García and Fernando Fernández. A Comprehensive Survey on Safe Reinforcement Learning. *Journal of Machine Learning Research*, 16:1437–1480, 2015.

Ather Gattami, Qinbo Bai, and Vaneet Aggarwal. Reinforcement Learning for Constrained Markov Decision Processes. In *Proceedings of the 24th International Conference on Artificial Intelligence and Statistics*, pages 2656–2664. PMLR, 2021.

Peter Geibel. Reinforcement Learning for MDPs with Constraints. In *17th European Conference on Machine Learning*, pages 646–653. Springer Berlin Heidelberg, 2006.

Florian Geißer and David Speck. Prost-DD — Utilizing Symbolic Classical Planning in THTS. Sixth International Probabilistic Planning Competition (IPC-6): planner abstracts, 2018.

Mahsa Ghasemi, Evan Scope Crafts, Bo Zhao, and Ufuk Topcu. Multiple Plans are Better than One: Diverse Stochastic Planning. In *Proceedings of the Thirty-First International Conference on Automated Planning and Scheduling*, pages 140–148. AAAI Press, 2021.

Robert Givan, Sonia M Leach, and Thomas Dean. Bounded-parameter Markov decision processes. *Artificial Intelligence*, 122:71–109, 2000.

Robert Givan, Thomas Dean, and Matthew Greig. Equivalence notions and model minimization in Markov decision processes. *Artificial Intelligence*, 147(1-2):163–223, 2003.

Mevludin Glavic, Raphaël Fonteneau, and Damien Ernst. Reinforcement Learning for Electric Power System Decision and Control: Past Considerations and Perspectives. *IFAC-PapersOnLine*, 50(1):6918–6927, 2017.

Nakul Gopalan, Marie desJardins, Michael L Littman, James MacGlashan, Shawn Squire, Stefanie Tellex, John Winder, and Lawson L S Wong. Planning with Abstract Markov Decision Processes. In *Proceedings of the Twenty-Seventh International Conference on Automated Planning and Scheduling*, pages 480–488. AAAI Press, 2017.

Abhijit Gosavi. Reinforcement Learning for Model Building and Variance-penalized Control. In *Proceedings of the 2009 Winter Simulation Conference*, pages 373–379. IEEE, 2009.

Carlos Guestrin, Relu Patrascu, and Dale Schuurmans. Algorithm-Directed Exploration for Model-Based Reinforcement Learning in Factored MDPs. In *Proceedings of the 19th International Conference on Machine Learning*, pages 235–242. Morgan Kaufmann, 2002.

Carlos Guestrin, Daphne Koller, Ronald Parr, and Shobha Venkataraman. Efficient Solution Algorithms for Factored MDPs. *Journal of Artificial Intelligence Research*, 19: 399–468, 2003.

Sehoon Ha, Peng Xu, Zhenyu Tan, Sergey Levine, and Jie Tan. Learning to Walk in the Real World with Minimal Human Effort. In *4th Conference on Robot Learning*, pages 1110–1120. PMLR, 2020.

Assaf Hallak, François Schnitzler, Timothy Arthur Mann, and Shie Mannor. Off-policy Model-based Learning under Unknown Factored Dynamics. In *Proceedings of the 32nd International Conference on Machine Learning*, pages 711–719. PMLR, 2015.

Eric A Hansen. An integrated approach to solving influence diagrams and finite-horizon partially observable decision processes. *Artificial Intelligence*, 294(103431):1–47, 2021.

Eric A Hansen and Thomas J Bowman. Improved Vector Pruning in Exact Algorithms for Solving POMDPs. In *Proceedings of the 36th Conference on Uncertainty in Artificial Intelligence*, pages 1258–1267. PMLR, 2020.

Mohammadhosein Hasanbeig, Alessandro Abate, and Daniel Kroening. Cautious Reinforcement Learning with Logical Constraints. In *Proceedings of the 19th International Conference on Autonomous Agents and MultiAgent Systems*, page 483–491. IFAAMAS, 2020.

Aria HasanzadeZonuzy, Dileep M Kalathil, and Srinivas Shakkottai. Learning with Safety Constraints: Sample Complexity of Reinforcement Learning for Constrained MDPs. In *Proceedings of the Thirty-Fifth AAAI Conference on Artificial Intelligence*, pages 7667–7674. AAAI Press, 2021.

Ammar Haydari and Yasin Yilmaz. Deep Reinforcement Learning for Intelligent Transportation Systems: A Survey. *IEEE Trans. Intell. Transp. Syst.*, pages 1–22, 2020.

Conor F Hayes, Roxana Radulescu, Eugenio Bargiacchi, Johan Källström, Matthew Macfarlane, Mathieu Reymond, Timothy Verstraeten, Luisa M Zintgraf, Richard Dazeley, Fredrik Heintz, Enda Howley, Athirai A Irissappane, Patrick Mannion, Ann Nowé, Gabriel de Oliveira Ramos, Marcello Restelli, Peter Vamplew, and Diederik M Roijers. A Practical Guide to Multi-Objective Reinforcement Learning and Planning. *arXiv preprint arXiv:2103.09568*, 2021.

Matthias Heger. Consideration of Risk in Reinforcement Learning. In *Proceedings of the 11th International Conference on Machine Learning*, pages 105–111. Morgan Kaufmann, 1994.

Wassily Hoeffding. Probability Inequalities for Sums of Bounded Random Variables. *Journal of the American Statistical Association*, 58(301):13–30, 1963.

Jesse Hoey, Robert St-Aubin, Alan J Hu, and Craig Boutilier. SPUDD: Stochastic Planning using Decision Diagrams. In *Proceedings of the Fifteenth Conference on Uncertainty in Artificial Intelligence*, pages 279–288. Morgan Kaufmann, 1999.

Ronald A Howard and James E Matheson. Risk-Sensitive Markov Decision Processes. *Management Science*, 18(7):356–369, 1972.

Eyke Hüllermeier and Willem Waegeman. Aleatoric and epistemic uncertainty in machine learning: an introduction to concepts and methods. *Machine Learning*, 110: 457–506, 2021.

Garud N Iyengar. Robust Dynamic Programming. *Mathematics of Operations Research*, 30(2):257–280, 2005.

Thomas Jaksch, Ronald Ortner, and Peter Auer. Near-optimal Regret Bounds for Reinforcement Learning. *Journal of Machine Learning Research*, 11:1563–1600, 2010.

Furqan Jameel, Uzair Javaid, Wali Ullah Khan, Muhammad Naveed Aman, Haris Pervaiz, and Riku Jäntti. Reinforcement Learning in Blockchain-Enabled IIoT Networks: A Survey of Recent Advances and Open Challenges. *Sustainability*, 12(12):1–22, 2020.

Nils Jansen, Bettina Könighofer, Sebastian Junges, Alex Serban, and Roderick Bloem. Safe Reinforcement Learning Using Probabilistic Shields (Invited Paper). In *31st International Conference on Concurrency Theory*, pages 1–16. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2020.

Chi Jin, Tiancheng Jin, Haipeng Luo, Suvrit Sra, and Tiancheng Yu. Learning Adversarial MDPs with Bandit Feedback and Unknown Transition. In *Proceedings of the 37th International Conference on Machine Learning*, pages 4860–4869. PMLR, 2020a.

Chi Jin, Akshay Krishnamurthy, Max Simchowitz, and Tiancheng Yu. Reward-Free Exploration for Reinforcement Learning. In *Proceedings of the 37th International Conference on Machine Learning*, pages 4870–4879. PMLR, 2020b.

Chi Jin, Zhuoran Yang, Zhaoran Wang, and Michael I Jordan. Provably Efficient Reinforcement Learning with Linear Function Approximation. In *Proceedings of Thirty Third Conference on Learning Theory*, pages 2137–2143. PMLR, 2020c.

Ying Jin, Zhuoran Yang, and Zhaoran Wang. Is Pessimism Provably Efficient for Offline RL? In *Proceedings of the 38th International Conference on Machine Learning*, pages 5084–5096. PMLR, 2021.

Anders Jonsson and Andrew G Barto. Active Learning of Dynamic Bayesian Networks in Markov Decision Processes. In *7th International Symposium on Abstraction, Reformulation, and Approximation*, pages 273–284. Springer, 2007.

Sebastian Junges, Nils Jansen, Christian Dehnert, Ufuk Topcu, and Joost-Pieter Katoen. Safety-Constrained Reinforcement Learning for MDPs. In *International Conference on Tools and Algorithms for the Construction and Analysis of Systems*, pages 130–146. Springer, 2016.

Sham Kakade and John Langford. Approximately Optimal Approximate Reinforcement Learning. In *Proceedings of the 19th International Conference on Machine Learning*, pages 267–274. Morgan Kaufmann, 2002.

Danial Kamran, Thiago D. Simão, Qisong Yang, Canmanie T. Ponnambalam, Johannes Fischer, Matthijs T. J. Spaan, and Martin Lauer. A modern perspective on safe automated driving for different traffic dynamics using constrained reinforcement learning. In *25th IEEE International Conference on Intelligent Transportation Systems (ITSC)*, pages 4017–4023. IEEE, 2022.

Michael J Kearns and Daphne Koller. Efficient Reinforcement Learning in Factored MDPs. In *Proc. Int. Joint Conf. on Artificial Intelligence*, pages 740–747. Morgan Kaufmann, 1999.

Michael J Kearns and Satinder P Singh. Near-Optimal Reinforcement Learning in Polynomial Time. *Machine Learning*, 49(2-3):209–232, 2002.

B Ravi Kiran, Ibrahim Sobh, Victor Talpaert, Patrick Mannion, Ahmad A Al Sallab, Senthil Yogamani, and Patrick Perez. Deep Reinforcement Learning for Autonomous Driving: A Survey. *IEEE Transactions on Intelligent Transportation Systems*, pages 1–18, 2021.

Levente Kocsis and Csaba Szepesvári. Bandit Based Monte-carlo Planning. In *17th European Conference on Machine Learning*, pages 282–293. Springer, 2006.

Anna Kosiorek. Safe Optimization of Steel Manufacturing with Reinforcement Learning. Master's thesis, Delft University of Technology, Delft, The Netherlands, 2020. URL `http://resolver.tudelft.nl/uuid:efbe886c-1b39-4696-9032-3fc1bbe7e445`.

Aviral Kumar, Justin Fu, George Tucker, and Sergey Levine. Stabilizing Off-Policy Q-Learning via Bootstrapping Error Reduction. In *Advances in Neural Information Processing Systems 32*, pages 11761–11771. Curran Associates, Inc., 2019.

Saurabh Kumar, Aviral Kumar, Sergey Levine, and Chelsea Finn. One Solution is Not All You Need: Few-Shot Extrapolation via Structured MaxEnt RL. In *Advances in Neural Information Processing Systems 34*, pages 8198–8210. Curran Associates, Inc., 2020.

Michail G Lagoudakis and Ronald Parr. Least-squares Policy Iteration. *Journal of Machine Learning Research*, 4(Dec):1107–1149, 2003.

Sascha Lange, Thomas Gabel, and Martin Riedmiller. Batch Reinforcement Learning. In Marco A Wiering and Martijn van Otterlo, editors, *Reinforcement Learning: State-of-the-Art*, pages 45–73. Springer Berlin Heidelberg, 2012.

Romain Laroche and Rémi Tachet des Combes. Multi-batch Reinforcement Learning. In *Proceedings of the 4th Multidisciplinary Conference on Reinforcement Learning and Decision Making*. RLDM, 2019.

Romain Laroche, Paul Trichelair, and Rémi Tachet des Combes. Safe Policy Improvement with Baseline Bootstrapping. In *Proceedings of the 36th International Conference on Machine Learning*, pages 3652–3661. PMLR, 2019.

Hoang Le, Cameron Voloshin, and Yisong Yue. Batch Policy Learning under Constraints. In *Proceedings of the 36th International Conference on Machine Learning*, pages 3703–3712. PMLR, 2019.

Y Lecun, L Bottou, Y Bengio, and P Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

Jongmin Lee, Youngsoo Jang, Pascal Poupart, and Kee-Eung Kim. Constrained Bayesian Reinforcement Learning via Approximate Linear Programming. In *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence*, pages 2088–2095. ijcai.org, 2017.

Sergey Levine, Aviral Kumar, George Tucker, and Justin Fu. Offline Reinforcement Learning: Tutorial, Review, and Perspectives on Open Problems. *arXiv preprint arXiv:2005.01643*, 2020.

Lihong Li. Sample Complexity Bounds of Exploration. In Marco A Wiering and Martijn van Otterlo, editors, *Reinforcement Learning: State-of-the-Art*, pages 175–204. Springer Berlin Heidelberg, 2012.

Lihong Li, Thomas J Walsh, and Michael L Littman. Towards a Unified Theory of State Abstraction for MDPs. In *International Symposium on Artificial Intelligence and Mathematics*, pages 1–10. ISAIM, 2006.

Lihong Li, Michael L Littman, Thomas J Walsh, and Alexander L Strehl. Knows What It Knows: A Framework For Self-Aware Learning. *Machine Learning*, 82(3):399–443, 2011.

Shiau Hong Lim, Huan Xu, and Shie Mannor. Reinforcement Learning in Robust Markov Decision Processes. *Mathematics of Operations Research*, 41(4):1325–1353, 2016.

Jinliang Liu, Liang Xiao, Guolong Liu, and Yifeng Zhao. Active authentication with reinforcement learning based on ambient radio signals. *Multimedia Tools and Applications*, 76(3):3979–3998, 2017.

Siqi Liu, Kay Choong See, Kee Yuan Ngiam, Leo Anthony Celi, Xingzhi Sun, and Mengling Feng. Reinforcement Learning for Clinical Decision Support in Critical Care: Comprehensive Review. *Journal of Medical Internet Research*, 22(7):1–16, 2020.

Vincent Liu, James Wright, and Martha White. Exploiting Action Impact Regularity and Partially Known Models for Offline Reinforcement Learning. *arXiv preprint arXiv:2111.08066*, 2021a.

Yongshuai Liu, Avishai Halev, and Xin Liu. Policy Learning with Constraints in Model-free Reinforcement Learning: A Survey. In *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence*, pages 4508–4515. ijcai.org, 2021b.

Yuping Luo and Tengyu Ma. Learning Barrier Certificates: Towards Safe Reinforcement Learning with Zero Training-time Violations. *arXiv preprint arXiv:2108.01846*, presented at the ICML 2021 Workshop on Reinforcement Learning for Real Life, 2021.

Travis Mandel, Yun-En Liu, Sergey Levine, Emma Brunskill, and Zoran Popovic. Offline Policy Evaluation Across Representations with Applications to Educational Games. In *Proceedings of the 13th International Conference on Autonomous Agents and MultiAgent Systems*, pages 1077–1084. IFAAMAS, 2014.

Hongzi Mao, Malte Schwarzkopf, Hao He, and Mohammad Alizadeh. Towards Safe Online Reinforcement Learning in Computer Systems. Presented at the NeurIPS 2019 Workshop on Machine Learning for Systems, `http://mlforsystems.org/assets/papers/neurips2019/towards_mao_2019.pdf`, 2019.

Mausam and Andrey Kolobov. *Planning with Markov Decision Processes: An AI Perspective*. Synthesis Lectures on Artificial Intelligence and Machine Learning. Morgan & Claypool Publishers, 2012.

Colin McDiarmid. Concentration. In *Probabilistic Methods for Algorithmic Discrete Mathematics*, pages 195–248. Springer Berlin Heidelberg, 1998.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level Control Through Deep Reinforcement Learning. *Nature*, 518(7540):529–533, 2015.

Volodymyr Mnih, Adrià Puigdomènech Badia, Mehdi Mirza, Alex Graves, Timothy P Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. Asynchronous Methods for Deep Reinforcement Learning. In *Proceedings of the 33rd International Conference on Machine Learning*, pages 1928–1937. PMLR, 2016.

Thomas M Moerland, Joost Broekens, and Catholijn M Jonker. Model-based Reinforcement Learning: A Survey. *arXiv preprint arXiv:2006.16712*, 2021.

Teodor Mihai Moldovan and Pieter Abbeel. Safe Exploration in Markov Decision Processes. In *Proceedings of the 29th International Conference on Machine Learning*, pages 1711–1718. Omnipress, 2012.

Daniel A M Moreira, Karina V Delgado, Leliane N de Barros, and Denis D Mauá. Efficient algorithms for Risk-Sensitive Markov Decision Processes with limited budget. *International Journal of Approximate Reasoning*, 139:143–165, 2021.

Rémi Munos. From bandits to monte-carlo tree search: The optimistic principle applied to optimization and planning. *Foundations and Trends in Machine Learning*, 7:1–129, 2014.

Rémi Munos. Distributional Reinforcement Learning, 2018. Horizon Maths `https://vimeo.com/304849090`.

Kimia Nadjahi, Romain Laroche, and Rémi Tachet des Combes. Safe Policy Improvement with Soft Baseline Bootstrapping. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 53–68. Springer International Publishing, 2019.

Grigory Neustroev and Mathijs M de Weerdt. Generalized Optimistic Q-Learning with Provable Efficiency. In *Proceedings of the 19th International Conference on Autonomous Agents and MultiAgent Systems*, pages 913–921. IFAAMAS, 2020.

Jun Hao Alvin Ng and Ronald P A Petrick. Incremental learning of planning actions in model-based reinforcement learning. In *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence*, pages 3195–3201. ijcai.org, 2019.

Arnab Nilim and Laurent El Ghaoui. Robust Control of Markov Decision Processes with Uncertain Transition Matrices. *Operations Research*, 53(5):780–798, 2005.

Frans A Oliehoek, Matthijs T J Spaan, Shimon Whiteson, and Nikos Vlassis. Exploiting locality of interaction in factored Dec-POMDPs. In *Proceedings of the 7th International Conference on Autonomous Agents and Multiagent Systems*, pages 517–524. IFAAMAS, 2008.

Frans A Oliehoek, Stefan J Witwicki, and Leslie P Kaelbling. Influence-Based Abstraction for Multiagent Systems. In *Proceedings of the Twenty-Sixth AAAI Conference on Artificial Intelligence*, pages 1422–1428. AAAI Press, 2012.

Ronald Ortner. Adaptive aggregation for reinforcement learning in average reward Markov decision processes. *Annals OR*, 208(1):321–336, 2013.

Takayuki Osa, Joni Pajarinen, Gerhard Neumann, J Andrew Bagnell, Pieter Abbeel, and Jan Peters. An Algorithmic Perspective on Imitation Learning. *Foundations and Trends® in Robotics*, 7(1-2):1–179, 2018.

Ian Osband and Benjamin Van Roy. Near-optimal Reinforcement Learning in Factored MDPs. In *Advances in Neural Information Processing Systems 27*, pages 604–612. Curran Associates, Inc., 2014.

Ian Osband, Charles Blundell, Alexander Pritzel, and Benjamin Van Roy. Deep Exploration via Bootstrapped DQN. In *Advances in Neural Information Processing Systems 29*, pages 4026–4034. Curran Associates, Inc., 2016.

Cosmin Paduraru. *Off-policy Evaluation in Markov Decision Processes.* PhD thesis, McGill University, Montreal, Canada, 2013.

Marcel Panzer and Benedict Bender. Deep reinforcement learning in production systems: a systematic literature review. *International Journal of Production Research*, pages 1–26, 2021.

Matteo Papini, Matteo Pirotta, and Marcello Restelli. Adaptive Batch Size for Safe Policy Gradients. In *Advances in Neural Information Processing Systems 30*, pages 3591–3600. Curran Associates, Inc., 2017.

Martin Pecka and Tomás Svoboda. Safe Exploration Techniques for Reinforcement Learning – An Overview. In *First International Workshop on Modelling and Simulation for Autonomous Systems*, pages 357–375. Springer, 2014.

Xue Bin Peng, Erwin Coumans, Tingnan Zhang, Tsang-Wei Edward Lee, Jie Tan, and Sergey Levine. Learning Agile Robotic Locomotion Skills by Imitating Animals. In *Robotics: Science and Systems XVI*, pages 1–12. roboticsproceedings.org, 2020.

Marek Petrik, Mohammad Ghavamzadeh, and Yinlam Chow. Safe Policy Improvement by Minimizing Robust Baseline Regret. In *Advances in Neural Information Processing Systems 29*, pages 2298–2306. Curran Associates, Inc., 2016.

Dung T Phan, Radu Grosu, Nils Jansen, Nicola Paoletti, Scott A Smolka, and Scott D Stoller. Neural Simplex Architecture. In *12th NASA Formal Methods Symposium*, pages 97–114. Springer, 2020.

Joelle Pineau. Safe and Sound Reinforcement Learning. Keynote at the 32nd International Conference on Algorithmic Learning Theory, 2021. URL `https://youtu.be/e6n-jM1f5_4`.

Matteo Pirotta, Marcello Restelli, Alessio Pecorino, and Daniele Calandriello. Safe Policy Iteration. In *Proceedings of the 30th International Conference on Machine Learning*, pages 307–315. PMLR, 2013.

Canmanie T Ponnambalam, Frans A Oliehoek, and Matthijs T J Spaan. Abstraction-Guided Policy Recovery from Expert Demonstrations. In *Proceedings of the Thirty-First International Conference on Automated Planning and Scheduling*, pages 560–568. AAAI Press, 2021.

Pascal Poupart, Aarti Malhotra, Pei Pei, Kee-Eung Kim, Bongseok Goh, and Michael Bowling. Approximate Linear Programming for Constrained Partially Observable Markov Decision Processes. In *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence*, pages 3342–3348. AAAI Press, 2015.

Bharat Prakash, Mohit Khatwani, Nicholas R Waytowich, and Tinoosh Mohsenin. Improving Safety in Reinforcement Learning Using Model-Based Architectures and Human Intervention. In *Proceedings of the Thirty-Second International Florida Artificial Intelligence Research Society Conference*, pages 50–55. AAAI Press, 2019.

Martin L Puterman. *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. John Wiley & Sons, Inc., 1st edition, 1994.

Yichen Qian, Jun Wu, Rui Wang, Fusheng Zhu, and Wei Zhang. Survey on Reinforcement Learning Applications in Communication Networks. *J. Commun. Inf. Networks*, 4(2): 30–39, 2019.

Zengyi Qin, Yuxiao Chen, and Chuchu Fan. Density Constrained Reinforcement Learning. In *Proceedings of the 38th International Conference on Machine Learning*, pages 8682–8692. PMLR, 2021.

Paria Rashidinejad, Banghua Zhu, Cong Ma, Jiantao Jiao, and Stuart Russell. Bridging Offline Reinforcement Learning and Imitation Learning: A Tale of Pessimism. In *Advances in Neural Information Processing Systems 35*. Curran Associates, Inc., 2021. *arXiv preprint arXiv:2103.12021*.

Harish Ravichandar, Athanasios S Polydoros, Sonia Chernova, and Aude Billard. Recent Advances in Robot Learning from Demonstration. *Annual Review of Control, Robotics, and Autonomous Systems*, 3(1):297–330, 2020.

Alex Ray, Joshua Achiam, and Dario Amodei. Benchmarking Safe Exploration in Deep Reinforcement Learning, 2019. URL `https://github.com/openai/safety-gym`.

Alberto Reyes, Matthijs T J Spaan, and L Enrique Sucar. An Intelligent Assistant for Power Plants Based on Factored MDPs. In *15th International Conference on Intelligent System Applications to Power Systems*, pages 1–6. IEEE, 2009.

Alberto Reyes, Pablo H Ibargüengoytia, Inés Romero-Leon, David Pech, and Mónica Borunda. Building Optimal Operation Policies for Dam Management Using Factored

Markov Decision Processes. In *Advances in Artificial Intelligence and Its Applications - 14th Mexican International Conference on Artificial Intelligence*, pages 475–484. Springer, 2015.

Martin Riedmiller, Jost Tobias Springenberg, Roland Hafner, and Nicolas Heess. Collect & Infer - a fresh look at data-efficient Reinforcement Learning. *arXiv preprint arXiv:2108.10273*, 2021.

Marc Rigter, Bruno Lacerda, and Nick Hawes. Risk-Averse Bayes-Adaptive Reinforcement Learning. In *Advances in Neural Information Processing Systems 35*. Curran Associates, Inc., 2021. *arXiv preprint arXiv:2102.05762*.

Melrose Roderick, Vaishnavh Nagarajan, and J Zico Kolter. Provably Safe PAC-MDP Exploration Using Analogies. In *Proceedings of the 24th International Conference on Artificial Intelligence and Statistics*, pages 1216–1224. PMLR, 2021.

Diederik M Roijers, Peter Vamplew, Shimon Whiteson, and Richard Dazeley. A Survey of Multi-Objective Sequential Decision-Making. *Journal of Artificial Intelligence Research*, 48:67–113, 2013.

Aviv Rosenberg and Yishay Mansour. Online Convex Optimization in Adversarial Markov Decision Processes. In *Proceedings of the 36th International Conference on Machine Learning*, pages 5478–5486. PMLR, 2019.

Stéphane Ross and Joelle Pineau. Model-Based Bayesian Reinforcement Learning in Large Structured Domains. In *Proceedings of the Twenty-Fourth Conference on Uncertainty in Artificial Intelligence*, pages 476–483. AUAI Press, 2008.

Maxim Rostov and Michael Kaisers. Robust Online Planning with Imperfect Models. Adaptive and Learning Agents Workshop at AAMAS, 2021. URL `https://ala2021.vub.ac.be/papers/ALA2021_paper_25.pdf`.

Julien Roy, Joshua Roger and, Girgis Romoff, Pierre-Luc Bacon, and Christopher Pal. Direct Behavior Specification via Constrained Reinforcement Learning. *arXiv preprint arXiv:2112.12228*, 2021.

Gavin Adrian Rummery and Mahesan Niranjan. On-Line Q-Learning Using Connectionist Systems. Technical report, Engineering Department, Cambridge University, Cambridge, United Kingdom, November 1994. URL `http://mi.eng.cam.ac.uk/reports/svr-ftp/auto-pdf/rummery_tr166.pdf`.

Reazul Hasan Russel and Marek Petrik. Beyond Confidence Regions: Tight Bayesian Ambiguity Sets for Robust MDPs. In *Advances in Neural Information Processing Systems 32*, pages 7049–7058. Curran Associates, Inc., 2019.

Régis Sabbadin, Florent Teichteil-Königsbuch, and Vincent Vidal. Planning in Artificial Intelligence. In Pierre Marquis, Odile Papini, and Henri Prade, editors, *A Guided Tour of Artificial Intelligence Research, Volume II: AI Algorithms*, pages 285–312. Springer International Publishing, 2020.

Scott Sanner. Relational Dynamic Influence Diagram Language (RDDL): Language Description, 2010. URL `http://users.cecs.anu.edu.au/~ssanner/IPPC_2011/RDDL.pdf`.

Harsh Satija, Philip S Thomas, Joelle Pineau, and Romain Laroche. Multi-Objective SPIBB: Seldonian Offline Policy Improvement with Safety Constraints in Finite MDPs. In *Advances in Neural Information Processing Systems 35*. Curran Associates, Inc., 2021. *arXiv preprint arXiv:2106.00099*.

William Saunders, Girish Sastry, Andreas Stuhlmüller, and Owain Evans. Trial without Error: Towards Safe Reinforcement Learning via Human Intervention. In *Proceedings of the 17th International Conference on Autonomous Agents and MultiAgent Systems*, pages 2067–2069. IFAAMAS, 2018.

Yagiz Savas, Melkior Ornik, Murat Cubuktepe, and Ufuk Topcu. Entropy Maximization for Constrained Markov Decision Processes. In *56th Annual Allerton Conference on Communication, Control, and Computing*, pages 911–918. IEEE, 2018.

Frank N H Schrama, Daan Merkestein, Mart Jansen, Walter Vortrefflich, and Bart van den Berg. Steel Plant Model for Optimization of Steel Plant Logistics. In *Proceedings of the 6th International Congress on the Science and Technology of Steelmaking*, pages 204–207. Chinese Society for Metals, 2015.

John Schulman, Sergey Levine, Pieter Abbeel, Michael I Jordan, and Philipp Moritz. Trust Region Policy Optimization. In *Proceedings of the 32nd International Conference on Machine Learning*, pages 1889–1897. PMLR, 2015.

John Schulman, Philipp Moritz, Sergey Levine, Michael I Jordan, and Pieter Abbeel. High-Dimensional Continuous Control Using Generalized Advantage Estimation. In *Proceedings of the 4th International Conference on Learning Representations*, pages 1–11. OpenReview.net, 2016.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal Policy Optimization Algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

Iulian V Serban, Alessandro Sordoni, Yoshua Bengio, Aaron Courville, and Joelle Pineau. Building End-to-end Dialogue Systems Using Generative Hierarchical Neural Network Models. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*, pages 3776–3784. AAAI Press, 2016.

Roshan Shariff and Csaba Szepesvári. Efficient Planning in Large MDPs with Weak Linear Function Approximation. In *Advances in Neural Information Processing Systems 33*, pages 19163–19174. Curran Associates, Inc., 2020.

Apoorva Sharma, James Harrison, Matthew Tsao, and Marco Pavone. Robust and Adaptive Planning under Model Uncertainty. In *Proceedings of the Twenty-Ninth International Conference on Automated Planning and Scheduling*, pages 410–418. AAAI Press, 2019.

Felipe Leno da Silva, Pablo Hernandez-Leal, Bilal Kartal, and Matthew E Taylor. Uncertainty-Aware Action Advising for Deep Reinforcement Learning Agents. In *The Thirty-Fourth AAAI Conference on Artificial Intelligence*, pages 5792–5799. AAAI Press, 2020.

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

David Silver, Julian Schrittwieser, Karen Simonyan, Ioannis Antonoglou, Aja Huang, Arthur Guez, Thomas Hubert, Lucas Baker, Matthew Lai, Adrian Bolton, Yutian Chen, Timothy P Lillicrap, Fan Hui, Laurent Sifre, George van den Driessche, Thore Graepel, and Demis Hassabis. Mastering the game of Go without human knowledge. *Nature*, 550(7676):354–359, 2017.

Thiago D Simão. Safe and Sample-Efficient Reinforcement Learning Algorithms for Factored Environments. In *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence*, pages 6460–6461. ijcai.org, 2019.

Thiago D Simão and Matthijs T J Spaan. Safe Policy Improvement with Baseline Bootstrapping in Factored Environments. In *Proceedings of the Thirty-Third AAAI Conference on Artificial Intelligence*, pages 4967–4974. AAAI Press, 2019a.

Thiago D Simão and Matthijs T J Spaan. Structure Learning for Safe Policy Improvement. In *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence*, pages 3453–3459. ijcai.org, 2019b.

Thiago D Simão, Romain Laroche, and Rémi Tachet des Combes. Safe Policy Improvement with an Estimated Baseline Policy. In *Proceedings of the 19th International Conference on Autonomous Agents and MultiAgent Systems*, page 1269–1277. IFAAMAS, 2020.

Thiago D Simão, Nils Jansen, and Matthijs T J Spaan. AlwaysSafe: Reinforcement Learning Without Safety Constraint Violations During Training. In *Proceedings of the 20th International Conference on Autonomous Agents and MultiAgent Systems*, pages 1226–1235. IFAAMAS, 2021.

Alexander L Strehl. Model-Based Reinforcement Learning in Factored-State MDPs. In *2007 IEEE International Symposium on Approximate Dynamic Programming and Reinforcement Learning*, pages 103–110. IEEE, 2007.

Alexander L Strehl and Michael L Littman. An analysis of model-based Interval Estimation for Markov Decision Processes. *Journal of Computer and System Sciences*, 74: 1309–1331, 2008.

Alexander L Strehl, Carlos Diuk, and Michael L Littman. Efficient Structure Learning in Factored-State MDPs. In *Proceedings of the Twenty-Second AAAI Conference on Artificial Intelligence*, pages 645–650. AAAI Press, 2007.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. Sequence to Sequence Learning with Neural Networks. In *Advances in Neural Information Processing Systems 27*, pages 3104–3112. Curran Associates, Inc., 2014.

Richard S Sutton and Andrew G Barto. *Reinforcement Learning: An Introduction*. MIT Press, 2 edition, 2018.

Richard S. Sutton, Doina Precup, and Satinder Singh. Between MDPs and Semi-MDPs: A Framework for Temporal Abstraction in Reinforcement Learning. *Artif. Intell.*, 112 (1-2):181–211, 1999.

Majid Alkaee Taleghan and Thomas G Dietterich. Efficient Exploration for Constrained MDPs. In *2018 AAAI Spring Symposia*, pages 313–319. AAAI Press, 2018.

Maryam Tavakol and Ulf Brefeld. Factored MDPs for Detecting Topics of User Sessions. In *Eighth ACM Conference on Recommender Systems*, pages 33–40. ACM, 2014.

Matthew E Taylor, Nicholas K Jong, and Peter Stone. Transferring Instances for Model-Based Reinforcement Learning. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 488–505. Springer, 2008.

Chen Tessler, Daniel J Mankowitz, and Shie Mannor. Reward Constrained Policy Optimization. In *Proceedings of the 7th International Conference on Learning Representations*, pages 1–11. OpenReview.net, 2019.

Philip S Thomas, Georgios Theocharous, and Mohammad Ghavamzadeh. High-Confidence Off-Policy Evaluation. In *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence*, pages 3000–3006. AAAI Press, 2015a.

Philip S Thomas, Georgios Theocharous, and Mohammad Ghavamzadeh. High Confidence Policy Improvement. In *Proceedings of the 32nd International Conference on Machine Learning*, pages 2380–2388. PMLR, 2015b.

Andrea Lockerd Thomaz and Cynthia Breazeal. Teachable robots: Understanding human teaching behavior to build more effective robot learners. *Artificial Intelligence*, 172(6-7):716–737, 2008.

Arryon D Tijsma, Madalina M Drugan, and Marco A Wiering. Comparing Exploration Strategies for Q-learning in Random Stochastic Mazes. In *IEEE Symposium Series on Computational Intelligence*, pages 1–8. IEEE, 2016.

Matteo Turchetta, Felix Berkenkamp, and Andreas Krause. Safe Exploration in Finite Markov Decision Processes with Gaussian Processes. In *Advances in Neural Information Processing Systems 29*, pages 4312–4320. Curran Associates, Inc., 2016.

Matteo Turchetta, Felix Berkenkamp, and Andreas Krause. Safe Exploration for Interactive Machine Learning. In *Advances in Neural Information Processing Systems 32*, pages 2887–2897. Curran Associates, Inc., 2019.

Matteo Turchetta, Andrey Kolobov, Shital Shah, Andreas Krause, and Alekh Agarwal. Safe Reinforcement Learning via Curriculum Induction. In *Advances in Neural Information Processing Systems 33*, pages 12151–12162. Curran Associates, Inc., 2020.

Aashma Uprety and Danda B Rawat. Reinforcement Learning for IoT Security: A Comprehensive Survey. *IEEE Internet of Things Journal*, 8(11):8693–8706, 2021.

Hado van Hasselt, Arthur Guez, and David Silver. Deep Reinforcement Learning with Double Q-learning. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*, pages 2094–2100. AAAI Press, 2016.

Harm van Seijen, Shimon Whiteson, and Leon J H M Kester. Efficient Abstraction Selection in Reinforcement Learning. *Computational Intelligence*, 30(4):657–699, 2014.

Harm van Seijen, Mehdi Fatemi, Romain Laroche, Joshua Romoff, Tavian Barnes, and Jeffrey Tsang. Hybrid Reward Architecture for Reinforcement Learning. In *Advances in Neural Information Processing Systems 31*, pages 5392–5402. Curran Associates, Inc., 2017.

José R Vázquez-Canteli and Zoltán Nagy. Reinforcement learning for demand response: A review of algorithms and modeling techniques. *Applied Energy*, 235:1072–1089, 2019.

Oriol Vinyals, Igor Babuschkin, Wojciech M Czarnecki, Michaël Mathieu, Andrew Dudzik, Junyoung Chung, David H Choi, Richard Powell, Timo Ewalds, Petko Georgiev, Junhyuk Oh, Dan Horgan, Manuel Kroiss, Ivo Danihelka, Aja Huang, Laurent Sifre, Trevor Cai, John P Agapiou, Max Jaderberg, Alexander Sasha Vezhnevets, Rémi Leblond, Tobias Pohlen, Valentin Dalibard, David Budden, Yury Sulsky, James Molloy, Tom Le Paine, Çaglar Gülçehre, Ziyu Wang, Tobias Pfaff, Yuhuai Wu, Roman Ring, Dani Yogatama, Dario Wünsch, Katrina McKinney, Oliver Smith, Tom Schaul, Timothy P Lillicrap, Koray Kavukcuoglu, Demis Hassabis, Chris Apps, and David Silver. Grandmaster level in StarCraft II using multi-agent reinforcement learning. *Nature*, 575(7782):350–354, 2019.

Nikos Vlassis, Mohammad Ghavamzadeh, Shie Mannor, and Pascal Poupart. Bayesian Reinforcement Learning. In Marco A Wiering and Martijn van Otterlo, editors, *Reinforcement Learning: State-of-the-Art*, pages 359–386. Springer Berlin Heidelberg, 2012.

Akifumi Wachi and Yanan Sui. Safe Reinforcement Learning in Constrained Markov Decision Processes. In *Proceedings of the 37th International Conference on Machine Learning*, pages 9797–9806. PMLR, 2020.

Akifumi Wachi, Yanan Sui, Yisong Yue, and Masahiro Ono. Safe Exploration and Optimization of Constrained MDPs Using Gaussian Processes. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence*, pages 6548–6556. AAAI Press, 2018.

Erwin Walraven and Matthijs T J Spaan. Column Generation Algorithms for Constrained POMDPs. *Journal of Artificial Intelligence Research*, 62:489–533, 2018.

Ruosong Wang, Simon S Du, Lin F Yang, and Russ R Salakhutdinov. On Reward-Free Reinforcement Learning with Linear Function Approximation. In *Advances in Neural Information Processing Systems 33*, pages 17816–17826. Curran Associates, Inc., 2020.

Zhaorong Wang, Meng Wang, Jingqi Zhang, Yingfeng Chen, and Chongjie Zhang. Reward-Constrained Behavior Cloning. In *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence*, pages 3169–3175. ijcai.org, 2021.

Zhe Wang and Tianzhen Hong. Reinforcement learning for building controls: The opportunities and challenges. *Applied Energy*, 269(115036):1–18, 2020.

Christopher John Cornish Hellaby Watkins. *Learning From Delayed Rewards*. PhD thesis, King's College, Cambridge, United Kingdom, 1989.

Hua Wei, Guanjie Zheng, Vikash Gayah, and Zhenhui Li. Recent Advances in Reinforcement Learning for Traffic Signal Control: A Survey of Models and Evaluation. *ACM SIGKDD Explorations Newsletter*, 22(2):12–18, 2021.

Tsachy Weissman, Erik Ordentlich, Gadiel Seroussi, Sergio Verdu, and Marcelo J Weinberger. Inequalities for the L_1 Deviation of the Empirical Distribution. Technical report, Hewlett-Packard Labs, Palo Alto, United States, June 2003. URL `https://www.hpl.hp.com/techreports/2003/HPL-2003-97R1.html`.

Wolfram Wiesemann, Daniel Kuhn, and Berç Rustem. Robust Markov Decision Processes. *Mathematics of Operations Research*, 38(1):153–183, 2013.

Yulei Wu, Zehua Wang, Yuxiang Ma, and Victor C M Leung. Deep reinforcement learning for blockchain in industrial IoT: A survey. *Comput. Networks*, 191(108004):1–11, 2021.

Chenjun Xiao, Ilbin Lee, Bo Dai, Dale Schuurmans, and Csaba Szepesvári. On the Sample Complexity of Batch Reinforcement Learning with Policy-Induced Data. *arXiv preprint arXiv:2106.09973*, 2021a.

Chenjun Xiao, Yifan Wu, Jincheng Mei, Bo Dai, Tor Lattimore, Lihong Li, Csaba Szepesvári, and Dale Schuurmans. On the Optimality of Batch Policy Optimization Algorithms. In *Proceedings of the 38th International Conference on Machine Learning*, pages 11362–11371. PMLR, 2021b.

Tengyang Xie and Nan Jiang. Q* Approximation Schemes for Batch Reinforcement Learning: A Theoretical Comparison. In *Proceedings of the Thirty-Sixth Conference on Uncertainty in Artificial Intelligence*, pages 550–559. PMLR, 2020.

Huan Xu and Shie Mannor. Parametric Regret in Uncertain Markov Decision Processes. In *Proceedings of the 48th IEEE Conference on Decision and Control (CDC) held jointly with the 28th Chinese Control Conference*, pages 3606–3613. IEEE, 2009.

Qisong Yang, Thiago D Simão, Simon H Tindemans, and Matthijs T J Spaan. WCSAC: Worst-Case Soft Actor Critic for Safety-Constrained Reinforcement Learning. In *Proceedings of the Thirty-Fifth AAAI Conference on Artificial Intelligence*, pages 10639–10646. AAAI Press, 2021.

Qisong Yang, Thiago D. Simão, Simon H. Tindemans, and Matthijs T. J. Spaan. Safety-constrained reinforcement learning with a distributional safety critic. *Machine Learning*, pages 1–29, 2022.

Ting Yang, Liyuan Zhao, Wei Li, and Albert Y Zomaya. Reinforcement learning in sustainable energy and electric systems: a survey. *Annu. Rev. Control.*, 49:145–163, 2020a.

Tsung-Yen Yang, Justinian Rosca, Karthik Narasimhan, and Peter J Ramadge. Projection-Based Constrained Policy Optimization. In *Proceedings of the 8th International Conference on Learning Representations*, pages 1–11. OpenReview.net, 2020b.

Kok-Lim Alvin Yau, Junaid Qadir, Hooi Ling Khoo, Mee Hong Ling, and Peter Komisarczuk. A Survey on Reinforcement Learning Models and Algorithms for Traffic Signal Control. *ACM Comput. Surv.*, 50(34):1–38, 2017.

Haeun Yoo, Ha Eun Byun, Dongho Han, and Jay H Lee. Reinforcement learning for batch process control: Review and perspectives. *Annual Reviews in Control*, 52:108–119, 2021.

Håkan L S Younes and Michael L Littman. PPDDL 1.0: An Extension to PDDL for Expressing Planning Domains with Probabilistic Effects, October 2004. URL `http://reports-archive.adm.cs.cmu.edu/anon/2004/CMU-CS-04-167.pdf`.

Andrea Zanette. Exponential Lower Bounds for Batch Reinforcement Learning: Batch RL can be Exponentially Harder than Online RL. In *Proceedings of the 38th International Conference on Machine Learning*, pages 12287–12297. PMLR, 2021.

Ruohan Zhang, Faraz Torabi, Lin Guan, Dana H Ballard, and Peter Stone. Leveraging Human Guidance for Deep Reinforcement Learning Tasks. In *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence*, pages 6339–6346. ijcai.org, 2019.

Shun Zhang, Edmund H Durfee, and Satinder P Singh. Minimax-Regret Querying on Side Effects for Safe Optimality in Factored Markov Decision Processes. In *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence*, pages 4867–4873. ijcai.org, 2018.

Wenshuai Zhao, Jorge Peña Queralta, and Tomi Westerlund. Sim-to-Real Transfer in Deep Reinforcement Learning for Robotics: a Survey. In *IEEE Symposium Series on Computational Intelligence*, pages 737–744. IEEE, 2020.

Liyuan Zheng and Lillian Ratliff. Constrained Upper Confidence Reinforcement Learning. In *Proceedings of the 2nd Conference on Learning for Dynamics and Control*, pages 620–629. PMLR, 2020.

ACKNOWLEDGEMENTS

This work would not have been possible without the support of many people. Therefore, I would like to express my gratitude to them.

First and foremost, I would like to thank my daily supervisor Matthijs Spaan, who has guided me during this journey. After I completed my master's degree in Brazil, he believed in my potential and gave me the opportunity to pursue a Ph.D. in the Netherlands. Matthijs was always extremely positive and generous. Meeting with him always gave me calmness, confidence, and focus. He introduced me to multiple researchers, which led to long-lasting collaborations. He also always incentivized me to embrace different opportunities. The day I arrived in the Netherlands, he lent me his old bike, which helped me immediately feel like a local. This was just the first demonstration of his generosity, which has never stopped.

I would also like to thank my copromotor Rob Stikkelman. In our meetings, Rob constantly challenged me to explain my work better. Over the years, this has been very helpful, allowing me to present my research more clearly. Our discussions have also helped me keep my research grounded.

Thanks to all the members of my defense committee, Robert Babuška, Bart De Schutter, Frans Oliehoek, Aske Plaat, and Marek Petrik, for their interest in this work and the insightful questions.

During this journey, I also had the pleasure of having multiple collaborators. In particular, I would like to thank Romain Laroche and Rémi Tachet des Combes for showing me a bit of the research life outside of the academic environment during my internship at Microsoft Research Montréal. I would also like to thank Nils Jansen for giving me excellent feedback during our collaboration and, later, the opportunity to work in his group, where I have been happily collaborating with Christoph Schmidl, Dennis Groß, Marnix Suilen, Merlijn Krale, Thom Badings, and Yannick Hogewind. Finally, thanks to my MSc thesis supervisor Leliane Nunes de Barros, for introducing me to the exciting field of decision-making under uncertainty.

The Algorithmics group has always been a friendly, inclusive, and welcoming workplace. I want to thank Cees Witteveen, Mathijs de Weerdt, and Neil Yorke-Smith for building this atmosphere and the new faculty members Wendelin Böhmer, Anna Lukina, Emir Demirović, and Sebastijan Dumančić for maintaining it. I would also like to thank Sophie den Hartog, Kim Roos, and Shémara van der Zwet for their invaluable support.

Thanks to Canmanie, Greg, Qisong, Natalia, Yang, Lei, Longjian, Koos, Anna, Laurens, Jesse, Stefan, and Ivo for our regular discussions over lunch and after work. Meeting with such a wonderful group was an excellent reason to go to work since interacting with them was like the sunshine I needed during the dark days of the dutch winter. A special thanks to Erwin and Frits for showing me around the office and inspiring me to finish this dissertation. I am also glad to see the new members of the Algorithmics group, Junhan, Moritz, Pascal, Noah, Joery, Eghonghon, Ksenija, Grigorii, and Oussama, bringing new energy to the group and keeping this friendly working space.

I am lucky to have made good friends in the Netherlands and in Brazil. To Carmel, Ashley, Taraneh, Yuri, Ebi, Robson, Ignasi, Kelvin, Jandson, Sabryna, João, Flavia, and César, I would like to say thanks for all the good moments in person and virtually. These moments have helped me unwind and take my head out of work. Moving to the Netherlands, I was also lucky to find the most awesome roommates I could only dream of in Delfgauw. Thanks to Leo, Lissi, Trisha, and Michael for receiving me so well in a new country and making me feel at home. And thanks to Edo, Charlie, Jonny, and Adrián for making this home even more fun.

As for my family, I would like to address them in Portuguese now. Muito obrigado a toda minha família pelo amor e suporte que me deram durante toda a minha vida, mesmo nos vendo pessoalmente apenas uma vez por ano durante esse período eu sempre senti seu afeto. Agradeço à tia Ângela e à tia Bene por me incentivarem a continuar estudando. Aos meus pais Eliza e Acácio e meus irmãos Carla e Luis Gustavo obrigado por sempre estarem ao meu lado. Finalmente, obrigado Mari por me inspirar diariamente, a forma como você se adaptou a nossa nova vida foi incrível. Não tenho palavras para descrever o quanto você me faz feliz. Vamos seguir juntos nessa nossa jornada.

List of Publications

Related to This Thesis

Thiago D Simão and Matthijs T J Spaan. Safe Policy Improvement with Baseline Bootstrapping in Factored Environment. In *Proceedings of the Thirty-Third AAAI Conference on Artificial Intelligence*, pages 4967–4974. AAAI Press, 2019.

Thiago D Simão and Matthijs T J Spaan. Structure Learning for Safe Policy Improvement. In *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence*, pages 3453–3459. ijcai.org, 2019.

Thiago D Simão. Safe and Sample-Efficient Reinforcement Learning Algorithms for Factored Environments. In *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence*, pages 6460–6461. ijcai.org, 2019.

Thiago D Simão, Romain Laroche, and Rémi Tachet des Combes. Safe Policy Improvement with an Estimated Baseline Policy. In *Proceedings of the 19th International Conference on Autonomous Agents and MultiAgent Systems*, page 1269–1277. IFAAMAS, 2020.

Thiago D Simão, Nils Jansen, and Matthijs T J Spaan. AlwaysSafe: Reinforcement Learning Without Safety Constraint Violations During Training. In *Proceedings of the 20th International Conference on Autonomous Agents and MultiAgent Systems*, pages 1226–1235. IFAAMAS, 2021.

Other Peer-Reviewed Publications

Ignasi Andrés, Leliane Nunes de Barros, Denis D. Mauá, and Thiago D. Simão. When a Robot Reaches Out for Human Help. In *Advances in Artificial Intelligence - IBERAMIA*, pages 277–289. Springer, 2018.

Qisong Yang, Thiago D Simão, Simon H Tindemans, and Matthijs T J Spaan. WCSAC: Worst-Case Soft Actor Critic for Safety-Constrained Reinforcement Learning. In *Proceedings of the Thirty-Fifth AAAI Conference on Artificial Intelligence*, pages 10639–10646. AAAI Press, 2021.

Qisong Yang, Thiago D. Simão, Simon H. Tindemans, and Matthijs T. J. Spaan. Safety-constrained reinforcement learning with a distributional safety critic. *Machine Learning*, pages 1–29, 2022.

Danial Kamran, Thiago D. Simão, Qisong Yang, Canmanie T. Ponnambalam, Johannes Fischer, Matthijs T. J. Spaan, and Martin Lauer. A Modern Perspective on Safe Automated Driving for Different Traffic Dynamics Using Constrained Reinforcement Learning. In *25th IEEE International Conference on Intelligent Transportation Systems (ITSC)*, pages 4017–4023. IEEE, 2022.

Marnix Suilen, Thiago D. Simão, David Parker, and Nils Jansen. Robust Anytime Learning of Markov Decision Processes. In *Advances in Neural Information Processing Systems*, volume 35. Curran Associates, Inc., 2022.

Thiago D. Simão, Marnix Suilen, and Nils Jansen. Safe Policy Improvement for POMDPs via Finite-State Controllers. In *Proceedings of the AAAI Conference on Artificial Intelligence*. AAAI Press, 2023.