Advances in Risk-Aware Offline Reinforcement Learning: A Study of Data Augmentation, Explainability, and Policy Selection

Giorgio Angelotti

Université Fédérale

Toulouse Midi-Pyrénées

THÈSE

En vue de l'obtention du

DOCTORAT DE L'UNIVERSITÉ DE TOULOUSE

Délivré par :

Institut Supérieur de l'Aéronautique et de l'Espace

Présentée et soutenue par :
Giorgio ANGELOTTI

le lundi 12 juin 2023

Titre :

Advances in risk-aware offline reinforcement learning :
A study of data augmentation, explainability, and policy selection

---

École doctorale et discipline ou spécialité :

EDSYS : Informatique et Robotique

Unité de recherche :

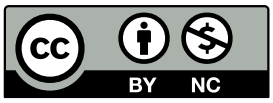
Équipe d'accueil ISAE-ONERA DECISIO

Directeur(s) de Thèse :

Mme Caroline PONZONI CARVALHO CHANEL (directrice de thèse)

Jury :

M. Régis SABBADIN Directeur de recherche MIAT INRAE Toulouse - Président
M. Nicolas DROUGARD Professeur Associé ISAE-SUPAERO - Co-encadrant de thèse
M. Rémi MUNOS Directeur de recherche INRIA Lille - Examinateur
MI. Emmanuel RACHELSON Professeur ISAE-SUPAERO - Examinateur
MI. Vincent THOMAS Maître de conférences Université de Lorraine - Examinateur
M. Marek PETRIK Associate Professor University of New Hampshire - Rapporteur
Mme Caroline PONZONI CARVALHO CHANEL Professeure Associée ISAE-SUPAERO - Directrice de thèse
M. Bruno ZANUTTINI Professeur Université de Caen Normandie - Rapporteur



Ad ubique caritatem diffundendum

Contents

Acknowledgements

In a scientific manuscript, this is the only page that allows for a departure from the scientific style. When writing, reading, or commenting on a paper, it's easy to be swept away by the narrative's power to decontextualize, and forget that behind each work, there are real people. Hence, as one among many representatives of such a humble class, those of humans, I deem it necessary to express my gratitude to all the imperfect and beautiful beings, entities, and places that God placed upon my path. Without you, this three-year-and-a-half-lasting journey would have been different. Maybe better. But probably not! Who can tell?

In primis, grazie a Nelson e grazie alla mia famiglia per essere stati ed essere sempre porto sicuro e sostegno immancabile.

Merci à Caroline et Nicolas. Au-delà de vos connaissances, vous m'avez transmis la méthodologie et la rigueur scientifique. Mais, plus important, vous m'avez enseigné à être patient et à croire dans mes propres intuitions jusqu'au bout, sans oublier la flexibilité et le côté humain. Si jamais j'ai un poste de responsabilité, je vise à être aussi compréhensif que vous.

Thanks to Ryanair and to all the low-cost flight companies that connected the places I could visit and live in, that now are closer to my heart. I love you, although we mutually scammed each other.

Merci à la ville de Toulouse, petit bijou du Sud-Ouest. En flânant dans tes ruelles, j'ai appris à apprécier la vraie France. Gràcies Barcelona per mostrar-me una forma de vida diferent tant a la italiana com a la francesa. Nua i sense guiris ets encara més elegant! Grazie Napoli, bella e terribile.

Merci à mes colocs du 32 rue de Metz: Boris, Charlène, Elsa, Gonzalo, Karine et Tigran. Pour le meilleur ou pour le pire, vous avez été une deuxième famille. Gracias a mis compis del 170 carrer de Casanova, Quim y (sant) Ander por haber compartido conmigo muchas experiencias Mike-osas.

Grazie ai miei amici napoletani: Belmonte, Marino, Edu, Matteo, Carmen, Peppe, Dario D.L., Dario B., Lanfranco, Catu, Marco, Kekkino, Riccardo, Mattia, Ida, Fabrizio, Manuel, Andrea e Celia. E come non rigraziare anche Stefano, Gigetti, Pali e Brenda, con cui ho condiviso una fetta di vita castiza.

Merci à mes potes toulousains: Ainhoa, Andi, Giovanni, Mahmoud, Sahar (Yoshra), Elisabet, Giorgio T. et Friedrich. On a partagé des expériences intenses. Merci aux gens de l'ISAE Supaero: Thibault, Fred, Corentin, Jérôme L., Emmanuel R., Valérie B., Dennis W., Xiaoqi et aux doctorants du DCAS. Un merci particulier à Adam et Chris pour leur aide avec les manips. Merci à Corinne J. d'ANITI. Merci également à tous les membres de mon comité de suivi de thèse pour leurs précieux conseils. Gracias a Natalia D.R. por su entusiasmo y apoyo a lo largo de nuestra colaboración.

Gracias a mis socios de Barcelonas, los reencontrados y los perdidos: Niccolò, Meret, Marieta, Giulia, el Conde, la ICFO mafia y la random peña de Erasmus. Fue guay pasar Covid juntos. Gracias a Hector Geffner, Gergely Neu, Anders Jonsson e Yuliya por darme la bienvenida a la Pompeu Fabra.

Thanks to all the friends I met or rejoined within the most disparate places: Laura dall'Ecuador; the Poles Karolina, Kuba, Julia, and Mariusz; the Germans Sophie and Valérie M.L. (pour le vélo); the Spanish Inés, Lucía, and Emma Cerveza; and the ones from my master's and Cité U. that I keep meeting over and over again Leo, Bea, Chantal, Anto, Florian, Martina, Ale, Tommy, Hélène...

Merci aux infirmiers, secouristes et médecins du secours en montagne et du CHIVA de Foix pour m'avoir soigné alors que le destin a décidé de mettre à l'épreuve ma détermination.

In conclusion, I'd like to express my sincere appreciation to all those I forgot. I am sure you are many. I will invite you for a drink or a coffee! Your contributions, big or small, have shaped this journey in countless ways, and I am grateful for each and every one of them.

List of works

1 Giorgio Angelotti, Nicolas Drougard and Caroline Ponzoni Carvalho Chanel (2020). 'Offline Learning for Planning: A Summary'. In: *Proceedings of the 1st Workshop on Bridging the Gap Between AI Planning and Reinforcement Learning at the 30th International Conference on Automated Planning and Scheduling*, pp. 153–161

2 Giorgio Angelotti, Nicolas Drougard and Caroline Ponzoni Carvalho Chanel (2022). 'Expert-guided Symmetry Detection in Markov Decision Processes'. In: *Proceedings of the 14th International Conference on Agents and Artificial Intelligence - Volume 2: ICAART,*. INSTICC. SciTePress, pp. 88–98

3 Giorgio Angelotti, Nicolas Drougard and Caroline Ponzoni Carvalho Chanel (2023b). 'Data Augmentation Through Expert-Guided Symmetry Detection to Improve Performance in Offline Reinforcement Learning'. In: *Proceedings of the 15th International Conference on Agents and Artificial Intelligence - Volume 2: ICAART,*. INSTICC. SciTePress, pp. 115–124

4 Giorgio Angelotti, Nicolas Drougard and Caroline Ponzoni Carvalho Chanel (2023a). 'An Offline Risk-aware Policy Selection Method for Bayesian Markov Decision Processes'. arXiv:2105.13431

5 Giorgio Angelotti and Natalia Díaz-Rodríguez (2023). 'Towards a more efficient computation of individual attribute and policy contribution for post-hoc explanation of cooperative multi-agent systems using Myerson values'. In: *Knowledge-Based Systems* 260, p. 110189

Acronyms

A3C Asynchronous Advantage Actor Critic

AI Artificial Intelligence

ALE Arcade Learning Environment

ANITI Artificial and Natural Intelligence Toulouse Institute

AOI Area of Interest

BCQ Batch Constrained Deep Q-Learning

BEAR Bootstrapping Error Accumulation Reduction

BMDP Bayesian MDP

CQL Conservative Q-Learning

CVaR Conditional Value at Risk

D4RL Datasets for Deep Data-Driven Reinforcement Learning

DDPG Deep Deterministic Policy Gradient

DNN Deep Neural Network

DPG Deterministic Policy Gradient

DQN Deep Q-Network

ECG Electrocardiogram

EM Expectation-maximization

ET Eye Tracker

EvC Exploitation vs Caution

FQE Fitted Q Evaluation

FVI Fitted Value Iteration

GUI Graphical User Interface

HMM Hidden Markov Model

HR Heart Rate

HRI Human-Robot Interaction

HRV Heart Rate Variability

HSVI Heuristic Search Value Iteration

IQN Implicit Quantile Network

ISAE Institut Supérieur de l'Aéronautique et de l'Espace

KSS Karolinska Sleepiness Scale

LIME Local Interpretable Model-Agnostic Explanations

LSL Lab Streaming Layer

MAS Multi-Agent System

MC Markov Chain

MC2PS Monte Carlo Confident Policy Selection

MCTS Monte Carlo tree search

MDP Markov Decision Process

MII Mixed-Initiative Interaction

ML Machine Learning

MOPO Model-based Offline Policy Optimization

MOReL Model-Based Offline Reinforcement Learning

MPC Model Predictive Control

MSE Mean Squared Error

MuJoCo Multi-Joint dynamics with Contact

NASA-TLX NASA Task Load Index

OPE Off-policy Evaluation

PAC Probably Approximately Correct

PBVI Point-Based Value Iteration

pdf Probability Density Function

PILCO Probabilistic Inference for Learning COntrol

PlaNet Deep Planning Network

pmf Probability Mass Function

POMCP Partially Observable MonteCarlo Planning

POMDP Partially Observable Markov Decision Process

PPO Proximal Policy Optimization

PSR Predictive State Representation

PWLC Piece-Wise Linear and Convex

QR Quantile Regression

QTD Quantile Temporal-Difference

RDC Randomized Dependence Coefficient

REM Random Ensemble Mixture

RL Reinforcement Learning

RNN Recurrent Neural Network

SARSA State-Action-Reward-State-Action

SARSOP Successive Approximations of the Reachable Space under Optimal Policies

SHAP SHapley Additive exPlanations

SimPLe Simulated Policy Learning

TD Temporal Difference

TD3 Twin Delayed Deep Deterministic policy gradient

TRPO Trust Region Policy Optimization

UCT Upper Confidence Bound 1 applied to trees

UnO Universal Off-Policy Evaluation

VaR Value at Risk

XAI Explainable Artificial Intelligence

XRL Explainable Reinforcement Learning

Advances in Risk-Aware Offline Reinforcement Learning: A Study of Data Augmentation, Explainability, and Policy Selection

Giorgio Angelotti

Abstract

In the field of Offline Reinforcement Learning, the goal is to learn a decision policy offline based on a previously collected batch of experiences and without additional interaction in a data-efficient and risk-sensitive manner. This dissertation presents several techniques for achieving this goal, with a focus on model-based methods: paradigms that first infer a behavioral model for the sequential decision-making problem and subsequently solve it by taking into account model uncertainty. The presented contributions include a method for augmenting a dataset of samples through detecting symmetries in the system dynamics, an algorithm for performing offline risk-sensitive policy selection called Exploitation vs Caution (EvC) resorting to the Bayesian Markov Decision Process framework, and a paradigm for explainability in multi-agent cooperative systems using Myerson analysis. Additionally, perspectives are discussed for applying the EvC approach to obtaining an adaptive interaction control policy in a human-robot scenario. Indeed, taking proper precautions, we adapted the EvC algorithm for risk-sensitive policy selection to be applied to the ISAE Firefighter Robot Game, which involves the optimization of adaptive policies to control the interaction between a firefighter robot and a human operator in a proof-of-concept scenario. Overall, the contributions of this thesis demonstrate the potential for the presented techniques to significantly improve the performance of Offline Reinforcement Learning algorithms and to be applied in a variety of real-world settings, including Human-Robot Interaction.

Introduction

The integration of robots and artificial intelligence into society has the potential to enhance the efficiency and effectiveness of various tasks while alleviating the burden of manual labor on humans. However, the path toward full automation is complex and requires the development of robust and reliable technologies that can operate safely and ethically in a variety of environments.

The diffusion of technological advancements requires them to comply with the regulations of the State where they are implemented. The European Commission (2020) released a White Paper on the subject, listing the guidelines that an *Artificial Intelligence (AI)* producer or service provider should follow with the aim of aligning not only with the laws of the single states of the European Union but also to be ready for what will be, in the future, a common international regulation.
The main concerns raised by the European Commission regarded risk assessment, safety, and liability of the deployment of AI-based technologies. Any citizen of the European Union should feel as safe in the presence of an automated agent as in the presence of a fellow human. Moreover, AI-based systems and hence the automated agents should abide by some quantifiable guarantees about their performances and the collateral damage in case of accidents, which should be minimal. Another issue from a legal perspective is the tracing back of responsibilities: in complex situations, the opaque nature of recent AI-based techniques, which can often be seen as a black box, hampers the distinction between incidents caused due to a malfunctioning system from the ones due to a deficit in the design of such a system. The European Commission, among many, advises abiding by the following guiding principles:

(1) Development of robust, risk-sensitive, and accurate technologies;

(2) Strive to provide clear explanations about the outputs of the used AI-based technologies, specifically detailing the reasons behind an automated agent's actions;

(3) Enforcement of human oversight, that can be enacted by (a) granting the right to the human to validate and deny the decision of the automated agent, (b) allowing the human operator to take full control of the system and to shut down the machine, (c) imposing *in the design phase* some operational constraints to the full autonomy of the automated agent in specific situations.

Automated systems with human oversight

The state of automation implemented in everyday life is often supervised by a human operator: *e.g.* autopilots for aircraft, autonomous vehicles, and robots in assembly lines. This complies with the principles (3.a,b) of the previous list. Full autonomy is permitted only when the chance of accidents is minimal. But how can an automated agent determine which action to take? This problem is called *decision-making*. An agent can enact simple decisions, *e.g.* an immediate preference; or "complex" decisions, *e.g.* based on some sort of preventive reasoning with the aim of reaching a

long-term goal. The process that yields the rules to deploy the said complex decisions is called *planning*. Let us explore this concept in greater detail.

Planning The book by Ghallab, Nau and Traverso (2004) defines planning as the *process of choosing and organizing actions while anticipating their effects to change the state of a system.* This necessitates the definition of what is known as a planning model. The selection of a planning model for an automated agent is influenced by various factors such as the nature of the environment, the types of actions, and the observability of the state of the system. Usually, the more realistic the assumptions of a planning model, the more complex the resulting planning problem will be. In this thesis, we will specifically focus on two classes of planning models: Markov Decision Process (MDP) and Partially Observable Markov Decision Process (POMDP).

Stochastic decision processes MDPs and POMDPs are built on the theory of Stochastic Processes. In this sense, the possible time evolution of a system is obtained through a transition function, *i.e.* a distribution that dictates the probability for the system to transit from one state to another given a deployed action. While MDPs solvers respect Bellman's Optimality Principle and should find a deterministic optimal behavioral policy, a POMDP is undecidable in infinite horizon settings (Madani, Hanks and Condon, 1999). Nevertheless, approximate solutions (sub-optimal policies) can be found using modern solvers like PBVI, SARSOP and POMCP (Pineau, Geoff Gordon, Thrun et al., 2003; Kurniawati, Hsu and W. S. Lee, 2008; Silver and Veness, 2010). We will dive deep into the technicalities of (PO)MDP solving in Chapter 1.

Reinforcement Learning Unfortunately, an effective representation of the world in terms of a stochastic process is not always available. Over the years, the Artificial Intelligence community has developed various approaches to enable planning in the absence of complete information. The said methods nowadays all fall clumsily under the big umbrella of *Reinforcement Learning (RL)* (Sutton and Andrew G. Barto, 2018), although in many cases the learning is not properly happening by *reinforcement*. In traditional online reinforcement learning, the agent learns by interacting with the environment online and adjusting its actions based on the immediate feedback it receives. However, this approach has several limitations and challenges, including the need for large amounts of data and the potential for the agent to get stuck in local maxima or to make suboptimal decisions that could lead to catastrophic consequences. RL methods methods experienced a significant performance boost when Mnih, Kavukcuoglu et al. (2015) combined Q-Learning (Watkins and Dayan, 1992) with a Deep Neural Network (DNN). This fusion allowed an automated agent to control a system with continuous state variables, achieving human-level performance. Since that breakthrough, excitement within the community has grown, leading to the publication of hundreds of papers featuring increasingly sophisticated algorithms, such as those by Schulman, Wolski et al. (2017), Silver, Hubert et al. (2018) and Marc G. Bellemare, Dabney and Rowland (2023). We will outline RL and Deep RL in Section 1.4.

Although Deep Reinforcement Learning algorithms have achieved tremendous success in achieving super-human performance in various board games and video games, their application in optimizing real-world tasks remains limited to sectors where potential collateral damage is minimal or, regrettably, deemed negligible, *e.g.* controlling the trajectory of stratospheric balloons over the Pacific Ocean (Marc G. Bellemare, Candido et al., 2020), optimizing a video codec for streaming websites (Mandhane et al., 2022).

Automated agents can optimally or sub-optimally execute the tasks they were programmed to handle. However, a mission can quickly become unmanageable in the event of unforeseen occur-

rences or situations that the machine was not hardcoded (or a-priori modeled) to cope with. What automated agents are lacking is the generalization capability that is necessary for automated agents to reach human-level intelligence (Geffner, 2018). Furthermore, it should be noted that modern Deep RL techniques addressing problems with multidimensional states often benefit from structured representations, such as spatial distances of pixels in images. In contrast, real-life robotic problems involve observations characterized by multidimensional and multimodal measurements from various sensors. Efforts have been made to develop *general agents* capable of performing tasks like playing video games and text generation simultaneously (Reed et al., 2022). While the results are encouraging, they are not yet dependable or prepared for real-life deployment. Consequently, human oversight remains a recommended safeguard.

Offline Reinforcement Learning

Offline Reinforcement Learning is a variant of traditional RL that allows agents to learn from previously collected data, rather than requiring online interaction with the environment (Levine et al., 2020). This approach has several advantages, including the possibility to learn from rare or potentially dangerous situations that may be difficult to simulate or undesirable to reach in the real world. Nevertheless, Offline Reinforcement Learning is affected by a plethora of nontrivial limitations. For instance, obtaining a high-performing control policy necessitates a diverse and extensive initial dataset, as further exploration is not permitted. This condition is often unmet in real-world applications involving human interaction since collecting a large dataset can be not only costly but also hazardous and time-consuming. Moreover, a policy derived from an offline reinforcement learning method can be not only suboptimal but also risky if the initial dataset of trajectories does not adequately represent the system's dynamics. Bearing this in mind, it is coveted to take precautions: the idea is to develop methods that optimize risk-sensitive objectives, ensuring safe and reliable results. In this thesis, we discuss and propose advances in Offline Reinforcement Learning starting from a small batch of experiences. Then, our methods are confronted with a Human-Robot Interaction (HRI) use case. The aim is to develop interaction control policies that enhance the performance of human-robot teams.

Human-robot systems

In recent years, the field of human-robot systems has seen a growing interest in adaptive human-robot interaction. The primary motivation behind this shift is the increasing acknowledgment that traditional, pre-programmed interaction strategies are insufficient to meet the diverse and dynamic needs of human users in real-world settings.

A key driver of this shift is the increasing availability of sophisticated sensors and actuators, enabling robots to perceive and respond to their environment in real time. This has led to the development of new control architectures, such as behavior-based and hybrid systems, which allow robots to transition between various interaction strategies based on the current context (Schulz, Kratzer and Toussaint, 2018).

Another significant factor is the expanding body of research on human cognition and behavior, providing insights into the elements that influence human-robot interaction and guiding the design of robots to better align with human expectations and preferences (Nomura et al., 2008). An example of this is the use of shared autonomy (Schilling et al., 2019), which is defined as "*a concept that describes how all the agents can remain autonomous, following overall their intentions and goals, but at the same time deal with the coordination of activities and resolution of possible*

conflicts".

In addition, there has been a growing interest in employing ML and AI methods to enable robots to learn and adapt to their human users over time. This includes techniques such as RL (Akkaladevi et al., 2018), inverse RL (M. S. Lee, Admoni and Simmons, 2021), probabilistic models (Jain and Argall, 2019) and stochastic sequential decision processes (Javdani, Srinivasa and Bagnell, 2015), which allow robots to learn from human feedback and optimize their behavior based on user preferences.

One type of shared autonomy is that of Mixed-initiative Human-Robot Interaction. Mixed-initiative control systems for human-robot interaction (Hearst et al., 1999) can be designed to adhere to the principle of limiting automation in specific situations, allowing the transfer of control to a human operator. Although humans can serve as fail-safe agents, they are not immune to making errors. To address this concern, mixed-initiative human-robot systems can be developed, in which both the human operator and the automated agent are treated as equals and can be assigned full control of a mission or subtasks. In this context, the human and machine form a multi-agent system, with their different characteristics being leveraged to achieve a shared goal. The roles of each operator can be opportunistically assigned based on the situation, rather than being predetermined (Caroline P. C. Chanel et al., 2020).

Collaboration with humans is crucial because they possess strong generalization and improvisation skills and can rapidly adapt to unfamiliar situations by drawing analogies with past experiences. Nonetheless, the solutions they find might not be optimal, and the human's current mental and physical state can significantly impact their decision-making capacity (Dehais et al., 2020).

To design an optimal control system for mixed-initiative human-robot interaction, it is essential to have a model of the human-robot system which itself includes a model of human behavior. Nevertheless, a general mathematical model that can accurately predict human behavior does not currently exist, as humans are biologically diverse and can exhibit varying behavior in different contexts based on factors such as attitude, background, education, age, gender, and current physical and mental condition. One approach to modeling human behavior involves inferring the probabilities of specific actions in a given context using a data-driven approach, such as fitting a stochastic process to discrete-time data (Pentland and A. Liu, 1999). For these reasons, POMDPs have been utilized as a framework to model human-robot systems with adaptive interaction (Nikolaidis, Ramakrishnan et al., 2015; Nikolaidis, Zhu et al., 2017). However, these models may not be fully generalizable to all humans.

When designing mixed-initiative systems, it is important to consider both human and machine capabilities and ensure that the roles of each agent are optimally distributed based on the situation. For instance, tasks requiring improvisation or generalization might be better suited for human operators, while tasks necessitating precise and repeatable actions may be more appropriate for automation.

Challenges

The theme dealt with in this thesis is far from trivial since the performance of an Offline Reinforcement Learning algorithm is significantly affected by the *"quality"* of the starting data set. This concept is rather vague, as the optimal amount of data required for obtaining a reliable control policy depends not only on the specific task but also on the objective of the optimization problem. For example, should we optimize the expected cumulative reward, for its worst-case scenario, or

some other distributional measure? Moreover, when transferring some of the proposed methods to the context of HRI and human behavior, we grappled with the difficulties of modeling a highly complex system. Eventually, to deliver a final *robust* and *interpretable* mixed-initiative interaction controller that is also *deployable* in a real-world scenario, we had to face several challenges and make design decisions of differing natures.

Here we summarize the main hardships encountered and the research questions that emerged in this thesis work.

Offline Reinforcement Learning

1. Offline statistical inference (offline learning):
   (a) do we have enough data to learn a *"good"* model?
   (b) is it possible to leverage some inherent structure in the dynamics to enable more data-efficient learning?
2. Robust and risk-aware planning under uncertainty:
   (a) how to quantify and take into account *model uncertainty*, *i.e.* the fact that the learned model could not be representative of the real-world dynamics or sufficiently general?
   (b) how to assess the *risk* of deploying a *"bad"* policy?
   (c) how to obtain a *robust* policy?
   (d) how to select the less risky or the most performing policy between many obtained with different approaches?

Real-world use case involving the application to Human-Robot Interaction

1. Representation:
   (a) is the POMDP the most suitable mathematical framework to model a Mixed-Initiative Control System for Human-Robot Interaction?
   (b) *what* are the hidden states? Are they finite? Countable? Can they be linked to the human's mental state or are they just a mathematical artifice?
   (c) can the POMDP representation (number of hidden states) be automatically learned from data, or assumptions must be made and an expert should set their number (or shape) depending on the context and his/her intuition?
   (d) can we learn and solve the most general representation of a POMDP, or *dimensionality reduction* should be executed to complete the planning task?
2. Offline statistical inference (offline learning):
   (a) is it possible to infer the transition and observation probabilities of any POMDP from data, or is this limited to a specific subset of POMDPs (*e.g.* finite states, countable POMDPs)?
   (b) what are the algorithms or approaches to *learn* a POMDP? Are they stable and reliable?
   (c) since any human being is unique, can the model we are learning be general or is it only specific to the individual whose records were included in the data set?

Explainability of Multi-Agent Systems

1. Explaining why the model took a specific action, hence lightening *black-box opacity* of AI systems;

2. Assessing the individual contribution to the team performance in cooperative Multi-Agent System (MAS).

Each of the listed topics and subtopics presents a research challenge deserving attention from several communities, as most of these themes lie at the frontier of science and are tremendously multidisciplinary, intersecting fields such as *mathematics*, *computer science*, *statistics*, *engineering*, *physiology*, and *neuroscience*.

Contributions

The principal finding of this thesis work is the development of machine learning tools to enhance real-world deployable Offline Reinforcement Learning methods, with particular attention to HRI use cases, or more in general, to human-automated agents interaction use cases.

In addition to developing Offline Reinforcement Learning methods, we aim to transfer this knowledge to a real-world application case by obtaining an adaptive controller for a mixed-initiative human-robot system. While adhering to a *hand-designed* model, we conceive an ad-hoc schema to map the live multidimensional observations of the state of the system to those of a small discrete (PO)MDP. Subsequently, we propose a detailed Bayesian method to incorporate *model uncertainty*, allowing us to learn the transition function from the data. Ultimately, we solve the model in a risk-sensitive fashion, obtaining a robust policy in line with point (1) of the European Commission's previously mentioned guidelines.

In detail, the contributions of this thesis are:

- Regarding **Offline Reinforcement Learning** in general:

  1. To address the research questions "do we have enough data to learn a *good* model?" and "is it possible to leverage some inherent structure in the dynamics to enable more data-efficient learning?", an approach is proposed to detect and exploit symmetries in the dynamics of a Markov Decision Process. This method relies on expert knowledge and statistical estimates to augment the batch used to offline learn a Markov Decision Process with symmetrical samples. As a result, the derived policy demonstrates improved performance, and the learning process becomes more data-efficient.

  2. To address the research questions "how to quantify and to take into account model uncertainty?", "how to assess the risk of deploying a *bad* policy?" "how to obtain a robust policy?", "how to select the less risky or the most performing policy between many obtained with different approaches?", a method is proposed to evaluate and select policies (and hyperparameters) offline for an offline learned Markov Decision Process. Risk-sensitive and robust policies obtained offline (using a fixed data set) are the ones that better reduce the chance of aftermaths at the time of real-world execution.

- As mentioned in the Introduction, we believe that for both performance and legal concerns (European Commission, 2020), autonomous agents are more likely to be widely adopted within the framework of Mixed-Initiative HRI. Consequently, we transfer our contribution on

offline policy selection to a real-world Human-Robot Interaction control problem use case, represented as a POMDP. In particular, we tackled the problem of the Firefighter Robot (Drougard et al., 2017). During this work, we encountered the challenges listed in paragraph **Real-world use case involving the application to Human-Robot Interaction**. In this particular context, we had to account for the unpredictable actions of a human agent as a contributing factor to the stochastic nature of the environment;

- To address the research issues about **Explainability of Multi-Agent Systems**, a paradigm is proposed for evaluating both agents' policies and individual attributes in a cooperative Multi-Agent System, assessing their contributions to a common goal.

Overview of the manuscript

The corpus of the dissertation is divided into two parts.

Part I focuses on the state-of-the-art. Due to the highly interdisciplinary nature of the subjects addressed, the state-of-the-art will be discussed across multiple chapters for better narrative flow. In **Chapter 1** we introduce the mathematical framework of MDPs and POMDPs, also showing how to solve a sequential decision-making problem expressed in this form. At the end of the chapter, we dedicate three sections to learning a model from experience, to Reinforcement Learning, and to explainability for (multi-)agent systems. **Chapter 2** is dedicated to offline learning for planning and obtaining robust behavioral policies in the presence of model uncertainty. This chapter explores state-of-the-art Offline Reinforcement Learning methods, including model-based, model-free, and sequence model versions. It addresses the challenges of tuning the algorithms' hyperparameters for offline MDP solving, offline policy evaluation, and selection. Additionally, the chapter discusses strategies for modeling model uncertainty and achieving risk-sensitive solutions. Some of the presented methods can also be applied to POMDPs once the challenges of learning or designing a representation for a hidden state are resolved. Human-Robot Interaction and Mixed-Initiative Systems are introduced in **Chapter 3**. This chapter also presents the HRI use case and summarizes the work completed thus far.

Part II focuses on the contributions proposed by this work. **In Chapter 4** we outline a method to detect alleged symmetries in an MDP in a total offline setting. A Bayesian method to encompass model uncertainty when an MDP is learned from data is proposed in **Chapter 5**, subsequently, the said formalism is leveraged to select a risk-sensitive policy between a set of candidate strategies. In **Chapter 6** some of the tools established in the previous chapters are readapted to first obtain an interpretable model for the human-robot mixed interaction and then to get a robust behavioral adaptive control system to drive the said interaction. In **Chapter 7** we advocate a method to compute not only the contribution of policies in multi-agent systems but also the contribution of agents' features to the final score. Finally, the **Conclusion** provides a summary of the entire work, discussing its limitations and future perspectives.

Part I

State-of-the-art

Chapter 1

Sequential decision-making under uncertainty

"The time to repair the roof is when the sun is shining" once said John F. Kennedy. The former president of the United States was referring to the necessity of taking decisions beforehand in the expectation that something will occur. The act of preparing an intelligent way of behaving to reach a goal is called planning. In layman's terms, planning is an approach for addressing sequential decision-making under uncertainty problems. But finding the right sequence of actions to fulfill a requirement is easier said than done. To obtain a quantitative answer, planning must be performed quantitatively and hence it must be grounded in a logical-mathematical formalism.

Optimal control theory is a form of mathematical optimization that solves the problem of finding the control (a function or a variable) for a dynamical system, optimizing an objective functional over a period of time. Boltyanskij, Gamkrelidze and Pontryagin (1956), Bellman (1957), Kalman (1963) and Smallwood and Sondik (1973) identified a set of necessary and sufficient conditions for the existence of a solution, *i.e.* an optimal control, and proposed a dynamic programming method to find the solution for sequential decision-making in problems with stochastic dynamics in discrete time. The advantages of this family of approaches include the mathematical proof of the existence of a solution and the plasticity of a model to be learned from data, once a representation has been assumed. The disadvantage, however, lies in the non-trivial burden of expressing the time evolution of the environment as a dynamical system or as a discrete-time stochastic process. Once a system is defined, the solution is specific to that configuration and may not generalize to even conceptually similar environments: small changes in the system's definition could render the discovered solution ineffective.

A general framework that allows to represent sequential decision-making under uncertainty problems with stochastic dynamics in discrete time is a Markov Decision Process (MDP). Markov Decision Processes, together with Partially Observable Markov Decision Processes, are at the foundations of this thesis. Indeed, this work focuses on learning the *"right"* model expressed in terms of a (PO)MDP starting from a batch of experiences previously collected by an agent.

Previously collected trajectories or experiences of past events form a potentially multidimensional time series: a collection of sequences of values that vary over time. Any multidimensional time series can be considered the realization of a discrete-time stochastic process.

Definition 1 (Discrete-time stochastic process)**.** *A discrete-time stochastic process S is a collection of random variables with value in $\mathcal{S}$: $S = \{S_t \in \mathcal{S} : t \in \mathbb{N}\}$ and in this case $\mathbb{N}$ serve as a discrete-time*

index. Generally, the probability of a realization of a specific sample is conditional to the realizations of the random variables relative to previous time indices: $\forall t \in \mathbb{N}$, $\forall (s_0, \dots, s_{t+1}) \in \mathcal{S}^{t+2}$,

$$Pr\left(S_{t+1} = s_{t+1} | S_0 = s_0, S_1 = s_1, \dots, S_t = s_t\right). \tag{1.1}$$

Definition 2 (Markov Chain)**.** *A discrete-time Markov Chain (MC) is a discrete-time stochastic process* S *with the Markov property, meaning that the probability of the next realization depends only on the present realization and not on the previous ones:* $\forall t \in \mathbb{N}$, $\forall (s_0, \dots, s_{t+1}) \in \mathcal{S}^{t+2}$,

$$Pr\left(S_{t+1} = s_{t+1} | S_0 = s_0, S_1 = s_1, \dots, S_t = s_t\right) = Pr\left(S_{t+1} = s_{t+1} | S_t = s_t\right). \tag{1.2}$$

Definition 3 (Stationary MC)**.** *If* $Pr\left(S_{t+1} = x | S_t = y\right) = Pr\left(S_t = x | S_{t-1} = y\right)$ $\forall t \in \mathbb{N}^*$, $\forall (x, y) \in \mathcal{S}^2$*, namely if the transition probability is time independent, the MC is said stationary.*

Given their suitability for learning dynamics from this type of data, this dissertation focuses on optimal control frameworks that manage the control of a discrete-time stochastic process adhering to the *Markov property*: Markov Decision Processes and their partially observable counterparts, Partially Observable Markov Decision Processes.

1.1 An introduction to MDPs

If the stochastic dynamics of the system can sequentially change according to the actions performed by an agent at every time step, a stochastic process becomes a stochastic decision process. The stochastic evolution of a sequential decision-making process under uncertainty can be formalized using the mathematical framework of MDPs (Puterman, 2005).

Definition 4 (Markov Decision Process)**.** *An MDP is formally defined as a tuple* $M = \langle \mathcal{S}, \mathcal{A}, T, R, \gamma \rangle$ *where* $\gamma \in [0, 1)$ *is called the discount factor;* $\mathcal{S}$ *is the domain of a discrete time stochastic process;* $\mathcal{A}$ *is a set of possible actions that an agent can perform at every time step* $t \in \mathbb{N}$*;* $T : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to [0, 1]$ *whose values are equal to* $Pr\left(S_{t+1} = s_{t+1} | A_t = a_t, S_t = s_t\right)$ *and represent the probability of transiting from the state* $s_t \in \mathcal{S}$ *to* $s_{t+1} \in \mathcal{S}$ *after taking action* $a_t \in \mathcal{A}$*;* $R : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ *is called the reward function and* $r_t = R\left(s_t, a_t\right)$ *is the reward that the agent gains after performing action* $a_t \in \mathcal{A}$ *in state* $s_t \in \mathcal{S}$*. A realization of an MDP can be recorded as a trajectory* $h = \{(s_t, a_t, s_{t+1}, r_t), t \geq 0\}$ *which is a succession of single time step state transitions, actions, and rewards.*

An MDP can be seen as an MC whose evolution is affected by the decisions of an agent at every time step (Figure 1.1).

Policies In a general manner, a policy is a function that prescribes actions to execute according to the current state of information. A policy can map states to actions, but it can also be history-dependent or randomized.

Definition 5. *A stochastic stationary policy* $\pi : \mathcal{S} \times \mathcal{A} \to [0, 1]$ *is a function that assigns the probability to take action* a *when in state* s*.*

Definition 6. *A deterministic stationary policy* $\pi_{det} : \mathcal{S} \to \mathcal{A}$ *is a function that to every state* $s \in \mathcal{S}$ *assigns an action* $a \in \mathcal{A}$ *to be taken.*

Definition 7 (Value Function)**.** *Let*

$$V_M^{\pi}(s) \overset{\text{def}}{=} \mathbb{E}_{\substack{A_t \sim \pi \\ S_t \sim T}} \left[\sum_{t=0}^{\infty} \gamma^t R(S_t, A_t) \,\middle|\, S_0 = s \right]. \tag{1.3}$$

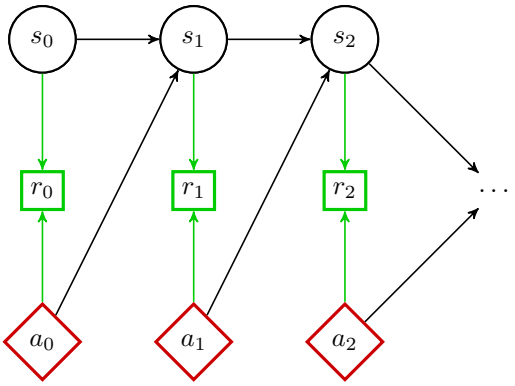


Figure 1.1: The first three time steps of an episode generated by an MDP. We see that s_t and a_t have an impact on r_t through a green arrow ($r_t = R(s_t, a_t)$) and also an impact on s_{t+1} through a black arrow ($S_{t+1} \sim \Pr(\cdot|S_t = s_t, A_t = a_t) = T(s_t, a_t, \cdot)$).

V_M^π is called the value function and assigns to every state $s \in \mathcal{S}$ the expected value over every possible trajectory h that starts at $s_0 = s$ of the cumulative reward geometrically weighted by the discount factor γ where the system dynamics has been dictated by the MC defined by the MDP transition function T and the policy π.

In layman's terms, V_M^π provides the average expected return when following policy π from state s. It represents the cumulative reward we can anticipate by deploying the policy in the given state.

It is worth noting that γ is the discount factor, which scales how much we value immediate rewards over distant ones.

We observe that the sum in (1.3) runs up to ∞. If $\gamma = 1$ this sum can diverge, but $\gamma < 1$ guarantees convergence. However, it is important to emphasize that a planning problem does not always take place in an infinite horizon. Indeed, the sum could also go up to any finite horizon time $H \in \mathbb{N}$.

Definition 8 (Q-value function)**.** *Let $Q_M^\pi : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ be a function that maps every couple of states and actions of an MDP to the average one-step reward that the agent gains by taking action a in state s and then following from that moment on deploying policy π:*

$$Q_M^\pi(s, a) = R(s, a) + \gamma \sum_{s'} T(s, a, s') V_M^\pi(s'). \tag{1.4}$$

In general, the goal of the practitioner is to find the optimal solution to the MDP M: a policy π^* which maximizes V_M^π for the initial state s_0, or even better, $\forall s$. It follows from Eq. (1.4) that for a stochastic policy $V_M^\pi(s) = \sum_a \pi(a|s) Q_M^\pi(s, a)$ while for a deterministic policy $V_M^\pi(s) = Q_M^\pi(s, \pi(s))$. The optimal value function can be obtained from the optimal Q value function: $V_M^*(s) = \max_a Q_M^*(s, a)$. In the next section we explore various methods to solve a discrete MDP.

1.1.1 Solving a discrete MDP

The optimality principle from Bellman (1957) states that *if every policy's quality can be measured by its policy's expected linear additive utility, there exists a policy that is optimal at every time step.* Moreover, following Def. 7, one can prove that a deterministic stationary policy is a solution to a discounted infinite horizon MDP (Mausam and Kolobov, 2012).

Several algorithms solve an MDP and they are all based on Bellman's Equation (Bellman, 1957).

Starting from Eq. (1.3) one can rewrite the Value function as follows (Puterman, 2005):

$$V_M^\pi(s) = \sum_a \pi(a|s) \left(R(s,a) + \gamma \sum_{s'} T(s,a,s') \mathbb{E}_{\substack{A_t \sim \pi \\ S_t \sim T}} \left[\sum_{t=0}^{\infty} \gamma^t R\left(S_t, A_t\right) \middle| \; S_0 = s' \right] \right), \tag{1.5}$$

$$V_M^\pi(s) = \sum_a \pi(a|s) \left(R(s,a) + \gamma \sum_{s'} T(s,a,s') V_M^\pi(s') \right). \tag{1.6}$$

Or, in the case of a deterministic policy:

$$V_M^\pi(s) = R(s,\pi(s)) + \gamma \sum_{s'} T(s,\pi(s),s') V_M^\pi(s'). \tag{1.7}$$

Bellman's Equation Eq. (1.7) can be written in linear form with a linear operator B_M^π which maps V_M^π to $R(s,\pi(s)) + \gamma \sum_{s'} T(s,\pi(s),s') V_M^\pi(s')$:

$$V_M^\pi = B_M^\pi V_M^\pi. \tag{1.8}$$

If a matrix-based representation of the model is feasible, *i.e.* $\mathcal{S}$ and $\mathcal{A}$ are finite and not too big, then the value function V_M^π can be directly computed:

$$V_M^\pi = \left(\mathbb{I} - \gamma T^\pi\right)^{-1} R^\pi, \tag{1.9}$$

with $\mathbb{I}$ being the identity operator, the $^{-1}$ after the parenthesis intended to be an inversion operator, $T_{ss'}^\pi = T\big(s,\pi(s),s'\big)$ and $R_s^\pi = R\big(s,\pi(s)\big)$. It is worth noting that the matrix $(\mathbb{I} - \gamma T^\pi)$ is always invertible for any $\gamma < 1$ (ibid.).

With this in mind, the optimal deterministic policy can be found as

$$\pi^* \in \text{argmax}_\pi \left[\mu_0^T \left(\mathbb{I} - \gamma T^\pi\right)^{-1} R^\pi \right]$$

where μ_0 is the distribution of possible initial states s_0.

However, such a straightforward computation is not always accomplishable since the operator inversion could require too much effort (or computational time) for too big $\mathcal{S}$ or $\mathcal{A}$. Nevertheless, in situations where operator inversion is computationally infeasible within a reasonable time frame, yet a discrete representation is still manageable, the problem can be approached differently. Given that $B^\pi M$ serves as a contraction in the function space $f : \mathcal{S} \to \mathbb{R}$, one can start with any function f and apply BM^π iteratively. Eventually, f will converge to V_M^π. Computing iteratively the value of a specific policy is called **Policy Evaluation**.

The two most famous algorithms to obtain the optimal value function and the optimal policy *by iteration* are, respectively, **Value Iteration** and **Policy Iteration**. These methods leverage Dynamic Programming: starting with an initial arbitrary value assigned to the function to estimate, some mathematical operations are performed accordingly to improve the estimate. The new estimate is used as input for the next iteration to further improve the accuracy until convergence (within a given threshold δ).

In Value Iteration (Algorithm 1) $V_{M,i}^*$ which is initialized with some arbitrary value at iteration $i = 0$. Then $V_{M,i+1}^*(s) = \max_a \left[R(s,a) + \gamma \sum_{s'} T(s,a,s') V_{M,i}^*(s') \right]$. The algorithm stops when

$|V^*_{M,i+1} - V^*_{M,i}| < \delta\frac{1-\gamma}{\gamma}$, yielding the δ-optimal policy (ibid.)

$$\pi^*_{i+1}(s) \in \text{argmax}_\pi \left[R(s,a) + \gamma \sum_{s'} T(s,a,s') V^*_{M,i+1}(s') \right].$$

Algorithm 1: Pseudocode for Value Iteration

Input: MDP M, threshold $\delta \in \mathbb{R}^+$
Output: δ-optimal deterministic policy π^*_i

Initialization: $V^*_{M,0}(s)$ arbitrarily initialized $\forall s \in \mathcal{S}$
$i \leftarrow 0$
$V^*_{M,i+1}(s) \leftarrow \max_a \left[R(s,a) + \gamma \sum_{s'} T(s,a,s') V^*_{M,i}(s')\right]$
while $|V^*_{M,i+1} - V^*_{M,i}| \geq \delta\frac{1-\gamma}{\gamma}$ **do**
    $i \leftarrow i + 1$
    for $s \in \mathcal{S}$ **do**
        $V^*_{M,i+1}(s) \leftarrow \max_a \left[R(s,a) + \gamma \sum_{s'} T(s,a,s') V^*_{M,i}(s')\right]$
for $s \in \mathcal{S}$ **do**
    $\pi^*(s) \leftarrow \text{argmax}_a \left[R(s,a) + \gamma \sum_{s'} T(s,a,s') V^*_{M,i}(s')\right]$
return π^*

In Policy Iteration (Algorithm 2) a deterministic policy π^*_i and $V^*_{M,i}$ are arbitrarily initiated at iteration $i = 0$. Subsequently, Policy Evaluation and Policy Improvement are performed in sequence until it is no longer possible to improve the value function with a different strategy. Policy Evaluation is effectuated by iterating $V^*_{M,i+1}(s) = R(s,\pi^*_i(s)) + \gamma \sum_{s'} T(s,\pi^*_i(s),s') V^*_{M,i}(s')$ until convergence. Policy Improvement just assigns to $\pi^*_{i+1}(s) = \text{argmax}_a \left[R(s,a) + \gamma \sum_{s'} T(s,a,s') V^*_{M,i}(s')\right]$. The algorithm stops when $\forall s \in \mathcal{S}$, $\pi^*_{i+1}(s) = \pi^*_i(s)$. Hence, it yields the δ-optimal policy π^*_{i+1}.

Algorithm 2: Pseudocode for Policy Iteration

Input: MDP M, threshold $\delta \in \mathbb{R}^+$
Output: δ-optimal deterministic policy π^*_i

Initialization: $V^*_{M,0}(s)$ arbitrarily initialized $\forall s \in \mathcal{S}$
$\pi^*_0(s), \pi^*_1(s)$ arbitrarily initialized $\forall s \in \mathcal{S}$ with $\pi^*_0 \neq \pi^*_1$
$i \leftarrow 0$
while $\pi^*_{i+1} \neq \pi^*_i$ **do**
    $i \leftarrow i + 1$
    begin ▷ Policy Evaluation
        repeat
            $\Delta \leftarrow 0$
            for $s \in \mathcal{S}$ **do**
                $V^*_{M,i}(s) \leftarrow R(s,\pi^*_i(s)) + \gamma \sum_{s'} T(s,\pi^*_i(s),s') V^*_{M,i-1}(s')$
                $\Delta \leftarrow \max(\Delta, |V^*_{M,i-1}(s) - V^*_{M,i}(s)|)$
        until
        $\Delta < \delta$
    begin ▷ Policy Improvement
        for $s \in \mathcal{S}$ **do**
            $\pi^*_{i+1}(s) \leftarrow \text{argmax}_a \left[R(s,a) + \gamma \sum_{s'} T(s,a,s') V^*_{M,i}(s')\right]$
return π^*_{i+1}

Both approaches suffer from the *curse of dimensionality*, as their scalability can be as poor as the cube of the size of the state space. Indeed, when a discrete representation of the value function or the policy is not realizable, *e.g.* when the states or the actions are numerous or have the cardinality of the continuum, other methods should be devised.

Monte Carlo tree search One way to tackle problems with *finite* big state and action spaces is resorting to a heuristic that guides the exploration in the search space. Possible outcomes of actions in the environment are computed and can be depicted as a tree that grows exponentially in search of a good sequence of moves. The work in (Keller and Helmert, 2013) classified several trial-based heuristic tree search schemes for finite horizon MDPs. Monte Carlo tree search (MCTS) is one of those (Swiechowski et al., 2022). Specifically, MCTS performs the search in the tree using random sampling. Due to the exponential growth of the search tree, an additional heuristic can be set in place to select the next child nodes. The Upper Confidence Bound 1 applied to trees (UCT) algorithm (Kocsis and Szepesvári, 2006) implements the Upper Confidence Bound method from the bandit literature to regulate the problem of *exploration vs exploitation*, telling when it is statistically profitable to expand the search tree in a direction rather than exploring new branches. Notice that this method is not guaranteed to provide the optimal policy. Sometimes the algorithm will blindly prune good leads only because the ripple effects of the strategy will be felt too many time steps forward into a future it did not explore.

Approximate Dynamic Programming When the state space has the cardinality of the continuum and a state can be representable by a finite set of features that can also take real values, a tabular or graph representation of the value function would be infeasible. Solving the problem with the previous methods after discretization of the space would only lead to *approximate* solutions (Whitt, 1978). But how fine should the discretization be? This depends on the dynamical system that dictates the evolution of the environment. The more sensitive the system is to control, the higher the resolution required. Munos and Moore (2002) developed an information theory-based approach to find an optimal discretization paradigm; however, it is inapplicable to states with a large number of features. Indeed, any discretization approach suffers from the curse of dimensionality.

A viable path to address this problem is to approximate the value function and/or the policy with linear function approximators (Bertsekas and J. N. Tsitsiklis, 1995; Munos, 2003). Linear function approximators express the value function and the policy in terms of a collection of parameters $\{w\}$ called *weights* that, when applied to a feature-represented state, would give an approximation to the output of the original function.

For example, let $\mathcal{S} = \mathbb{R}^n$ and hence $s = (\phi_1, \dots, \phi_n)$ where every $\phi_i \in \mathbb{R}$ and ϕ stands for a feature, *e.g.* ϕ_1 could be the longitudinal position of a ball, ϕ_2 its longitudinal velocity and so on. Then, the following approximation can be made:

$$V_M^*(s) \approx \langle w^* | s \rangle = \sum_{i=1}^{n} w_i^* \phi_i. \tag{1.10}$$

Finding the optimal approximate value functions amounts to computing the n optimal weights $(w_i^*)_{i=1}^n$. Since $n \ll \infty$, a control with this method can be obtained for non-finite state MDPs at the expense of quality. Unfortunately, using linear approximators, the obtained control and the estimate of the value function will likely be so imprecise that a real implementation of these algorithms for the control of real-world problems can be useful only in very specific instances. As we will see in Section 1.4.1, the full power of function approximation is leveraged by Deep

Reinforcement Learning that exploits Deep Neural Network (DNN) which are universal function approximators (Cybenko, 1989; Hornik, Stinchcombe and H. White, 1989; Hornik, 1991).

In real-world situations, an agent may not always be able to perfectly observe the current state of the MDP. Sometimes it can only perform partial measurements, *e.g.* when the agent is a robot with imperfect sensors. There is a need to cope with these kinds of situations that are not representable using an MDP. The extension of the framework to deal with partial observability is discussed in the next section.

1.2 An introduction to POMDPs

Definition 9 (Hidden Markov Model). *Let S be a discrete-time MC with values in $\mathcal{S}$ and O be a discrete-time stochastic process with values in Ω. The pair (S, O) is a Hidden Markov Model (HMM) if S is not directly observable and*

$$Pr\left(O_t = o_t \middle| \{S_s = s_s\}_{s=0}^{t}, \{O_s = o_s\}_{s=1}^{t-1}\right) = Pr\left(O_t = o_t | S_t = s_t\right). \tag{1.11}$$

A POMDP is a sequential decision-making process to control an HMM (Figure 1.2).

Definition 10 (Partially Observable Markov Decision Process). *A POMDP is defined as a $8-tuple$ $M = \langle \mathcal{S}, \mathcal{A}, T, R, \Omega, \mathcal{O}, b_0, \gamma \rangle$ where $\mathcal{S}$, $\mathcal{A}$, T, R and γ would define an MDP whose states $s \in \mathcal{S}$ are not fully observable by the agent. Indeed, the latter at each discrete time step receives an observation $O_t \in \Omega$, that is the space of possible observations; $\mathcal{O} : \mathcal{A} \times \mathcal{S} \times \Omega \to [0, 1]$ is an observation function that represents the probability of observing $O_t = o_t$ when the state of the system is $S_t = s_t$ and the agent had taken the action $A_{t-1} = a_{t-1}$; b_0 is the initial belief and denotes the prior the agent possesses about the initial state. Formally, $\mathbb{P}(\mathcal{S})$ is the space of probability functions defined over $\mathcal{S}$ and $b_t \in \mathbb{P}(\mathcal{S})$.*

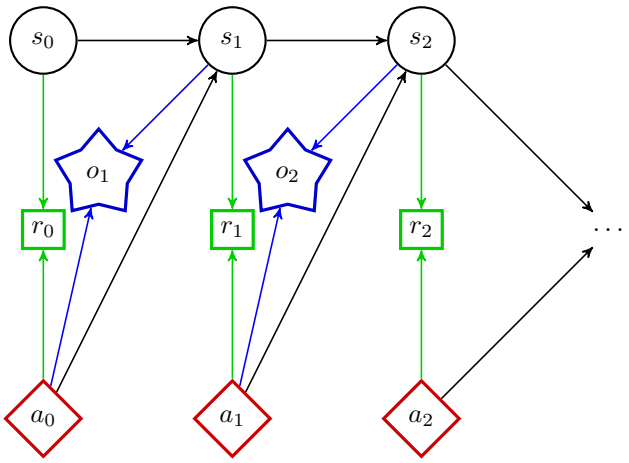


Figure 1.2: The first three time steps of an episode generated by a POMDP. We see that s_t and a_t have an impact on r_t through a green arrow ($r_t = R(s_t, a_t)$) and also an impact on s_{t+1} through a black arrow ($S_{t+1} \sim \Pr(\cdot|S_t = s_t, A_t = a_t) = T(s_t, a_t, \cdot)$). Moreover, aside from $t = 0$, a_t and s_{t+1} also have an impact on the observation o_t through a blue arrow that corresponds to $O_t \sim \Pr(.|S_t = s_t, A_t = a_t) = \mathcal{O}(a_{t-1}, s_t, .)$.

Since the state is not fully observable the agent may leverage the whole history of observations and actions performed to develop, in a Bayesian sense, a belief distribution over possible states at every time step. The belief can be expressed in terms of the original transition and observation

functions. When the agent performs the action a_t and receives an observation o_t, the belief b_t gets updated according to the following rule:

$$b_{t+1}(s_{t+1}) = Z_{t+1}^{-1}\mathcal{O}(a_t, s_{t+1}, o_{t+1}) \sum_{s_t \in \mathcal{S}} T(s_t, a_t, s_{t+1})\, b_t(s_t) \tag{1.12}$$

where Z_{t+1} is a factor that ensures normalization. It is worth noting that the belief is a complete information state, in the sense that all information needed to decide is contained in b.

1.2.1 Solving a discrete POMDP

The belief state properties, particularly the belief update rule in Eq. (1.12), define a Markov Process (Drougard, 2015). This allows us to transform the POMDP into a belief MDP and obtain a solution using the same procedures as in the MDP case. In other words, the solution to a POMDP involves finding the policy that maximizes the value function by exploiting available information and updating its belief accordingly. The solution can take the form of a deterministic functional over the space of possible beliefs: $\pi : \mathbb{P}(\mathcal{S}) \to \mathcal{A}$.

Notice that even with finite states, finite observations, and finite actions POMDP, the belief space $\mathbb{P}(\mathcal{S})$ is not finite, therefore finding the optimal policy is not a trivial task. Research indicate that finding an optimal policy for a finite-horizon POMDP is PSPACE-complete (Papadimitriou and J. N. Tsitsiklis, 1987), and for an infinite horizon, it is undecidable (Madani, Hanks and Condon, 1999).

The value function(al) of a finite horizon POMDP can be represented by a Piece-Wise Linear and Convex (PWLC) function as:

$$V_M^{\pi}[b] = \max_{\alpha \in \Gamma} \langle \alpha | b \rangle = \max_{\alpha \in \Gamma} \sum_{s \in \mathcal{S}} \alpha(s) b(s) \tag{1.13}$$

with Γ being the space of α-vectors. Not only Eq. (1.8) can be easily rewritten for a POMDP, but most importantly the work in Smallwood and Sondik (1973) showed that any successive application of the optimal Bellman operator on the value function keeps it PWLC. The exploitation of this property lays the foundation of many POMDP solvers.

Value Iteration for POMDPs can be untimely due to the cardinality of the belief space which needs to be finely discretized. Notwithstanding, solving a finite horizon POMDP is PSPACE-hard (Papadimitriou and J. N. Tsitsiklis, 1987), and solving an infinite horizon POMDP is undecidable (Madani, Hanks and Condon, 1999).

Cassandra (1998) devised strategies to prune the search of α-vectors, but these algorithmic improvements are only effective on small problems. Pineau, Geoff Gordon, Thrun et al. (2003) developed Point-Based Value Iteration to find an approximate solution: they perform Value Iteration only on a fixed small subset of representative beliefs, *i.e.* the said reachable belief points by playing the POMDP dynamics. Another approximate point-based method is Heuristic Search Value Iteration (HSVI). It solves a POMDP by approximating both a lower and an upper bound of the optimal value function by leveraging the fact that the error of the approximation of V^* is smaller for later beliefs due to the discount factor γ. The improvement of both bounds is the heuristic that drives the search for successive belief points. Kurniawati, Hsu and W. S. Lee (2008) developed the algorithm SARSOP that leverages tighter bounds than HSVI and then drives the search into the areas with the most promising (*reachable*) belief points. Recent POMDP algorithms like POMCP (Silver and Veness, 2010) and DESPOT (Somani et al., 2013) take inspiration from MCTS to yield

approximate solutions to bigger problems.

1.3 Learning a model from experience

So far we outlined MDPs and POMDPs whose model is known. This can be the case in contexts whose rules are very logical and defined by humans: *e.g.* board and video games. What if we have access to a trajectory h but we have no information about the dynamics that drove the stochastic process that generated h?

The *"easiest"* case is a scenario in which the formal representation of a finite-state MDP is known, but not the model dynamics, *i.e.* the transition function T. The problem of inferring the dynamics from data is called *model learning*. If the data set is fixed, later on, we will talk of *offline* model learning and this will be the topic of the next chapter. If further interaction with the environment is allowed during the learning phase we call it *online* model learning.

Being more formal, let $\mathcal{D} = \{(s_i, a_i, r_i, s_{i+1}) : 0 \leq i \leq N\}$ be a collection of transitions (s_i, a_i, r_i, s_{i+1}), we learn a model from $\mathcal{D}$ by fitting the most likely transition function. This amounts to assigning to the probability of every transition its frequency in $\mathcal{D}$.

Definition 11 (Trivial model)**.**

$$\hat{T}(s, a, s') = \frac{\sum_{i=0}^{N} \delta_{s_i,s}\delta_{a_i,a}\delta_{s_{i+1},s'}}{\sum_{j=0}^{N} \delta_{s_j,s}\delta_{a_j,a}}. \tag{1.14}$$

From now on and throughout the dissertation we will refer to $\hat{T}$ as the trivial model.

Learning a POMDP is a completely different task from learning an MDP due to the partial observability of the hidden states. Indeed, the observed data set will be composed as follows: $\mathcal{D} = \{(o_i, a_i, r_i, o_{i+1}) : 0 \leq i \leq N\}$. While in the case of an MDP the transitions between states S_t are Markovian, and hence it is possible to just estimate the frequencies of realizations of single time step transitions to approximate the transition function, POMDPs are beasts of a different kind. What the agent observes in a POMDP is the realization of O_t which could be a non-Markovian stochastic process. Therefore, learning a single-step transition function between observations o_t, as we did for the states s in Eq. (1.14), is not feasible.

First, a strategy to decouple the time series of actions taken during the process must be defined. Let us consider the probability of observing $O_t = o_t$ and $A_t = a_t$:

$$\begin{aligned}\Pr\Big(O_t = o_t, A_t = a_t \Big| \{S_s = s_s\}_{s=0}^{t}, \{O_s = o_s\}_{s=1}^{t-1}, \{A_s = a_s\}_{s=0}^{t-1}\Big) \\ = \Pr\left(A_t = a_t | \{O_s = o_s\}_{s=1}^{t}, \{A_s = a_s\}_{s=0}^{t-1}\right) \Pr\left(O_t = o_t | S_t = s_t, A_{t-1} = a_{t-1}\right),\end{aligned} \tag{1.15}$$

notably, the action depends on the whole history of observable features. If, for instance, the action has been chosen by a uniform random policy, then $\Pr\left(A_t = a_t | \{O_s = o_s\}_{s=1}^{t}, \{A_s = a_s\}_{s=0}^{t-1}\right) = c$, $\forall t$, with c being a constant. After substitution in Eq. (1.15) we have

$$\begin{aligned}\Pr\Big(O_t = o_t, A_t = a_t \Big| \{S_s = s_s\}_{s=0}^{t}, \{O_s = o_s\}_{s=1}^{t-1}, \{A_s = a_s\}_{s=0}^{t-1}\Big) \\ = c \cdot \Pr\left(O_t = o_t | S_t = s_t, A_{t-1} = a_{t-1}\right).\end{aligned} \tag{1.16}$$

The right-hand side of the last equation is proportional to the emission probability of an HMM. Hence, with a proper renormalization, we can just prune a POMDP of its decision-making perk

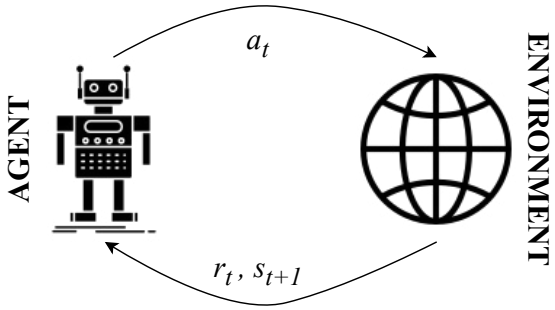


Figure 1.3: In Reinforcement Learning at every time step t an agent iteratively interacts with the environment that transits from a state s_t to a new state s_{t+1}. The agent earns a reward r_t and is informed about the transition.

and consider the underlying HMM. Therefore, the problem becomes that of inferring both the transition function between hidden states and the observation emission probability of an HMM.

This problem is very complex since it has to deal with the inference of a transition probability between unseen quantities and the relative probability for one of these states to *"emit"* a given observation. However, if both the emission probability and the transition function have a known prior distribution, then one could use an Expectation-maximization (EM) method to tackle this issue. An algorithm that performs EM is the one created by Baum and Welch (Bilmes et al., 1998). A strong limitation of this approach is its sensibility to the initial condition. Indeed, different initial conditions could lead to different local maxima of the likelihood.

Secondly, once the transition and observation functions have been inferred, one should find a way to reintroduce the possibility of taking decisions and then define a rule for "splitting" the obtained functions across actions. We are going to define one in Chapter 6.

1.4 Reinforcement Learning

In the previous section, we dealt with the task of learning a model from a fixed batch of experiences. What can be done if further interaction with the environment is available with the aim of gathering more information? The said problem falls into the domain of Reinforcement Learning (RL) (Figure 1.3). The field of RL bloomed with the introduction of Temporal Difference (TD) algorithms (Sutton, 1988) like the famous **Q-learning** (Watkins and Dayan, 1992) and **REINFORCE** (Williams, 1992).

Temporal Difference A TD method exploits the time-delayed return signals to iteratively update statistical estimates of a given quantity. To understand how it works let us define a way to sequentially update an estimate of the average of samples.

Let X be a random variable with values in $\mathbb{R}$ and let estimate the average value A_k of k samples $x \sim X$ $\{x_1, \dots, x_k\}$:

$$A_k = \frac{\sum_{i=1}^{k} x_k}{k}. \tag{1.17}$$

Then, consider that another sample x_{k+1} is available. A_k gets updated to A_{k+1} as follows:

$$A_{k+1} = \frac{kA_k + x_{k+1}}{k+1} = A_k + \frac{(k-k-1)A_k + x_{k+1}}{k+1} = A_k + \alpha_k \underbrace{(x_{k+1} - A_k)}_{\text{TD error}} \tag{1.18}$$

where $\alpha_k = \frac{1}{k+1}$ is called the learning rate and $(x_{k+1} - A_k)$ the Temporal Difference error. Notice

that the TD error is a difference between the value sampled at *"time"* $k+1$ and the already computed average at *"time"* k. The said property lends the name to the approach. Please also notice that $(\alpha_k)_{k\geq 1}$ is a decreasing sequence since the more values have been taken into account to compute the current average, the less the latter needs to be corrected with the information provided by a new sample.

Q-learning Q-learning is a RL algorithm that resorts to Temporal Difference (TD) estimation to compute Q^* in a completely data-driven fashion without needing any information about the model M. The algorithms at its core have a way to iteratively update the estimate of Q^* using new data.

Given a trajectory $h = \{(s_t, a_t, s_{t+1}, r_t), t \geq 0\}$,

$$Q^*_{i+1}(s_t, a_t) = Q^*_i(s_t, a_t) + \alpha_i \Big(\overbrace{r_t + \gamma \max_a Q^*_i(s_{t+1}, a)}^{\text{TD target}} - Q^*_i(s_t, a_t) \Big)_{\text{TD error}} \tag{1.19}$$

where the dependence on M has been omitted, i is the iteration number, and $\{\alpha_i \in (0,1)\}$ is the sequence of learning rates. Q-learning is called an *off-policy* learner because it updates Q assuming that for state s_{t+1} the best action is given by the greedy policy (Sutton and Andrew G. Barto, 2018), *i.e.* the one that maximizes Q over $\mathcal{A}$, even if the greedy policy is not the one that the agent is currently following. An *on-policy* method would be one for which the agent updates the policy that it is following. Notice that, while in Eq. (1.18) the samples used to update the estimate all come from the same distribution, in Q-learning it is the current partial estimate itself that is *bootstrapped* and maximized over actions to compute the new sample, which is called the TD target. Notwithstanding, Q-learning is guaranteed to converge to a positively biased optimal Q value function Q^* if, for an infinite sequence of data, the following conditions are respected (J. N. Tsitsiklis, 1994):

$$\sum_{t=0}^{\infty} \alpha_t = \infty \text{ and } \sum_{t=0}^{\infty} \alpha_t^2 < \infty. \tag{1.20}$$

Bootstrapping is guilty of generating the positive bias in Q-learning (Smith and Winkler, 2006).

Exploitation vs exploration Once the estimate of the Q-value has been updated, the agent interacts again with the environment to collect more data. If it follows a *greedy* policy, *i.e.* the best according to its current estimate, he would probably never sufficiently explore the world and thus would never have access to the necessary information to let the algorithm converge to the optimal Q-value. With this scope in mind, Q-learning adopts *exploration* strategies. At every time step, after the update of the Q-value has been made, there is a rule that establishes if the agent must follow the greedy policy or pick a random action just to explore the environment. Usually, the rate of exploration should decrease in time to make the algorithm converge. Indeed, we can imagine that at the beginning of the iterations, the agent needs to consistently explore to get acquainted with the environment, but later on, its exploration should be more moderated and more fine-tuned across some region that is already proving good returns. Nevertheless, depending on the environment, sometimes exploring can lead to irreversible outcomes. Therefore, other more conservative algorithms have been developed like State-Action-Reward-State-Action (SARSA) (Sutton and Andrew G. Barto, 2018). SARSA is an *on-policy* learner: the update of the Q-value is made using the actual data and not the greedy policy. This allows the algorithm to be more robust but could make it converge only to near-optimal policies. Finding the right balance between collecting new

information and maximizing the returns is called the *exploration vs exploitation dilemma*.

Temporal Difference and Monte Carlo methods Q-learning and SARSA both leverage the bootstrap of the estimate one time step ahead to update the Q function. As already stated, this method is biased even though it has a reduced variance (Smith and Winkler, 2006).

Other paradigms use a full trajectory to update the estimate of the Q function like a **Monte Carlo** estimation method, which is not biased but suffers from a great variance. The variance spawns from combining all the randomness that comes from both a stochastic environment and a possibly stochastic policy throughout a full episode.

To exploit the information contained in a whole trajectory while limiting the growth of the variance, Sutton (1988) conceived TD(λ), an algorithm that updates the estimate of the Q-value by performing a weighted average (with decaying weights) of TD errors with multiple step returns.

Continuous MDPs Likewise MDP solvers, Q-learning requires a discrete representation of the MDP in the sense that it needs to store a value for each state and action pair. When this is not feasible, Q-learning can be adopted with function approximators that can be trained through gradient descent. Indeed, intuitively Eq. (1.19) could be also interpreted as a derivative along the path towards convergence. To that end, one can write

$$\frac{\delta Q_i^*(s_t, a_t)}{\delta i} \approx \frac{Q_{i+1}^*(s_t, a_t) - Q_i^*(s_t, a_t)}{\alpha_i} = r_t + \gamma \max_a Q_i^*(s_{t+1}, a) - Q_i^*(s_t, a_t). \tag{1.21}$$

In this sense, moving forward along this path amounts to adding to the current estimate of $Q_i^*(s_t, a_t)$ a term proportional to the *"gradient"* (the TD error): $\alpha_i \frac{\delta Q_i^*(s_t, a_t)}{\delta i}$. This may remind us of the gradient descent update rule if we replace i by θ, with θ being the parameters of the function approximator. It is important to note that the TD error for a sample can be computed exactly.

The deadly triad Sutton and Andrew G. Barto (2018) refer to function approximation, bootstrapping, and off-policy learning as the *deadly triad*. Unfortunately, when the RL approach dwells on these three methods, the estimated values can diverge and learning might not occur. The convergence conditions of Q-learning with linear function approximation have been widely investigated in the work in Melo and M. I. Ribeiro (2007).

Policy gradients and actor-critic methods A different perspective is optimizing just a parametrized policy without recurring to value function estimation. This is the way paved by **REINFORCE** (Williams, 1992). In REINFORCE transitions are sampled by acting with a parametrized policy $\pi_\theta(a_t|s_t)$. At every time step the parameters θ are updated by $\theta_{i+1} \leftarrow \theta_i + \alpha_i \frac{\delta J(\theta_i)}{\delta \theta}$, where $J(\theta_i)$ is a parametrized expected utility $J(\theta_i) = \mathbb{E}_{\pi_{\theta_i}}[\sum_t R(S_t, A_t)]$. When the data set $\mathcal{D}$ contains N trajectories, the gradient of J can be estimated as follows:

$$\frac{\delta J(\theta_i)}{\delta \theta} \approx \frac{1}{N} \sum_{j=1}^{N} \left(\sum_{t=1}^{H} \frac{\delta \log \pi_{\theta_i}(a_t^j|s_t^j)}{\delta \theta} \left(\sum_{t'=t}^{H} R(s_{t'}^j, a_{t'}^j) \right) \right), \tag{1.22}$$

which amounts to the average over trajectories of the sum along a trajectory of the gradient of the logarithm of the policy with respect to the parameters multiplied by $\sum_{t'=t}^{H} R(s_{t'}^j, a_{t'}^j)$ which are the *returns-to-go* from time t forward. Kakade (2001) proposed an approach to compute policy gradients with gradient descent. The said technique directly optimizes a policy for the maximum discounted reward in an *on-policy* way.

Another class of RL algorithms that exploit both TD and linear function approximation is Actor-Critic (Konda and J. Tsitsiklis, 1999). In this context, a critic estimates the value function using a linear approximator while the actor is updated in an approximate gradient direction based on the current critic's estimate. Later on, Deterministic Policy Gradient (DPG) (Silver, Lever et al., 2014) extended the approach to tackle continuous action spaces. Long story short, actor-critic approaches merge value-based methods (like Q-learning) and policy-gradient-based methods. Approximately, the return-to-go term in Eq. (1.22) is replaced by the estimate of a value function obtained with a value-based technique, *e.g.* policy evaluation or Q-learning.

Distributional RL A big step forward has been taken with the advent of Distributional RL (Morimura et al., 2010; Marc G. Bellemare, Dabney and Rowland, 2023). In this developing field, instead of computing the value function, which assigns to a state the scalar value corresponding to the average discounted return, the algorithms compute return-distribution functions, which map every state to the distribution of future discounted returns. More formally, Eq. (1.6) becomes

$$G_M^\pi(s) \overset{Dist.}{=} R^\pi + \gamma G_M^\pi(S') \tag{1.23}$$

where $\overset{Dist.}{=}$ means equality in distribution, $G_M^\pi(s)$ is a random variable, R^π is also a random variable, S' is the next state random variable according to policy π and MDP M and finally $G_M^\pi(S')$ is a random variable. Indeed, the work in Marc G. Bellemare, Dabney and Munos (2017) showed that the distributional Bellman operator is a contraction for a maximal form of the Wasserstein metric. Inferring the full distribution of returns instead of just the expected value makes the optimization task more suitable to be adopted in *risk-sensitive* problems where avoiding low return trajectories is extremely important since one could optimize for maximizing some low quantile of the said distribution (Morimura et al., 2010). To compute a distribution, several precautions must be taken. First, one should choose a suitable representation for the distribution to learn: for a uniformly spaced discretization, we refer to it as a categorical representation, while for a finite number of uniformly-weighted particles with parameterized locations, we refer to it as a quantile representation.

1.4.1 Deep Reinforcement Learning

When Mnih, Kavukcuoglu et al. (2015) put together the marvelous expressive power of Deep Neural Network (DNN) with Q-learning the gates of a new era opened wide. Indeed, a sequence of linear operators followed by sigmoids can approximate any function if the sequence is long enough (Cybenko, 1989). Later on, this result has been extended to cover different network architectures (Hornik, Stinchcombe and H. White, 1989; Hornik, 1991). In layman's terms, Mnih, Kavukcuoglu et al. (2015) showed that for MDPs with continuous states and finite actions also non-linear function approximators could be used to estimate the Q value of Eq. (1.19) and created a DNN architecture called Deep Q-Network (DQN) (Figure 1.4). As shown intuitively in Eq. (1.21), DQN updates the Q value by Stochastic Gradient Descent using a batch made of the last observed transitions. At every iteration, the Mean Squared Error (MSE) between the TD target and the current Q value is minimized. Since it sits on top of the deadly triad, DQN's convergence is a very delicate matter. The training is susceptible to the tuning of hyperparameters such as the learning rate or the ones that regulate the exploration strategy. However, when convergence occurs DQN showed human-level control capabilities.

In a short time, several improvements have been built over DQN. The research community

has created families of environments to benchmark new approaches. For example, OpenAI Gym (Brockman et al., 2016) includes various classes of environments, such as simple discrete state and action MDPs, classic control problems with continuous states and discrete actions, as well as very complex and multidimensional scenarios based on the physics simulation engine Multi-Joint dynamics with Contact (MuJoCo) (Todorov, Erez and Tassa, 2012), which feature continuous state and action spaces. The Arcade Learning Environment (ALE) is a collection of emulated vintage console games (M. G. Bellemare et al., 2013), where the state space is usually provided as pixels describing the image that a human player would see on the screen.

To accelerate convergence the Prioritized Experience Replay method proposed heuristics to build the batch provided to the architecture at each iteration (Schaul et al., 2016; Lahire, Geist and Rachelson, 2022): instead of serving the last observed transitions, samples are picked based on the impact they had during the training. The idea is that a sample that results in a big step in weight space provides more information and needs to be prioritized. To the end of reducing the positive estimation bias due to the TD method and bootstrap, Hasselt, Guez and Silver (2016) resorted to updating two different DQN with different periods. This strategy not only reduces the bias but also leads faster to convergence. DNN-based version of fundamental RL methods different from Q-learning started to spread.

Among the ones based on the actor-critic literature, we name Asynchronous Advantage Actor Critic (A3C) from Mnih, Badia et al. (2016), Deep Deterministic Policy Gradient (DDPG) from T. P. Lillicrap et al. (2016) which extends DPG. To reduce the function approximation error Fujimoto, Hoof and Meger (2018) proposed Twin Delayed Deep Deterministic policy gradient (TD3). Concurrently, actor-critic approaches with a stochastic actor have been developed obtaining at the time state-of-the-art results (Haarnoja et al., 2018).

Natural policy gradients were united with DNN in the job of Schulman, Levine et al. (2015) that presented Trust Region Policy Optimization (TRPO) and its extension Proximal Policy Optim-

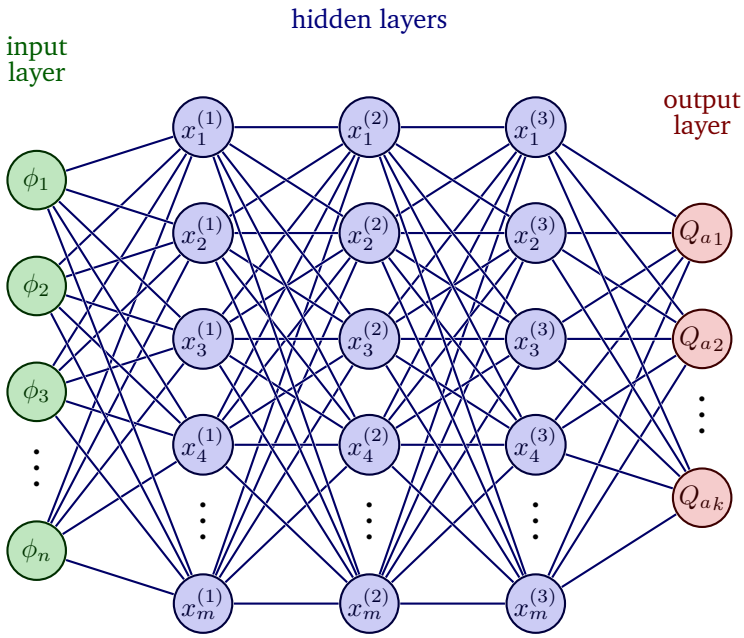


Figure 1.4: A representation of a fully-connected Deep Q-Network with three hidden layers. The architecture receives as input the feature representation of a state $s = (\phi_1, \ldots, \phi_n)$. Linear operators with adjustable weights plus non-linear activation functions are applied to s in sequence, resulting in the hidden layers $x^{(i)}$. Finally the network outputs (Q_{a_j}), $1 \leq j \leq k = |\mathcal{A}|$, one for every action. The whole output corresponds to the estimate of $Q(s, a)$.

ization (PPO) (Schulman, Wolski et al., 2017). In particular, PPO is considered one of the most stable algorithms for hyperparameter selection even today (2023).

Deep Reinforcement Learning models seduced the general public with *AlphaGo* (and *AlphaGo Zero*) (Silver, Huang et al., 2016; Silver, Schrittwieser et al., 2017). This architecture, leveraging value-networks, policy-networks, MCTS (and self-play in the case of *AlphaGo Zero*) managed to beat the champion of Go of the time. The result was remarkable because Go is a board game that was considered for decades out of the reach of RL algorithms. Later on, *AlphaGo Zero* has been improved in *AlphaZero* to master also other board games like chess and shogi (Silver, Hubert et al., 2018), showing an extraordinary ability to adapt to various game rules. However, all of these models require a rule specification to perform the Monte Carlo tree search of profitable next moves. At present *MuZero* is the current upgrade of the *AlphaGo* family and allows to plan in a learned environment, without needing a-priori specifications of the rules (Schrittwieser et al., 2020).

The application of Distributional RL with DNN took place in the same work by Marc G. Bellemare, Dabney and Munos (2017). The authors proposed C51, a neural network architecture that estimates the probability masses of 51 atoms of the Q-value distribution. The number of atoms is arbitrary and can be decided beforehand when the range of the Q value function is discretized. C51 became at the time the state-of-the-art baseline over the Atari 2600 games of the Arcade Learning Environment (ALE) (M. G. Bellemare et al., 2013). Even though the purpose of the algorithm is that of maximizing the expected future discounted return (like classic Deep RL paradigms), keeping track of the whole distribution should convey more precise estimates in an approximate context. The following year C51 was outclassed by DQN with Quantile Regression (QR) that, instead of estimating the probability masses of an equally distributed number of atoms in the range of the Q value distribution, aimed to compute an equally spaced number of quantiles (Dabney, Rowland et al., 2018). The work in Dabney, Rowland et al. (ibid.) introduced the Quantile Temporal-Difference (QTD) for the estimation of quantiles of the return distribution. This work was extended by the Implicit Quantile Network (IQN) (Dabney, Ostrovski et al., 2018) that instead of estimating a fixed number of quantiles it implicitly parametrizes the full quantile function by randomly sampling several quantiles. A further extension was done by the work in D. Yang et al. (2019) that aims to fully parametrize the quantile function. Recently, the work in Rowland et al. (2023) analyzed the convergence of QTD.

The previously mentioned baselines belong to the large family of **Model-free RL**, which is distinct from **Model-based RL**. Model-free paradigms, in line with the original RL motivations, aim to directly estimate value functions or policies without providing any information about the environments and their dynamics. On the other hand, a significant branch of the literature focuses on model-based approaches, which aim to learn a so-called *world-model* that acts as a single (or multiple) step simulator of the dynamics.

Given the data, a model-based approach infers the next state when the current state of the system is s and action a is taken. Planning then occurs in this fictional environment. Historically, model-based approaches are designed to periodically deploy the planned policy to further extend the dataset (Sutton, 1990), thereby improving the quality of the world model. In the next chapter, we will discuss model-based approaches related to offline learning, as we believe that such methods are well-suited for an offline learning scheme.

1.5 Explainability of computed policies

When a policy is derived from a DNN, the agent may exhibit counter-intuitive behavior. To bridge the gap between algorithmic implementations and reliability, it is crucial to *explain* the rationale behind the neural network's weight configuration, i.e., why a specific action is chosen over another in a given state. More generally, and beyond neural network weights, this need for explainability extends to all machine learning models.

In the context of explaining the behavior of automated agents, the reasoning behind their actions becomes increasingly entangled and complex as the number of interacting agents grows. Therefore, developing methods to generate explanations for Multi-Agent System (MAS) is of utmost importance.

What progress has been made in this area? Explainable Artificial Intelligence (XAI) has gained considerable attention among researchers due to the expanding use of algorithms and black-box methods in real-world applications, which affect public trust, policy-makers, and lawmakers. Indeed, *explainability* stands out as one of the pillar requirements of trustworthy AI demanded by the EU Whitepaper on Responsible AI (European Commission, 2020). Although current XAI methods address the explainability of regression and classification models, there has been limited research on the explainability of behavioral policies in both single and multi-agent systems, and more concretely in the realm of RL. The most widely-used baselines, as proposed in the works by S. M. Lundberg and S.-I. Lee (2017) and S. M. Lundberg, Erion et al. (2020), extend the Shapley analysis (Aumann and Shapley, 2015). Originally, Shapley analysis was designed to determine the proportional importance of players in a coalitional game with the goal of distributing a common share more fairly to each participant.

1.5.1 Coalitional games

In this section, we will define transferable utility coalitional games and introduce both Shapley and Myerson values. Shapley values underpin several XAI paradigms. As Myerson values play a central role in the work presented in Chapter 7, they warrant a comprehensive introduction.

Definition 12. *[Transferable utility coalitional game (Peters, 2008)] Let $\mathcal{C}$ be a finite set of players ($|\mathcal{C}| \in \mathbb{N}_+$) and let characteristic function $v : \mathcal{P}(\mathcal{C}) \to \mathbb{R}$ with $\mathcal{P}(\mathcal{C})$ being the power set of $\mathcal{C}$, i.e. a coalition of features. v is called the characteristic function and maps subsets of players, also called coalitions, to real numbers.*

Let v be endowed of the following property:

$$v(\varnothing) = 0. \tag{1.24}$$

A transferable utility coalitional game G is defined as the tuple $G = (\mathcal{C}, v)$.

The characteristic function v describes the worth (utility) of a coalition of players in the game when they cooperate. The worth of an empty coalition is zero (see Eq. (1.24)). The word player is used just to provide intuition and coherence with Shapley's game theory background (Molnar, 2020). A member of the set of players $\mathcal{C}$ could represent anything, and indeed later on in the thesis, it will be composed of individual agent's attributes and policies.

Definition 13 (Shapley value (Aumann and Shapley, 2015))**.** *Shapley analysis allows to compute the Shapley value of a player $i \in \mathcal{C}$ in a transferable utility coalitional game $G = (\mathcal{C}, v)$, and it is*

defined as:

$$\phi_i(v) = \sum_{K \subseteq \mathcal{C} \setminus \{i\}} \frac{|K|!\,(|\mathcal{C}| - |K| - 1)!}{|\mathcal{C}|!} \left(v\left(K \cup \{i\}\right) - v\left(K\right)\right). \tag{1.25}$$

The Shapley value assigns to every player a real number corresponding to its importance in the game. It is defined as the weighted average of the difference in the worth of every possible coalition with and without the player. The Shapley value redistributes the worth of players abiding by the following properties:

1. *Efficiency*: The contribution of each player in the transferable utility coalitional game adds up to the characteristic value v computed on the coalition including all players

$$\sum_{i \in \mathcal{C}} \phi_i(v) = v(\mathcal{C}); \tag{1.26}$$

2. *Symmetry*: If adding element i or element j to any coalition that initially does not include these elements results in the same evaluation of the characteristic function, then elements i and j have the same contribution. Formally,

$$\begin{aligned} &\text{if } v(K \cup \{i\}) = v(K \cup \{j\})\ \forall K \subseteq \mathcal{C} \setminus \{i, j\} \\ &\implies \phi_i(v) = \phi_j(v); \end{aligned} \tag{1.27}$$

3. *Linearity*: The contribution of a player to two transferable utility coalitional games played by the same team but with different characteristic functions v and w, can be linearly combined. That is:

$$\begin{aligned} &\text{let } G_1 = (\mathcal{C}, v) \text{ and } G_2 = (\mathcal{C}, w) \\ &\implies \phi_i(av + bw) = a\phi_i(v) + b\phi_i(w), \forall i \in \mathcal{C},\ \forall (a, b) \in \mathbb{R}^2; \end{aligned} \tag{1.28}$$

   Note that linearity is defined over the space of characteristic functions. It is distinct from the Decomposition property of graph-constrained transferable utility coalitional games (to be defined in Equation 1.30), as the latter allows for the linear decomposition of v computation over a coalition into the linear combination *of the same* v evaluated over the connected components of the graph.

4. *Null player*: if adding a player i to every coalition that did not have it does not affect the computation of the characteristic function, then the contribution of element i is zero.

$$\begin{aligned} &\text{If } v(K \cup \{i\}) = v(K)\ \forall K \subseteq\ \mathcal{C} \setminus \{i\} \\ &\implies \phi_i(v) = 0 \text{ and player } i \text{ is said to be } \textit{null}. \end{aligned} \tag{1.29}$$

In summary, $\phi_i(v)$ is the Shapley value of element $i \in \mathcal{C}$ in the transferable utility coalitional game defined by $(\mathcal{C}, v)$. v is the characteristic function that maps coalitions of elements of the set $\mathcal{C}$ to real numbers, representing the payoff or return of the game within the context of Shapley analysis. Shapley created this method to quantify the contributions of (human) team members working towards a goal, represented by a measurable economic benefit. Intuitively, v can be thought of as the amount of payoff a subset of "players" (regardless of size) earns or loses when they cooperate (see Figure 1.5 for an example). In Chapter 7, Equation 1.24 will play a crucial role in explaining the importance of agents' *policies* and *attributes*, such as features that characterize an

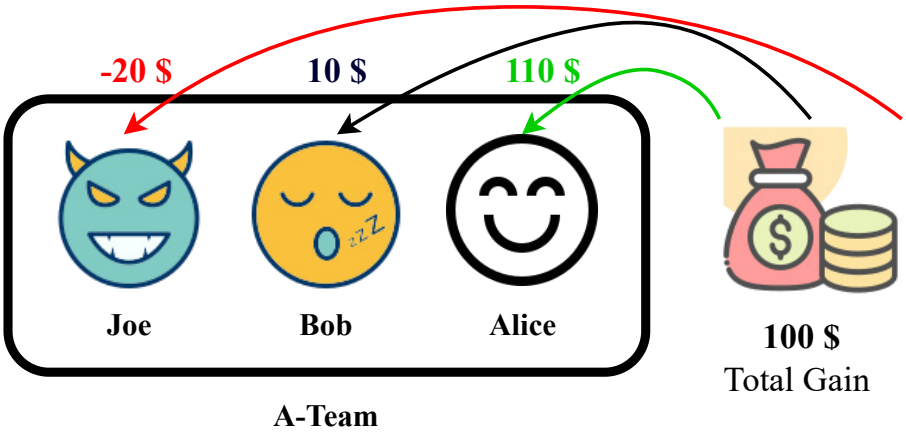


Figure 1.5: Example: a group of computer engineers called the A-Team is paid $ 100 for a project. The members of the group are Alice, a very diligent and happy worker, Bob, who is always sleepy and unproductive, and Joe, an incognito saboteur working for a competing company. A Shapley analysis would find, for instance, a contribution of $ 110 for Alice, $ 10 for Bob, and − $ 20 for Joe. Joe's deeds are detrimental to the project and in an ideal world, he should refund the others.

agent, including its speed, strength, and more.

The algorithm to calculate Shapley values of players can be found in Algorithm 3, where Line 5 corresponds to Equation 1.25. Definition 12 refers to $\mathcal{C}$ as being a finite set of elements, it does

Algorithm 3: Exact Shapley Values Computation

Input: v characteristic function of the coalitional game $(\mathcal{C}, v)$
Output: Shapley value $\phi_i(v)$, $\forall i \in \mathcal{C}$
1 **Initialization:** $\phi_i(v) = 0 \; \forall i \in \mathcal{C}$
2 **for** $i \in \mathcal{C}$ **do**
3 Generate the power set $\mathcal{P}\left(\mathcal{C} \backslash \{i\}\right)$
4 **for** $K \in \mathcal{P}\left(\mathcal{C} \backslash \{i\}\right)$ **do**
5 $\phi_i(v) \leftarrow \phi_i(v) + \frac{|K|!(|\mathcal{C}|-|K|-1)!}{|\mathcal{C}|!}\left(v\left(K \cup \{i\}\right) - v\left(K\right)\right)$
6 **return** ϕ

not specify of what *kind*. It is crucial that all mathematical properties listed in Definition 12, from Equations 1.24 to 1.29, hold.

For instance, maintaining mathematical consistency requires adherence to Eq. (1.24). When applying this approach to real-world scenarios or RL environments, one cannot consider a simulation in which a player (policy or feature) is *entirely removed*. How does one handle the desire for a void coalition or a coalition without specific features or policies? Establishing rules is essential for: 1) facilitating the removal of policies/features, and 2) ensuring that when all elements of $\mathcal{C}$ are to be "removed," then $v(\varnothing) = 0$. This explanation aligns with the empirical findings of Heuillet, Couthouis and Díaz-Rodríguez (2022). In their simulations, they observed that replacing the agents' policies with the *No-Op* policy (do nothing) resulted in the most reliable outcomes. However, it is important to note that this rule is not universally applicable, as its effectiveness depends on the behavior of the characteristic function v. To calculate the Shapley values, the coalitional game should have a transferable utility (Peters, 2008), meaning that players cooperate to gain a common outcome, and subsequently redistribute the spoils in a worth-based proportional fashion.

Definition 14 (Transferable utility coalitional game over a graph)**.** *Let $G = (\mathcal{C}, v)$ be a transferable utility coalitional game and let $\mathcal{G}$ be a graph that has the set of players $\mathcal{C}$ as a set of nodes. We can*

define the transferable utility coalitional game G over the graph $\mathcal{G}$ by limiting the interaction amongst players through the establishment of edges in the graph. The said transferable utility coalitional game will be denoted as $G_{\mathcal{G}}$.
Let μ be a function that maps the set of nodes K of a subgraph of $\mathcal{G}$ to the subsets of nodes corresponding to the connected subgraphs covering K, i.e. to its minimum connected node cover (the cover of the nodes of $\mathcal{G}$ that induces the minimum number of connected subgraphs).

The characteristic function v is hence endowed of the following property in addition to Shapley values' properties 1-4 that is due to the limited interaction between participants dictated by the graph structure:

5. *Coalition decomposition*: the characteristic function evaluated on a coalition K is equal to the sum of the characteristic function evaluated over the elements of the minimum node cover of the subgraph $\mathcal{G}$ whose nodes are in K:

$$v(K) = \sum_{\sigma \in \mu(K)} v(\sigma) \tag{1.30}$$
$$\text{with } \mu \text{ such that } \forall K \in \mathcal{P}(\mathcal{C}), \bigsqcup_{\sigma \in \mu(K)} \sigma = K,$$
$$\forall \sigma \neq \sigma' \in \mu(K)^2, \ \sigma \cap \sigma' = \varnothing,$$
$$\forall (k, l) \in \sigma, \text{ there is a path linking } k \text{ and } l \text{ in } \mathcal{G}.$$

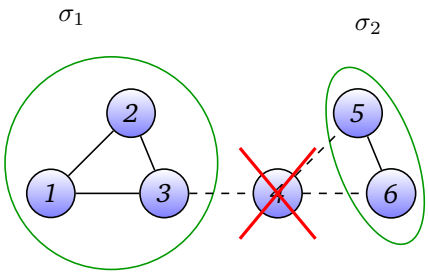


Figure 1.6: Example of a graph Γ over a set of nodes $\mathcal{C} = \{1, 2, 3, 4, 5, 6\}$. When removing the node 4 the graph can be covered by the sets of connected nodes: $\sigma_1 = \{1, 2, 3\}$ and $\sigma_2 = \{5, 6\}$. Coalition decomposition (Equation 1.30) allows to compute $v(\sigma_1 \cup \sigma_2) = v(\sigma_1) + v(\sigma_2)$.

where σ is a dummy variable that spans the elements of the minimum node cover of $\mathcal{G}$. Note that the elements of a node cover are sets of nodes.

For example, in Figure 1.6, we have $v(K = (\sigma_1 \cup \sigma_2)) = v(\sigma_1) + v(\sigma_2)$. This decomposition might not be possible if the game were not graph-constrained, as there could be edges connecting coalition σ_1 and coalition σ_2 even when visualizing a graph structure.

Definition 15 (Myerson value (R. B. Myerson, 1977, 1980))**.** *The Myerson value of a player $i \in \mathcal{C}$ in a transferable utility coalitional game over a graph $G_{\mathcal{G}}$ is indeed defined as the Shapley value (Equation 1.25) of the graph-constrained game.*

Myerson provided a first axiomatization of the problem of allocating importance to the members in a graph-constrained transferable utility coalitional game in terms of equity, efficiency, and fairness (R. B. Myerson, 1977). Later, the work in R. B. Myerson (1980) demonstrated that the only allocation rule for importance in graph-constrained transferable utility coalitional games that abides by all the necessary properties is equivalent to the computation of the Shapley values.

Algorithm 4: Exact Myerson Values Computation

Input: $\mathcal{G}$ graph over the set of players $\mathcal{C}$ of the coalitional game $(\mathcal{C}, v)$, v characteristic function of the coalitional game $(\mathcal{C}, v)$
Output: Myerson value $\phi_i(v)$, $\forall i \in \mathcal{C}$

1 **Initialization:** $\phi_i(v) = 0\ \forall i \in \mathcal{C}$, *Coalitions*: $\mathcal{P}(\mathcal{C}) \to \mathbb{R}$ initially undefined ($\forall \sigma \in \mathcal{P}(\mathcal{C})$,
2 *Coalitions*(σ) = und.)
3 **for** $i \in \mathcal{C}$ **do**
4 Generate the power set $\mathcal{P}(\mathcal{C}\backslash\{i\})$
5 Decompose each $K \in \mathcal{P}(\mathcal{C}\backslash\{i\})$ to $\mu(K)$, the sets of connected nodes minimally covering the subgraph with vertices K
6 Decompose each $K \cup \{i\}$ to $\mu(K \cup \{i\})$, the sets of connected nodes minimally covering the subgraph with vertices $K \cup \{i\}$
7 **for** $K \in \mathcal{P}(\mathcal{C}\backslash\{i\})$ *ordered by increasing* $|K|$ **do**
8 **if** *Coalitions*(K) = *und.* **then**
9 **for** $\sigma \in \mu(K) \wedge$ *Coalitions*(σ) = *und.* **do**
10 Coalitions$(\sigma) \leftarrow v(\sigma)$
11 Coalitions$(K) \leftarrow \sum_{\sigma \in \mu(K)}$ Coalitions(σ)
12 **if** *Coalitions*$(K \cup \{i\})$ = *und.* **then**
13 **for** $\sigma \in \mu(K \cup \{i\}) \wedge$ *Coalitions*(σ) = *und.* **do**
14 Coalitions$(\sigma) \leftarrow v(\sigma)$
15 Coalitions$(K \cup \{i\}) \leftarrow \sum_{\sigma \in \mu(K \cup \{i\})}$ Coalitions(σ)
16 $\phi_i(v) \leftarrow \phi_i(v) + \frac{|K|!(|\mathcal{C}|-|K|-1)!}{|\mathcal{C}|!}\big(\text{Coalitions}\,(K \cup \{i\}) - \text{Coalitions}\,(K)\big)$
17 **return** ϕ

Computationally, the key idea is that the property in Equation 1.30 can be exploited to execute fewer computations as many subsets $K \in \mathcal{P}(\mathcal{C})$ can share the same connected components $\mu(K)$.

It is worth noting that the time complexity of computing the theoretical optimal exact Shapley values is $O(2^{|\mathcal{C}|})$ while the time complexity of computing Myerson values is $O(2^X)$, with $X \leq |\mathcal{C}|$ being a constant proportional to the minimum number of connected nodes covering the graph $\mathcal{G}$ needed to form any coalition.

Algorithm 4 demonstrates how Dynamic Programming can exploit the graph structure to perform fewer computations: once each coalition K with and without a feature i is decomposed into the minimum number of sets of connected nodes covering $\mathcal{G}$ (Lines 4-5), a dictionary containing the already computed value for every small coalition can be expanded by calculating the values starting from coalitions with increasing size (Lines 6-15). When possible, Coalition Decomposition (Equation 1.30) is exploited (Line 10 and Line 14).

Not every transferable utility coalitional game benefits from a prior domain knowledge structure, such as a graph, that restricts the interaction among players. Nevertheless, when domain knowledge structure is available, Myerson values can help take advantage of the graph structure to compute each contribution beyond the participation of each player, thereby explaining the relevance of both individual attributes and policies of agents in a multi-agent environment in terms of a hierarchy of interacting features.

SHapley Additive exPlanations (SHAP) With the concepts of Shapley analysis in mind, S. M. Lundberg and S.-I. Lee (2017) and S. M. Lundberg, Erion et al. (2020) developed SHAP to provide explanations of machine learning models for regression and classification. In this context, the players of the coalitional game were the input features used to perform the classification or regression task. SHAP proposes a method to allocate importance values that is not only global but

also local, through the inference of a kernel to compute importances for features with a changing value, therefore connecting Shapley analysis and previous local explanations techniques for XAI like Local Interpretable Model-Agnostic Explanations (LIME) (M. T. Ribeiro, S. Singh and Guestrin, 2016) and DeepLIFT (Shrikumar, Greenside and Kundaje, 2017). An example of SHAP can be found in Figure 1.7.

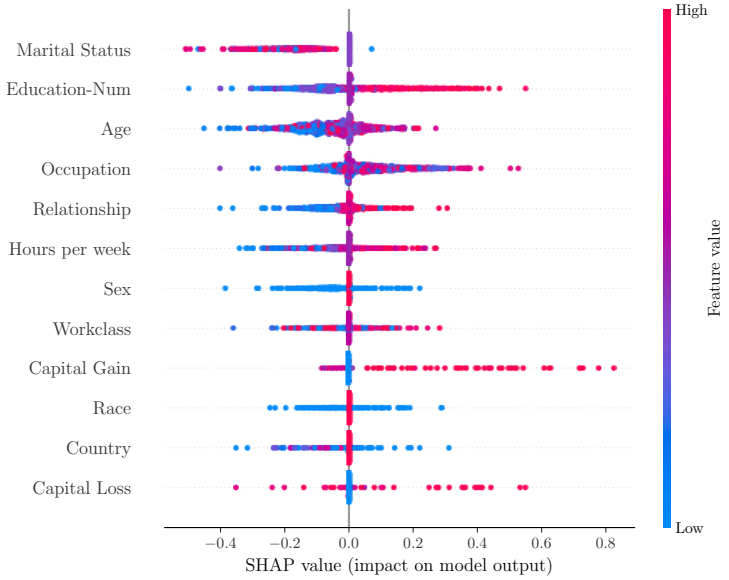


Figure 1.7: The example is taken from SHAP's documentation that uses the standard adult census income dataset from the UCI machine learning data repository (S. Lundberg, 2018). A k-nearest neighbors classifier is trained using sci-kit learn and then SHAP explains the predictions. With a summary plot, it is shown that marital status is the most important feature on average, but other features (such as capital gain) can have more impact on a particular individual (sample).

1.5.2 Explanations for Reinforcement Learning

State-of-the-art taxonomies of Explainable Reinforcement Learning (XRL) methods (Arrieta et al., 2020; Heuillet, Couthouis and Díaz-Rodríguez, 2021) categorize current approaches into two classes: transparent methods, which are interpretable by design, and post-hoc explainability techniques.

Transparent techniques A model is considered *transparent* if it can be directly explained due to its inherent interpretability. This means that not only is the output of the method explainable, but also the computation flow itself, which earns it the name *transparent*. Transparent techniques are composed of a sequence of straightforward, human-understandable functions, such as decision trees.

Post-hoc techniques Post-hoc techniques encompass a set of methods that provide a (usually non-trivial) posterior interpretation of the output. Several XRL baselines employ post-hoc techniques (Heuillet, Couthouis and Díaz-Rodríguez, 2021). In the context of post-hoc techniques, recent work by Heuillet, Couthouis and Díaz-Rodríguez (2022), inspired by SHapley Additive exPlanations (SHAP) (S. M. Lundberg and S.-I. Lee, 2017), developed an approach to obtain estimates of agent importance in cooperative multi-agent RL environments using a Monte Carlo estimation of classical Shapley values (Aumann and Shapley, 2015). In their work (Heuillet, Couthouis and

Díaz-Rodríguez, 2022), a direct computation of importances for a player in a coalitional cooperative game is translated to the context of a Multi-Agent Cooperative System using a simulator.

Although the work in Heuillet, Couthouis and Díaz-Rodríguez (2022) produces reasonable predictions of individual agents' policy importance, it does not extensively discuss why one policy is more important than another or what individual attributes are required from each player to make the said policy effective.

Concurrently, (Metulini and Gnecco, 2022) measured the importances of players in a basketball game using generalized Shapley analysis offline, starting from a set of demonstrations. They did not use a simulator but relied on sequences of action statistics recorded during matches. However, their approach required an ad-hoc and expert-given definition of the characteristic function, which needed to be defined accordingly.

In this context, there is potential for more sophisticated and computationally efficient approaches to explain both the importance of agents' policies and their individual attributes.

1.6 Conclusions

In this chapter, we have explored the problem of sequential decision-making under uncertainty, which we formulated as a Markovian stochastic process. We discussed the control of such a process using a Markov Decision Process (MDP) and reviewed the literature on methods for solving an MDP that rely on the Bellman Optimality Condition. These methods range from classical Value Iteration algorithms to more modern approaches that leverage Reinforcement Learning and Temporal Difference algorithms, as well as expressive function approximators like Deep Neural Networks.

We also addressed the problem of sequential decision-making under uncertainty and partial observability, where the agent lacks complete information about the current state of the system. We introduced the formalization of this problem using a Partially Observable Markov Decision Process (POMDP) and reviewed some approaches for solving a POMDP.

Since both (PO)MDP solvers (with and without function approximation) can produce policies that may be deemed counter-intuitive, we discussed the state-of-the-art in explainability and interpretability for Reinforcement Learning and Machine Learning models. In 2023, explainability in decision-making using Machine Learning-based techniques is an active research area. However, current paradigms face limitations, such as scalability issues and the need for simulators, which require further attention. In the realm of Offline Reinforcement Learning, an explainable and interpretable policy could provide additional validation for technical practitioners and reassure non-expert audiences.

Key Takeaways

- Sequential decision-making under uncertainty is a challenging problem that can been tackled solving an MDP or a POMDP.
- Reinforcement Learning and Temporal Difference algorithms can solve an unknown MDP by trial and error.
- Utilizing function approximators like DNNs, RL can handle MDPs with continuous state and action spaces, albeit achieving convergence can be more challenging.
- Shapley-based analysis can aid in explaining Machine Learning (and RL models), shedding light on counter-intuitive policies and black-box techniques.

In the upcoming chapter we are going to shed light on sequential decision-making in unknown environments using a fully offline data-driven approach. In this context, the goal is to learn the "optimal" policy from a fixed data set of pre-collected experiences. To address the challenge of planning in an unknown environment without the possibility of further exploration, we will review the literature on novel algorithms that take into account the risk of the learned models not being representative of the actual process that generated the dataset, and the resulting policy being ineffective at the time of deployment.

Chapter 2

Offline learning for planning

When addressing a sequential decision-making under uncertainty problem in the real world the practitioner has to cope with the fact that the involved probabilities of transitions in most cases are not known exactly.

In this chapter, we assess the infamous issue of learning an *"optimal"* behavioral policy in a completely offline data-driven fashion. In order to know how to act, one has to possess some knowledge about the consequences of her/his action on the system or at least on the realization of the wished goal. Predicting the time evolution of a system has been a task that physicists have undertaken for centuries. By observing a collection of demonstrations of a phenomenon, they attempt to identify the function that best captures the underlying dynamics. In this context, we will tackle the problem of learning a model to further plan in a (PO)MDP. The general idea of offline learning for planning refers to the inference and the development of methods that, starting from a fixed data set, assist the planning phase. Between these methods, one could consider for example the tasks of learning an abstraction, a model, or directly obtaining a policy.

As we have seen in Eq. (1.14), learning a discrete MDP from data seems quite straightforward. We have to assign to the probability of a transition its frequency of occurrence in the data set $\mathcal{D}$. When the size of the data set $|\mathcal{D}|$ is big enough, the trivial model should tend to the *true* one $\hat{T} \overset{|\mathcal{D}|\to\infty}{\to} T$, *i.e.* the one that was used to generate the data.

How do we know if we have enough data? In practice, this assessment is too hard to be done since the said quantity strongly depends on both the complexity of the environmental dynamics and the data-collecting policy.

But even so, once a model has been learned, which hyperparameters, such as the discount factor, should a practitioner use to obtain a relevant policy?

This chapter provides a review of the literature on Offline Reinforcement Learning, covering methods and approaches for obtaining risk-aware policies, as well as techniques for performing offline evaluation and selection of the most robust policy.

2.1 The importance of the planning horizon

When one applies Value Iteration or Policy Iteration to solve an MDP, she/he always has to define the Value function. In Definition 7 we notice the presence of a power series involving a factor γ, called the discount factor. The factor γ weights future rewards assigning to them geometric decreasing importance. This allows the value function to be well defined, *i.e.* to ensure convergence

properties, even for an infinite horizon MDP:

$$V_M^\pi(s) = \mathbb{E}_\pi\left[\sum_{t=0}^{\infty}\gamma^t R(S_t, A_t)\right] \leq \sum_{t=0}^{\infty}\gamma^t R_{\max} = \frac{1}{1-\gamma}R_{\max}. \tag{2.1}$$

Mathematically, discounting is sufficient for convergence in the infinite planning horizon case and has its roots in the Discounted Utility Model theorized by Samuelson (1937). How should a practitioner choose the discount factor? Ideally, one would want γ as close as possible to 1 to maximize the return along a full trajectory. Sometimes, γ is also to discount finite horizon MDP because a planner would prefer more greedy policies in order to mitigate uncertain outcomes. But why should one keep track of model uncertainty also if a model is not learned from data? To understand it, we should not see γ as a term artificially put into the value function for convergence, and neither as an arbitrary decision of discounting future rewards.
From a stochastic process perspective, imagine an MC that has the same states as the one linked to the original MDP, plus another absorbing end state $\mathcal{S}_{\text{new}} = \mathcal{S} \cup \{\texttt{end}\}$. Then let us define the new transition probability function:

$$\Pr_{\text{new}}\big(S_{t+1} = s_{t+1}\big|S_t = s_t\big) = \begin{cases} \gamma\Pr\big(S_{t+1} = s_{t+1}\big|S_t = s_t\big) & \text{if } s_{t+1} \neq \texttt{end}, \\ 1-\gamma & \text{if } s_{t+1} = \texttt{end}, \\ 1 & \text{if } s_t = \texttt{end} \wedge s_{t+1} = \texttt{end}. \end{cases} \tag{2.2}$$

In the new process, the dynamics between states of the old MC flows exactly as before. The end state is a way to insert model uncertainty into the Markov Chain. At every time step, with a fixed probability $1-\gamma$, the process can end, or end up in some non-predictable phase, that for the practitioner is as good as finished (Lattimore and Hutter, 2014).

The probability of a trajectory $(s_1, \ldots, s_\tau) \in \mathcal{S}^\tau$ starting from s_0 is:

$$\begin{aligned} \Pr_{\text{new}}\big(S_\tau = s_\tau, \ldots, S_1 = s_1\big|S_0 = s_0\big) &= \prod_{t=0}^{\tau-1}\Pr_{\text{new}}\big(S_{t+1} = s_{t+1}\big|S_t = s_t\big) \\ &= \gamma^\tau\prod_{t=0}^{\tau-1}\Pr\big(S_{t+1} = s_{t+1}\big|S_t = s_t\big) \\ &= \gamma^\tau\Pr\big(S_\tau = s_\tau, \ldots, S_1 = s_1\big|S_0 = s_0\big). \end{aligned} \tag{2.3}$$

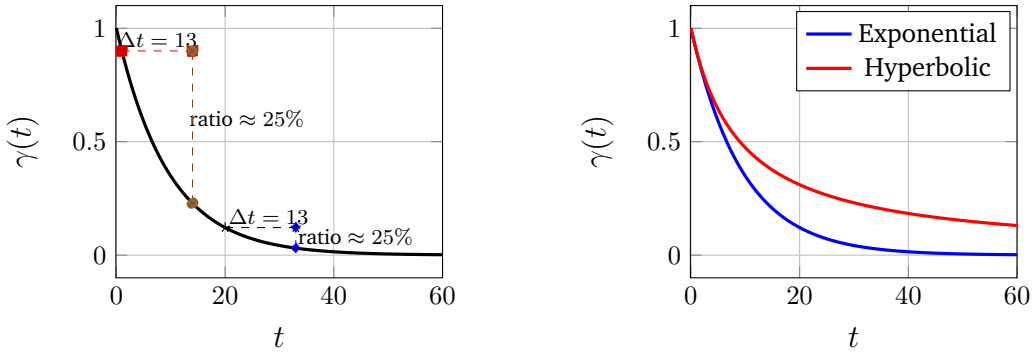
The factor γ^τ that appears in the last expression is the same that we see in the definition of the value function and it would naturally arise in this new setup if we put the constraint that at every time step during the system evolution, we never want to end up in the ending state. Therefore, the planner can forget about the existence of the ending state and model this eventuality by inserting a factor γ in the value function.

Are there other ways of considering model uncertainty based on discount strategies? The answer to this question is many (ibid.). For example, human beings and animals when taking decisions abide by an *hyperbolic* discount strategy rather than exponential (Green and J. Myerson, 2004):

$$V_{\text{hyp.}}^\pi(s) = \mathbb{E}_\pi\left[\sum_{t=0}^{\infty}\frac{1}{1+kt}R(S_t, A_t)\big|S_0 = s\right] \tag{2.4}$$

with k being a constant. Hyperbolic discounting $\gamma(t) = \frac{1}{1+kt}$ is considered an *inconsistent* discounting strategy due to the resulting policies, which can generate temporary preferences for smaller, immediate rewards over larger, delayed ones. This preference structure can lead to future re-

gret for the agent as their decisions may not align with their long-term interests. This effect is due to the hyperbolic function that distorts the values of rewards obtained at different times r_{t_2} and r_{t_1} considering the distance between the time of obtainment of the reward and the current time step, but without keeping a fixed ratio (Laibson, 1997). For example, the ratio between the values assigned to rewards (in the same state and action) obtained at time t_2 and time t_1 are $\frac{1+kt_2}{1+kt_1} \neq \frac{1+k(t_2-t_1)}{1+k}$, where at the right-hand side we are considering that t_1 is the initial time step. This is not the case for exponential discounting $\gamma(t) = \gamma^t$, where the ratio between rewards gained at different time steps is always function of to their difference in time $\gamma^{t_2-t_1}$ independently of the starting instant (see Figure 2.1). Sozou (1998) showed that in a stochastic process



(a) Ratio of discounts at different time intervals for exponential discounting.

(b) Comparison between hyperbolic and exponential discounting strategies.

Figure 2.1: Discounting strategies. (a) Exponential discounting strategy for $\gamma(1) = 0.9$. It is worth noting that exponential discounting reduces the discount factor proportionally by the same coefficient when a period Δt elapses, independently of the initial time, *e.g.* $\frac{\gamma(t+13)}{\gamma(t)} = \frac{\gamma(14)}{\gamma(1)} = \frac{\gamma(33)}{\gamma(20)} \approx 0.25$. (b) Exponential discounting strategy for $\gamma_{exp}(1) = 0.9$ and hyperbolic discounting $\gamma_{hyp}(t) = \frac{1}{1+\frac{1}{9}t}$. At $t = 1$ also $\gamma_{hyp}(1) = 0.9$.

framework considering a Bayesian prior over the possible rate of *"hazard"* $1-\gamma$ automatically results in different discounting strategies. This proves that discountings that are different from the exponential one are not truly inconsistent, but justified when risk is taken into account. A delta function prior (hazard rate exactly known) yields the exponential discount strategy that we have encountered so far. However, a prior that includes uncertainty over the hazard, for example, an exponential prior, yields the *hyperbolic* discount strategy. Nevertheless, the computation of a value function that is characterized by hyperbolic discounting is incompatible with the Bellman equation, hence all the algorithms that exploit the Bellman operator can not be straightforwardly used to solve an MDP with hyperbolic discount. Fedus et al. (2019) reformulated the problem by writing the hyperbolic Q-function as an integral over *"classic"* Q values computed with different discount factors γ. In their work, the authors also approximate the integral with a finite sum of multiple Q values showing promising results. The algorithm was tested in an environment compatible with a real ending absorbing state, and hence not representative of model uncertainty.

Ultimately, from what has been stated so far, interpreting the hazard rate as the chance of ending the usual dynamics of the Markov Decision Process should be extremely useful in accounting for model uncertainty and reducing the planning horizon in an offline learning context.

To that end, N. Jiang, Kulesza et al. (2015) empirically showed that when an MDP is learned from a finite batch there exists an optimal discount factor $\gamma^* \leq \gamma_{\text{ev.}}$ that, when used to obtain the

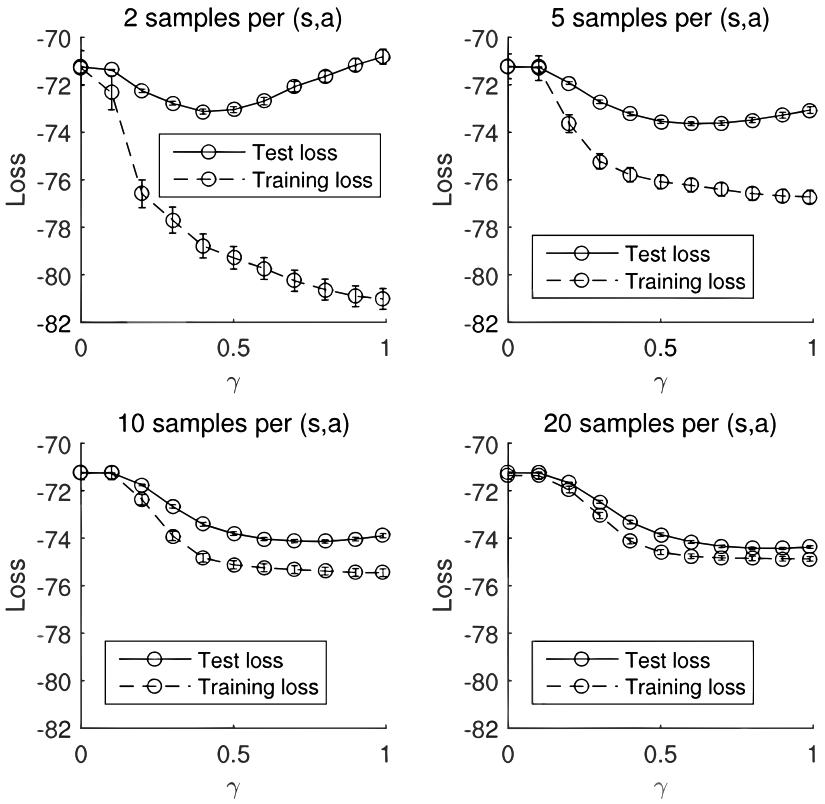


Figure 2.2: Figure taken from N. Jiang, Kulesza et al. (2015). A trivial MDP is learned from batches of different sizes (specified on top of each subfigure). The training loss is the average negative value of the policy $\pi^*_{\hat{M},\gamma}$, which is obtained solving the trivial MDP $\hat{M}$ with discount factor γ, on the trivial MDP $\hat{M}$: $-\frac{1}{|\mathcal{S}|}\sum_{s\in\mathcal{S}} V^{\pi^*_{\hat{M},\gamma}}_{\hat{M},\gamma_{eval}}(s)$. The test loss is the average negative value of the same policy on the actual MDP M: $-\frac{1}{|\mathcal{S}|}\sum_{s\in\mathcal{S}} V^{\pi^*_{\hat{M},\gamma}}_{M,\gamma_{eval}}(s)$. The minimum of the loss for the actual MDP is obtained for a specific discount factor $\gamma^* < \gamma_{eval}$. γ^* approaches γ_{eval} when the batch size grows, meaning that the more the model is accurate, the more long-term planning can be convenient.

optimal policy for the trivial model, performs better than the policy obtained with the discount factor used to evaluate the policy γ_{ev}, *i.e.* two policies are obtained for the trivial model $\hat{T}$ using both γ^* and γ_{ev} and the performance of the policies is then evaluated in the true environment T using γ_{ev} (see Figure 2.2). Notwithstanding, this evaluation is done a posteriori and no strategy is proposed to select γ^*.

2.2 Risk assessment

Planning in a learned environment is a delicate matter. Indeed, if the model learned does not perfectly depict the reality the whole chain of sequential decisions might *"crumble"*, as an agent might choose an action based on anticipated future scenarios that do not materialize. We have seen that the planning horizon can naturally arise in the problem of seeking an optimal policy if the agent also considers the *risk* of ending up in a terminal state or out of the model forecasts. Pondering the said eventuality is crucial because deploying in the real world a strategy that is not risk-sensitive might lead to dramatic aftermaths.

Determining the optimal policy for an MDP with an uncertain or imprecise transition function

is a challenge that practitioners have been addressing since the seventies (Satia and Lave, 1973). Numerous attempts have been made to develop a robust solution for a learned MDP. Historically, the initial approaches considered the probability of transitioning in discrete states and actions MDPs to be bounded within a polytope in the space of probability vectors. For example, the polytope $\mathbb{T} = T : |T(s,a,s') - \hat{T}(s,a,s')| < \delta, \forall (s,a,s') \in \mathcal{S} \times \mathcal{A} \times \mathcal{S}$, where $\delta > 0$ is a threshold that limits the distance of any model in $\mathbb{T}$ from the trivial one. The works in Satia and Lave (1973), C. C. White and Eldeib (1994), Givan, Leach and Dean (2000), Iyengar (2005) and Nilim and El Ghaoui (2005) proposed algorithms that compute an optimal *robust* policy in the sense that, at every time step, a malevolent *nature* picks from $\mathbb{T}$ the transition function that minimizes the average discounted return that the agent seeks to maximize, *i.e.* min-max optimization problem. However, polytope-based approaches are difficult to be used in real-life since the threshold δ is a hyperparameter that needs to be ambiguously tuned.

Interestingly, maximizing with respect to the worst case is a protocol that has been explored also for (online) robust planning in non-stationary MDPs in which the model uncertainty is exasperated by time-changing dynamics (Lecarpentier and Rachelson, 2019).

The works in Shapiro and Kleywegt (2002) also suggested optimizing such a min-max problem, but instead of constraining the model dynamics to live in a polytope, it conceived the model uncertainty in a Bayesian fashion, thinking that the learned model $\hat{T}$ might be distributed according to a prior that included the information on the uncertainty. Nonetheless, the prior in this approach was provided by an oracle, making it difficult for this method to find practical applications in real-world scenarios.

The idea of framing the uncertainty about the model in a probabilistic way inspired approaches that envisaged solving alternative (distributionally based) risk-utility functions. Delage and Mannor (2010) formalized the **chance-constrained MDP** framework, which is a classic MDP except for both the reward function R and the transition function T that can be distributed according to some prior, resorting to the Bayesian formalism. The robust optimization problem is addressed by maximizing the **percentile criterion**. In this context, the planner seeks to find the policy that maximizes a percentile (or a quantile) of the distribution of value functions, which are now random variables distributed according to the Bayesian priors of R and T. The said priors can be also inferred from a data set of transitions. The conjugate prior for T, in the case of finite states and actions MDP, takes a Dirichlet form (Raiffa, Schlaifer et al., 1961).

Priors In particular, consider $|\mathcal{S}|$ random variables Y_i with $i \in \{1, \dots, |\mathcal{S}|\}$ describing the the probability of $(S = s^*, A = a^*) \rightarrow (S' = s_i)$, given a batch of demonstrations $\mathcal{D}$. The random variables Y_i, according to a Dirichlet (uniform and uninformative) prior, can be expressed as:

$$\tilde{\tau}^{s^*,a^*}\left(y_1, \dots, y_{|\mathcal{S}|} \middle| n_1, \dots, n_{|\mathcal{S}|}\right) = \Gamma(\nu) \prod_{i=1}^{|\mathcal{S}|} \frac{y_i^{n_i}}{\Gamma(n_i + 1)} \tag{2.5}$$

where, Γ is the Euler gamma function, n_i counts how many times the transition $(s^*, a^*) \rightarrow s_i$ appears in $\mathcal{D}$ and $\nu = \sum_{k=1}^{|S|} (n_k + 1)$ (ibid.). The most likely configuration for this distribution is $\hat{y}_i = \dfrac{n_i}{\sum_{k=1}^{|S|} n_k}$ while the expected value is $\mathbb{E}_{\tilde{\tau}}[Y_i] = \dfrac{n_i + 1}{\nu}$.

Defining a prior for the reward function in a natural way is a challenging task because the reward function can theoretically be any real-valued function. Without additional knowledge, selecting one prior over another could introduce inappropriate bias in the problem's formalization.

Risk measures The percentile criterion is what in the field of mathematical finance has already been known as the Value at Risk (VaR), which is a *risk measure*. Indeed, risk measures are widely studied in mathematical finance due to the necessity to rationally quantifying the risk of investments (Artzner et al., 1999). Two popular risk measures are the Value at Risk (VaR) and the Conditional Value at Risk (CVaR). It is hence important not to reinvent the wheel, and instead establish formal and quantitative ways to define risk. Only after this has been accomplished can the optimization problem be reformulated, taking this new, crucial feature into account in order to obtain *robust* policies.

In alignment with the method of Delage and Mannor (2010), we first establish risk metrics before moving on to the development of risk-aware utility functions.

Let M be a random variable governed by a probability measure Pr on its domain $\mathbb{M}$, and $u : \mathbb{M} \to \mathbb{R}$ be a measurable function such that $\mathbb{E}\big[u(M)\big] < +\infty$. Following the works in Rockafellar and Uryasev (2002), we define the cumulative distribution function of $u(M)$ as:

$$\Psi(a) = Pr\big(u(M) \leq a\big). \tag{2.6}$$

While in finance or insurance industry, losses that should be minimized (by looking for the optimal decision) are considered, in the MDP framework, the function u is a utility function that should be maximized (by seeking the optimal strategy). Let us consider a (low) risk level $q \in (0, 1)$, that corresponds to the (high) confidence level in (ibid.).

Definition 16 (Value at Risk)**.** *The Value at Risk (VaR) of the utility function u, at the risk level q is*

$$a_q = \inf\big\{a \in \mathbb{R} | \Psi(a) > q\big\}. \tag{2.7}$$

The work in Föllmer and Schied (2016) offers a similar definition for random variables. The definition of the Conditional Value at Risk is slightly different from the one in the work in Rockafellar and Uryasev (2002), because again, in the MDP context, the lowest gains (to be maximized) are considered, and not the highest losses (to be minimized) like in finance.

Definition 17 (Conditional Value at Risk)**.** *The Conditional Value at Risk (CVaR) of the utility function u at risk level q (Föllmer and Schied, 2016) is*

$$\phi_q = \sup_{z \in \mathbb{R}} (z - q^{-1}\mathbb{E}\left[z - u(M)\right]_+). \tag{2.8}$$

With the intent of solving for a risk-sensitive metric efficiently, the works in Petrik, Ghavamzadeh and Chow (2016), Petrik and Russel (2019) and Behzadian et al. (2021) developed methods to cut the support of the Bayesian prior into what is called an *ambiguity set*, *i.e.* a smaller set of possible transition distributions. Such works propose dynamic programming algorithms to solve the said optimization problem obtaining excellent results. Later on, Soft-Robust algorithms were presented (Lobo, Ghavamzadeh and Petrik, 2021). Soft-Robust optimization consists in solving for a weighted average of the classic MDP target (the average discounted return of the trivial model) and a risk-sensitive utility function like the VaR or the CVaR with respect to model uncertainty. This approach is currently the state-of-the-art method and also relies on ambiguity sets.

We point out that reducing the support of the Bayesian prior to an ambiguity set could discard very unlikely regions of the distribution space, resulting in over-conservative policies, in the sense that the obtained policies will be too similar to the batch collector one.

2.3 Offline Reinforcement Learning

Surprisingly, a completely separate branch of research stemmed from the Reinforcement Learning community with the aim of obtaining an optimal policy for an agent in an offline setting using a fixed batch of demonstrations. Even though the purpose is very similar to what we discussed in the last section, it seems that many RL researchers might be unaware of the investigations conducted on offline solving of finite state and action MDPs, as evidenced by the references in their works.

This field goes by the name of *Offline RL*, even though some years ago people were also calling it *Batch RL* (Levine et al., 2020). Historically, it stemmed from the adaptation of RL paradigms to contexts where further interaction with the environment is not possible. One of the fundamental pillars of RL came up short: improving the estimates of the return by better exploring the state-action space. As a consequence, at least the query about balancing the trade-off between exploration and exploitation is no more. However, since exploration was beneficial to the obtainment of a good policy, certain precautions have to be taken to compute robust policies.

Offline RL methods can be grouped into three families: **Model-based algorithms**, **Model-free approaches**, and **Sequence models**.

The main difference between model-free and model-based approaches is depicted in Figure 2.3.

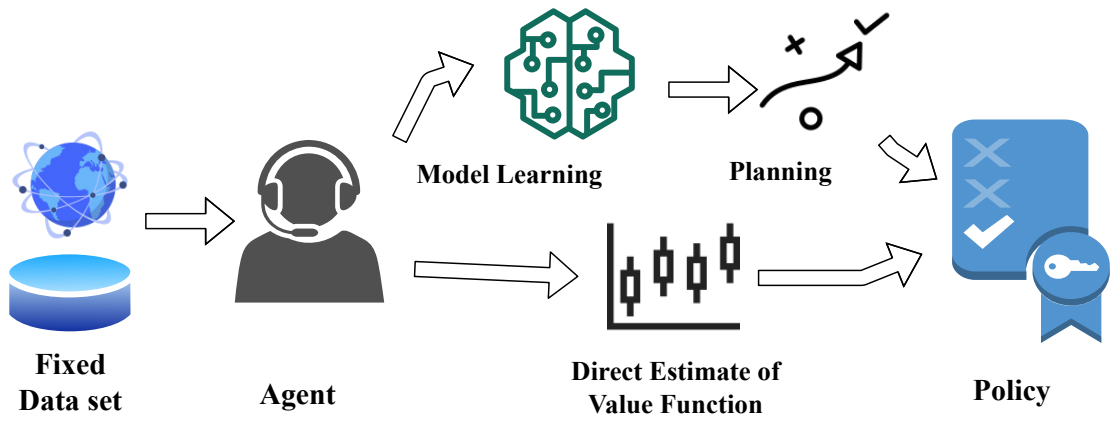


Figure 2.3: Model-based methods learn a world model from data and then plan using it, model-free paradigms fit directly the optimal value functions or policies.

Being successful in an offline learning problem is highly dependent on both the domain and the dataset. In recognition of this, Fu et al. (2020) provided the community with Datasets for Deep Data-Driven Reinforcement Learning (D4RL): a collection of environments, supported by batches of data collected using different policies, which serve as a foundation for benchmarking various offline RL paradigms.

2.3.1 Model-based offline RL

Model-based RL is that Machine Learning paradigm that first learns a *world model* from a data set of demonstrations, *i.e.* a one-step simulator that receives as input the current state and the chosen action and gives as output the next state and the reward, and then uses the world model as a surrogate of the real environment for further interaction.

First attempts of Model-based RL were deployed in an online context for finite states and actions MDPs in which the world model was learned at the same time as a value function and/or a policy (Atkeson and Santamaria, 1997; Sutton and Andrew G. Barto, 2018). One of the first

paradigms belonging to this family was Dyna-Q, which alternated Q-learning in the real environment and Q-learning in a world model that was periodically updated with the newly collected information (Sutton, 1990). Concurrently, to reproduce non-linear dynamics for the control of complex systems, approaches based on parametric function approximators like neural-networks (Draeger, Engell and Ranke, 1995) and Bayesian non-parametric models like Gaussian Processes were proposed (Girard et al., 2002; Kocijan et al., 2004). Encouraging results led researchers to believe that combining planning in an approximate world model with more traditional RL algorithms could lead to optimal policies with less interaction in the true environment (Abbeel, Quigley and Ng, 2006). A significant step forward in the development of data-efficient model-based approaches for controlling complex systems was made by Probabilistic Inference for Learning COntrol (PILCO) (Deisenroth and Carl E. Rasmussen, 2011). PILCO inferred a Gaussian Process and optimized a policy for it by analytically computing gradients. This method was inspired by Model Predictive Control (MPC), also known as moving horizon control (C. E. García, Prett and Morari, 1989; Camacho and Alba, 2013). MPC predicts the dynamics of the controlled system over a finite time horizon while computing an optimal control that minimizes a cost functional and respects dynamical constraints. A new version of PILCO that used the expressive power of DNN was created, obtaining better results than the previous version (Gal, McAllister and Carl Edward Rasmussen, 2016). While the results of PILCO were impressive at the time, it still suffered from the inherent drawbacks of Gaussian Process inference, such as the smoothness assumption of Gaussian kernels. Indeed, Gaussian Processes are suitable for incorporating model uncertainty, but they are not well-suited for inferring high-dimensional dynamics.

In parallel, inspired by Dyna-Q and guided by the breakthroughs of DQN, researchers began developing DNN architectures capable of periodically learning and planning with deterministic *"imaginated"* rollouts (S. Gu et al., 2016; Nagabandi et al., 2018).

Kamthe and Deisenroth (2018) merged the idea of obtaining a probabilistic world model via a Gaussian Process and the MPC solver, allowing them to obtain data-efficient controllers in unknown environments that also could respect dynamical constraints.

To overcome the shortcomings of Gaussian Processes, and hence having a system capable of handling both low and high data regimes, the work in Chua et al. (2018) proposed an approach where first a Probabilistic Ensemble of DNN is fitted to the data and then MPC is launched to obtain the optimal control over stochastic rollouts. The authors showed how this architecture could capture both epistemic uncertainty, *i.e.* uncertainty about the dynamics due to a lack of sufficient data, and aleatoric uncertainty, *i.e.* the one that stems from inherent stochasticity of a system like observation noise. The diversification of aleatoric and epistemic uncertainty has been proven beneficial to the development of risk-sensitive criteria like a safeguard against model bias and aversion to noise (Depeweg et al., 2018).

The Deep Planning Network (PlaNet) architecture managed the task of inferring a world model for environments whose states were represented by images and planning in the latent state space Hafner, Timothy Lillicrap, Fischer et al. (2019). The architecture is split into four main parts: a transition model, an observation model, and a reward model, and a policy model. Uncertainty arising from partial observability in images was addressed through the use of a so-called *recurrent state-space model* for designing the latent dynamical model. Training on the latent dynamics was conducted via image reconstruction. Building upon the PlaNet approach, Hafner, Timothy Lillicrap, Ba et al. (2019) developed Dreamer, which extended the original method by planning directly in the latent model without reconstructing images.

Subsequently and for the first time, a model-based architecture called Simulated Policy Learning (SimPLe) managed to achieve "decent" performances in games of the Atari Learning Environ-

ment. It leveraged a discrete latent embedding before the reconstruction of the image. Readapting Dreamer with the same idea (a discrete representation instead of a Gaussian latent variable) led to DreamerV2, the first model-based architecture that reached human-level performances on Atari (Hafner, T. Lillicrap et al., 2020).

Concurrently, as the last follow-up of the *AlphaGo* architectures, *MuZero* learns the immediate reward function, the value function, and the best policy and uses them to plan both by estimation and Monte Carlo Tree Search (Schrittwieser et al., 2020). The combination of the learned components is used during both the learning and the planning process and allows the architecture to achieve state-of-the-art performances across several board games.

Finally going offline As already stated, the spirit of the cited Model-based RL approaches is to learn a model from demonstration, use it to plan, and then deploy this policy in the real environment to collect more data, hence improving the quality of the world model. Continuously reiterating the approach eventually leads to an optimal policy for the automated agent. If the procedure is stopped at the first iteration, the learning is completely offline. At some stage, the community started to realize that to fulfill completely offline learning some precautions should be taken in order to limit the possible risks of learning an unrepresentative model and hence deploying a dangerous policy. With this in mind, almost in parallel to the previously introduced approaches, Model-based Offline Policy Optimization (MOPO) (T. Yu et al., 2020) and Model-Based Offline Reinforcement Learning (MOReL) (Kidambi et al., 2020) were presented.

MOPO optimizes for uncertainty-penalized MDPs: a new MDP $\tilde{M}$, which has dynamics identical to a trivially learned model $\hat{M}$, but the reward function is adjusted by a term proportional (through a factor $\lambda > 0$) to an estimate of a transition-dependent epistemic uncertainty $\tilde{u}(s, a)$:

$$\tilde{R}(s, a) = \hat{R}(s, a) - \lambda \tilde{u}(s, a). \tag{2.9}$$

In such a way, the optimal policies of $\tilde{M}$ would avoid uncertain regions of the state action space, *i.e.* those for which $\tilde{u}$ is significative. However, obtaining a good estimate $\tilde{u}$ is easier said than done. In the practical implementation of the method, the authors suggest training multiple Gaussian regressors using DNN to predict the model dynamics, and then selecting the variance of the highest regressor for that state-action pair as an estimate of the model uncertainty. Furthermore, tuning λ is far from trivial.

The second approach, MOReL, adds to a trivial model a greatly penalized absorbing state. Simultaneously, an estimator of the uncertainty about a transition is obtained from the data. If, when using the world model, the agent incurs in a state action pair (s, a) for which $\tilde{u}(s, a) > \lambda$, then MOReL would let the state transit to the absorbing one. In this case, $\lambda > 0$ is a threshold for the acceptable epistemic uncertainty per state-action pair. Therefore, when optimizing for the policy, the agent will avoid all uncertain state-action pairs. The idea of transiting to penalized absorbing states for uncertain state-action pairs seems similar to the reasons that drive the introduction of a discount factor γ in the MDP framework, outlined in Section 2.1. However, λ is the same for all state-action pairs.

Likewise, even in this case estimating the model uncertainty is a very complicated task, as acknowledging that an estimate of a transition is incorrect might subtly imply that the actual truth is known. But the truth is not available. Consequently, it is challenging to formulate a precise estimator without creating a cyclical and self-perpetuating problem, akin to the metaphor of a snake biting its own tail.

2.3.2 Model-free offline RL

Model-free offline RL is the class of architectures that directly compute an optimal policy or estimate the values function using a set of demonstrations.

Model-free offline RL paradigms are specifically designed to fit value functions and policies using the same Replay Buffer that an online agent would use. However, they take special precautions to obtain robust policies, without engaging in further exploration, as demonstrated by Lange, Gabel and Riedmiller (2012).

Policy Constraining One of the first Deep Offline RL methods was proposed by the work in Jaques et al. (2017). The authors, in their attempt to create an automated generator for sequences, proposed to train first an action sequence generator based on a RNN and then a second generator using a DQN approach. They used the pdf for a sequence of actions inferred with the Recurrent Neural Network (RNN) to compute a Kullback-Leibler divergence between the said pdf and a stochastic policy obtained via the RL architecture. More in detail, the training phase of the Deep RL network was modified for the loss function to penalize stochastic policies that yield sequences of actions which are *too different* from the one contained in the data set.

The former idea inspired almost every other offline model-free paradigm in the years to follow. Fujimoto, Conti et al. (2019) and Fujimoto, Meger and Precup (2019) formalized the concept in Batch Constrained Deep Q-Learning (BCQ), obtaining control policies in the offline context both for discrete and continuous action environments of OpenAI Gym. Kumar, Fu et al. (2019) enhanced BCQ by providing a more comprehensive explanation for the necessity of constraining the learned policy to not deviate too much from the one(s) that generated the dataset. In fact, since offline Q-learning solely focuses on fitting the Q-value, attempting to estimate the function for state-action pairs not present in the dataset would not only be pointless but also *detrimental*. The error in fitting the Q-value function for these pairs would accumulate due to the Q-value bootstrapping in Q-learning, leading to artificial overestimation of the Q-value for *out-of-distribution state-action pairs* (see Figure 2.4). For this reason, the method Bootstrapping Error Accumulation Reduction

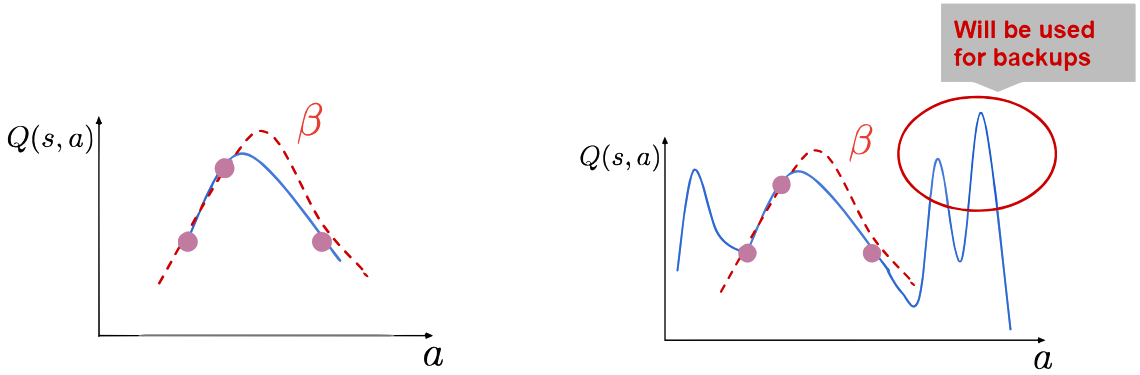


Figure 2.4: Taken from the presentation of Kumar, Fu et al. (2019) at NeurIPS 2019. Q-function training does not query the Q-function value at unobserved states but could query it at unseen actions. While the estimate of the Q-function for a fixed state s and variable action a using three samples in the batch, represented as red dots, could be quite close to the Bayes optimal (dashed curve β) for actions not too distant from the samples (left). The estimate of Q can incur into artifacts for actions far from the samples, and those values could be used in bootstrapping (right).

(BEAR) that constrains the optimal policy to select actions that lie in the support of the distribution of state-action transitions in the batch was proposed.

Eventually, with Conservative Q-Learning (CQL) (Kumar, Zhou et al., 2020), the problem was

attacked from a similar yet different angle. Instead of constraining the optimal policy to lie in the same support of the transition batch distribution, it is the very same Q function that should abide by a regularization. The authors proposed a method for Offline Policy Evaluation (that will be the topic in a later section) and Offline Policy Improvement.

Offline Policy Improvement aims to obtain a lower bound in the true environment for the Q-value of the to-be-deployed policy. This is done by minimizing the expectation of the Q-value under the distribution of states explored in batch $\mathcal{D}$ and actions taken according to the policy. Additionally, it maximizes the expectation under the distribution of state-action pairs inferred from the batch. This is done by balancing the trade-off between batch regularization and Q-learning using a hyperparameter $\tilde{\alpha}$.

$$\hat{Q}_{k+1} \leftarrow \operatorname{argmin}_Q \left[\tilde{\alpha} \left(\underbrace{\mathbb{E}_{s\sim\mathcal{D},a\sim\pi}[Q] - \mathbb{E}_{s\sim\mathcal{D},a\sim\hat{\pi}_\mathcal{D}}[Q]}_{\text{batch regularization}} \right) + \underbrace{\frac{1}{2}\mathbb{E}_{(s,a,s')\sim\mathcal{D}}[(Q - \hat{\mathcal{B}}^\pi \hat{Q}_k)^2]}_{\text{Q-learning MSE TD loss}} \right]. \quad (2.10)$$

Subsequently, to acquire the optimal policy considering this regularization term, the estimate of the Q-value must be improved by performing a maximization over π and another yet-to-be-defined regularization term $\mathcal{R}(\pi)$. This term could be, for example, the Kullback-Leibler divergence between π and a prior policy:

$$\hat{Q}_{k+1} \leftarrow \operatorname{argmin}_Q \max_\pi \left[\tilde{\alpha} \left(\underbrace{\mathbb{E}_{s\sim\mathcal{D},a\sim\pi}[Q] - \mathbb{E}_{s\sim\mathcal{D},a\sim\hat{\pi}_\mathcal{D}}[Q] - \mathcal{R}(\pi)}_{\text{batch regularization}} \right) + \underbrace{\frac{1}{2}\mathbb{E}_\mathcal{D}[(Q - \hat{\mathcal{B}}^\pi \hat{Q}_k)^2]}_{\text{Q-learning MSE TD loss}} \right]. \quad (2.11)$$

Optimizing for this $\mathcal{R}$ dependent min-max problem yields policies that are more robust than the one computed with the previous baselines when applied on the domains and batches from D4RL.

Despite the improvement in the performance of offline methods, they still suffer from the need to tune a set of hyperparameters, much like online methods. Hyperparameter tuning is even more challenging in the offline regime, as the environment cannot provide any feedback regarding their validity.

The recent work in Fujimoto and S. S. Gu (2021) considered a minimalist approach for batch regularization, which, for continuous action space problems, involved simply adding MSE regularizing term between the action provided by the optimal policy to be obtained and the one selected for the sample in the dataset. According to the author, this could lead to algorithms that have the same performances as state-of-the-art ones while reducing the set of hyperparameters to tune and regularization terms to be applied, *e.g.* why use the Kullback-Leibler divergence and not other distances?

Ensemble method Instead of focusing on constraining the learning phase like the methods previously mentioned, the work in Agarwal, Schuurmans and Norouzi (2020) proposed a different solution: the Random Ensemble Mixture (REM) approach. In the latter, the authors train a family of Q-function approximators defined by randomly mixing probabilities on a simplex and then taking the average value as the final output. This way of regularizing the Q-value, based solely on randomly mixing the values of multiple Q approximators, may seem trivial but proved more performant than vanilla DQN and distributional methods in the offline context. However, it has been shown to be inferior to CQL (Kumar, Zhou et al., 2020) or more recent architectures.

2.3.3 Offline RL as a sequence model

The incredible successes of the Natural Language Processing community using Transformer-based architectures for analyzing and creating text generator models (Devlin et al., 2019; Brown et al., 2020) inspired Deep RL researchers.

A trajectory of a sequential decision process is nothing less than a sequence of states, actions, next states, and rewards. By applying a suitable discretization to tokenize continuous variables used to represent states, actions, and rewards, a trajectory can be mapped into a string whose alphabet is the set of categorical tokens used. Based on this insight and on the remarkable ability of transformers to learn distributions and correlations between tokens in sequences, a sequential decision-making problem can be mapped into generating a specific *"text-like"* sequence model, wishing for the sequence of actions that maximizes the sequence of rewards (L. Chen et al., 2021; Janner, Q. Li and Levine, 2021; Reed et al., 2022) (see Figure 2.5). Since Natural Language Processing approaches exploit a fixed data set of sequences, these methods are particularly suitable for offline learning for planning. Even at the level of an initial adaptation of transformers for offline sequential decision-making, the current architectures already claim to have reached the performance of state-of-the-art model-free and model-based approaches. The implementation of

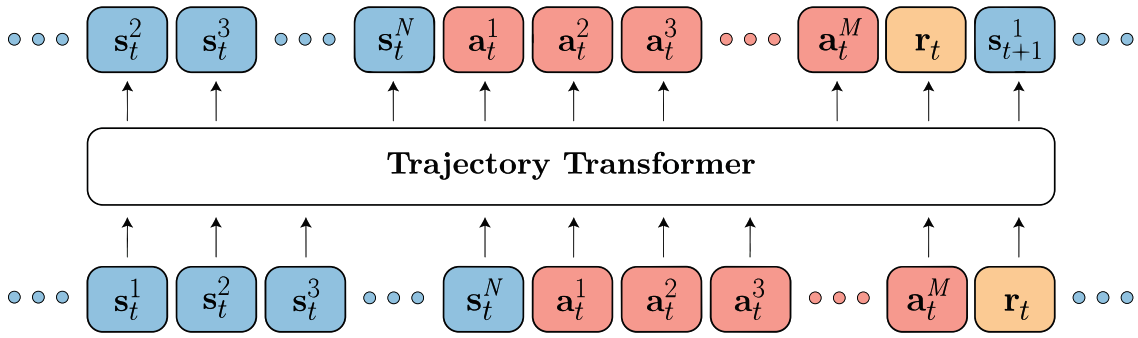


Figure 2.5: Taken from the work in Janner, Q. Li and Levine (2021): *The Trajectory Transformer trains on sequences of (discretized) states, actions, and rewards. Planning with the Trajectory Transformer mirrors the sampling procedure used to generate sequences from a language model.*

sequence models for high-dimensional state and action spaces deserves further investigation since the tokenization approach could potentially suffer from the curse of dimensionality.

2.3.4 Resume

In the end, we believe that the most promising research path would be those that attempt to take into account risk and model uncertainty in a distributional way. In our view, further research should be promoted on risk-sensitive policy constraining in the model-free paradigm and on Bayesian representations of model uncertainty in the model-based framework.

2.4 Abstractions for data efficiency

As previously noted, solving an MDP can be a tedious computational task. On top of that, as mentioned, learning an MDP from data may not be straightforward. Notwithstanding, sometimes the state-action space of the sequential decision-making problem is endowed with a particular structure that allows putting sets of states and actions into peculiar classes of equivalence. Once these

classes of equivalence have been established, one could consider a *simpler* (smaller) MDP whose state-action pairs are just representatives of the said classes. The simpler MDP is called an *abstraction* (see Figure 2.6). Planning in the abstraction would be easier than planning in the original

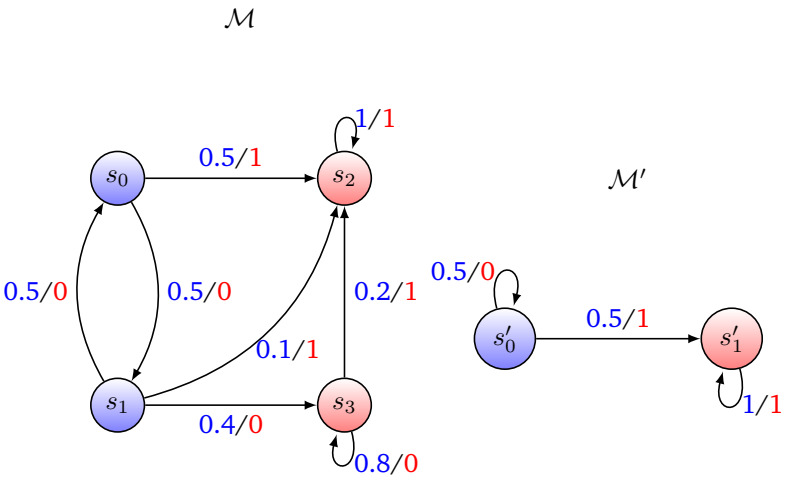


Figure 2.6: Representation of an MDP $\mathcal{M}$ (left) and its abstraction $\mathcal{M}'$ (right). Both MDPs have different number of states but the same number of actions. The states s_0 and s_1 of $\mathcal{M}$ are mapped to s'_0 in $\mathcal{M}'$ (blue) while s_2 and s_3 are mapped to s'_1 (red). Transitions between states are shown with black arrow and probability of transition with respect to the two actions are displayed aside the arrows. The probability of transition with the first action is displayed in blue and the probability of transition with the second action is shown in red. The same reward is assigned to states and actions of the same combination of *colors*. Planning in $\mathcal{M}'$ provides the same optimal policy than planning in $\mathcal{M}$ after lifting.

model. Moreover, up to an appropriate transformation, by construction, every policy obtained in the abstraction can be deployed into the original decision process without loss in performance.
The interest in abstraction schemes rose with the hope of reducing a complex, continuous states MDP into a discrete more tractable one (Whitt, 1978; Bertsekas and J. N. Tsitsiklis, 1995; Munos and Moore, 2002; Munos, 2003).

Several schemes of abstraction have been proposed during the years (L. Li, Walsh and M. Littman, 2006), among many we mention:

1. abstractions preserving the complete model, *e.g.* *homomorphisms* (Dean and Givan, 1997), *symmetries* (B. Ravindran and A. G. Barto, 2001) and *bisimulation* (Givan, Dean and Greig, 2003);

2. abstractions that approximate the original model up to a given error, *e.g.* *approximate homomorphisms* (Balaraman Ravindran and Andrew G. Barto, 2004) and *bisimulation metrics* (Ferns, Panangaden and Precup, 2004).

As far as it concerns the first class, the works in Dean and Givan (1997) and Givan, Dean and Greig (2003) introduced the notions of MDP homomorphism (structure-preserving maps between the original MDP and one characterized by a factored representation) and stochastic bisimulation to automatically partition the state space of an MDP and to find aggregated and factored representations. For those of the second class, B. Ravindran and A. G. Barto (2001) extended the previous works on state abstractions to include the concept of symmetry. Subsequently, the work in Balaraman Ravindran and Andrew G. Barto (2004) considered approximate homomorphisms. It is worth reporting the definition of homomorphism in an MDP.

Definition 18 (MDP Homomorphism)**.** *An MDP homomorphism h (Balaraman Ravindran and Andrew G. Barto, 2004) from an MDP $M = \langle \mathcal{S}, \mathcal{A}, T, R, \gamma \rangle$ to an MDP $M' = \langle \mathcal{S}', \mathcal{A}', T', R', \gamma \rangle$ is a surjection from $\mathcal{S} \times \mathcal{A}$ to $\mathcal{S}' \times \mathcal{A}'$, defined by a tuple of surjections (f, g), with $h(s, a) = \big(f(s), g(a)\big)$, where $f : \mathcal{S} \to \mathcal{S}'$ and $g : \mathcal{A} \to \mathcal{A}'$ such that $\forall (s, s') \in \mathcal{S}^2, a \in \mathcal{A}$:*

$$T'\big(f(s), g(a), f(s')\big) = \sum_{s'' \in [s']_f} T(s, a, s''), \tag{2.12}$$

$$R'\big(f(s), g(a)\big) = R(s, a). \tag{2.13}$$

where $[s']_f = f^{-1}\Big(\big\{f(s')\big\}\Big)$, i.e. $[s']_f$ is the set of states for which the application of f results in the state $f(s') \in \mathcal{S}'$.

Later on, Narayanamurthy and Balaraman Ravindran (2008) showed that the fully automatic discovery of symmetries in a discrete MDP is as hard as verifying whether two graphs are isomorphic. Concurrently, Taylor, Precup and Panagaden (2009) relaxed the notion of bisimulation to allow for the attainment of performance bounds for approximate MDP homomorphisms. Approximate homomorphisms are of particular interest in continuous state MDPs where a hard mapping to an aggregated representation could be impractical. In this context, Ferns, Panangaden and Precup (2004) developed a bisimulation pseudometric to extend the concept of bisimulation relation. The automatic discovery of representations using the bisimulation pseudometric has been investigated in recent years using DNNs and obtaining theoretical guarantees for such a methodology (Ruan et al., 2015; Abel et al., 2020; Castro, 2020). From a different perspective, Mandel et al. (2016) developed an algorithm that aims to cluster MDPs states in a Bayesian sense to solve the MDP in a more data efficient way, even when an underlying homomorphic or symmetric structure is not present. Recently, van der Pol, Kipf et al. (2020) used a contrastive loss function that enforces action equivariance on a to-be-learned representation of an MDP. Their approach resulted in the automatic learning of a structured latent space which can be used to plan in a more data-efficient fashion. Finally, van der Pol, Worrall et al. (2020) introduced MDP Invariant Networks, a specific class of DNN architectures that guarantees by construction that the optimal MDP control policy obtained through other Deep RL approaches will be invariant under some set of symmetric transformations and hence providing more sample efficiency to the baseline when the symmetry is present.

In the context of offline learning, knowledge of these structures prior to training could be exploited by practitioners to improve and augment the batch. Information extrapolated from some transitions could be valid for all state-action pairs in the same class of equivalence, including those unexplored in the batch. Therefore, a method to detect a homomorphism or symmetry, even if not fully automatic, could be a useful tool.

2.5 Offline Policy Evaluation and Selection

Is there a way to predict how performant a policy obtained offline will be when deployed in the real environment? This question is very subtle, since if such a prediction was possible, why not exploit it to improve the policy? Putting things clear, every algorithm has its own, implicit or explicit, method to evaluate a policy. For instance, the CQL update rule for Q described in Equation 2.10 serves exactly the purpose of evaluating offline the Q function of a given policy π. In this context, this section is dedicated to Offline Policy Evaluation methods that were not originally conceived in a

policy calculation pipeline, or that successively have been used exclusively for Off-policy Evaluation (OPE).

Having one or more *"neutral"* policy evaluation schemes is necessary to guide and motivate the selection between policies obtained via different approaches. We classified the said approaches into two groups: Importance Sampling-based methods and regression-based methods.

Regression-based methods Maybe one of the first regression-based methods was Fitted Value Iteration (FVI) (Munos and Szepesvári, 2008), even though having been originally developed to work with a world model (a generative model). The algorithm can iteratively estimate the value function starting from a batch with theoretical bounds on its rate of convergence. It works as follows: first, a set $\mathcal{F}$ of possible value functions is defined; in practical applications, it is a class of parameterized functions like a Deep Neural Network. A value-function $V_0 \in \mathcal{F}$ is randomly chosen. Next, the following steps are iteratively performed at every iteration $k \geq 0$: 1) a Monte Carlo estimate of the Bellman operator applied on V_k using the samples in the batch is performed to obtain a temporary $\hat{V}$; 2) the p-norm based empirical loss between $\hat{V}$ and a function f is minimized for $f \in \mathcal{F}$, the argmin will be V_{k+1}. The authors acknowledged that this approach could suffer from the curse of dimensionality and advocated for the necessity of more adaptive methods.

For this purpose, Le, Voloshin and Yue (2019) proposed a method for (offline) OPE with Probably Approximately Correct (PAC) -style bounds: Fitted Q Evaluation (FQE). The approach is similar to FVI: $\mathcal{F}$ in this case is the set of functions parameterized by weights and representable by a DNN architecture. Furthermore, instead of obtaining an estimate of the Bellman operator applied on V as in FVI's step (1), it is the Q-learning TD target the one to be computed for each sample.

The work in Paine et al. (2020) showed that FQE is a robust method for offline policy (and/or hyperparameter) selection even in high dimensional environments. However, how to tune the hyperparameters of FQE itself (learning rate, neural network layers, etc.) is still an open question.

To provide a reply to this question, the work in Zhang and N. Jiang (2021) proposed a *nearly* hyperparameter-free method for OPE. Nevertheless, and in our view, the said approach is highly computationally costly and insensible to a lack of data coverage, which can result in an over-optimistic evaluation in a small data regime.

Recently, C.-H. H. Yang et al. (2021) proposed a pessimistic method based on both regression and statistical inference to select the policy with the highest worst-case scenario performance between a set of candidates. While promising, a worst-case scenario optimization might result in too conservative choices.

Importance Sampling-based methods Importance Sampling estimators (Rubinstein, 1981) deal with the special problem of estimating functions of *e.g.* according to a distribution using data drawn from another distribution. In the case of OPE, one would like to estimate, for instance, the value function for a policy π when the data in the batch were collected using another policy $\pi_{\mathcal{D}}$. For clarity, let us consider the problem of estimating the expected value of the random variable X over the distribution f using samples drawn from the distribution f'. One could write:

$$\mathbb{E}_f[X] = \int_x x f(x) dx = \int_x x \frac{f(x)}{f'(x)} f'(x) dx = \mathbb{E}_{f'}\left[X \frac{f(X)}{f'(X)}\right], \tag{2.14}$$

that would correspond to the following Importance Sampling estimators.
Classic importance sampling:

$$\mathbb{E}_f[X] \approx \frac{1}{n} \sum_{i=1}^{n} x_i \frac{f(x_i)}{f'(x_i)}, \tag{2.15}$$

and weighted importance sampling:

$$\mathbb{E}_f[X] \approx \frac{1}{\sum_{i=1}^{n} \frac{f(x_i)}{f'(x_i)}} \sum_{i=1}^{n} x_i \frac{f(x_i)}{f'(x_i)} = \sum_{i=1}^{n} x_i w_i. \tag{2.16}$$

The terms proportional to $\frac{f(x_i)}{f'(x_i)}$ are called the *importance sampling weights*.

Importance Sampling estimators have been used to estimate off-policy the value function of an MDP in Precup (2000) and Precup, Sutton and Dasgupta (2001) without and with function approximation. In these cases, the importance sampling weight is the ratio between the *stochastic* policy to evaluate over the *stochastic* policy used to collect the data: $w_t = \frac{\pi(a_t|s_t)}{\pi_{\mathcal{D}}(a_t|s_t)}$. Since the value functions are defined as the expected cumulative reward over a trajectory, the importance sampling weight assigned to a whole collected trajectory of H time steps will be $\prod_{t=1}^{H} w_t$. Importance sampling methods suffer from a high variance that grows with the horizon H.

The work in Dudik, Langford and L. Li (2011) introduced a doubly robust estimator for bandits relying on importance sampling weights both for rewards and policy-induced transitions. N. Jiang and L. Li (2016) extended the said work to MDPs, obtaining more robust estimators than the state-of-the-art. Thomas and Brunskill (2016) merged doubly robust estimators for MDPs and a model-based approach to obtain a more data-efficient off-policy estimator.

With the scope of bounding the variance for infinite-horizon trajectories, Q. Liu et al. (2018) developed an importance sampling OPE resorting to a direct estimation of the stationary state density ratio between the target and behavior policies.

Recently, the work in Chandak et al. (2021) proposed Universal Off-Policy Evaluation (UnO), an importance sampling-based method to estimate offline the performance of a policy adhering to risk-sensitive measures like the Value at Risk and the Conditional Value at Risk of the cumulative discounted reward along a trajectory. UnO is a non-parametric and model-free estimator that allows for the estimation of risk-sensitive metrics based on quantiles by estimating the full cumulative distribution of returns of a fixed policy π starting from a pre-collected batch of experiences. It is the said estimation of the full cumulative distribution that enables the computation of metrics like the VaR and t2(he CVaR. However, UnO and other Importance Sampling-based techniques are only capable of accurately estimating the distribution of returns for stochastic policies, while many state-of-the-art policies are deterministic. Although the set of deterministic policies is a particular subset of stochastic policies, the computation of the importance sampling ratio for deterministic policies collapses onto a Kronecker delta, leading to inaccurate offline evaluations. Therefore, there is a need for the development of a technique that can evaluate and select offline robust deterministic policies.

Eventually, further investigation is coveted since the current baselines suffer from one or several of the following limitations: high variance, dependence on additional hyperparameters, and difficulty in correctly representing the uncertainty.

2.6 What about offline POMDP learning?

When partial observability enters the scene, learning a model from demonstrations becomes an utterly complicated task. Nevertheless, offline POMDP learning is of great importance for robotics' planning (Kurniawati, 2022; Lauri, Hsu and Pajarinen, 2023) since 1) usually robotic agents have access only to error-affected measurements of physical quantities, and 2) a representative model of the system dynamics might be impossible to be written by hand, especially when there are humans in the loop.

The greatest issue in offline POMDP learning is intrinsic and radicated at the root of any possible learning procedure: which representation is the correct one? Remember that in a POMDP the agent has only access to *"observations"*, while the system evolves in time by transitioning across changing, partially observables or *"hidden"* states. To put any learning algorithm in place, one must first deal with choosing how to represent the hidden state space: is it finite or endowed by the cardinality of the continuum? If it is finite, how many states are there? If a practitioner or an automatic algorithm should opt for a continuum space state, would the POMDP be solvable or more approximations would be needed?

The second main issue is model uncertainty. For instance, in the case of a discrete MDP it is almost straightforward to include uncertainty about the estimate of a model within a Dirichlet distribution initialized with the frequency of transitions in the data set (see Eq. (2.5)). However, such opportune modeling is not available when dealing with POMDPs. Moreover, one should encompass not only the uncertainty about the transition function but also the observation function. Offline POMDP learning is a problem way harder than offline MDP learning.

In the context of such challenges, it is worth considering the role of Predictive State Representations (PSRs) (Michael Littman and Sutton, 2001). PSRs offer a different perspective to traditional POMDP models by representing the state of a system as a vector of predictions for future observations, instead of as hidden states. This predictive approach can provide a more intuitive, data-driven understanding of the system's dynamics, and could potentially address some of the representation and uncertainty issues associated with POMDPs. However, while PSRs offer a promising avenue, they come with their own challenges, including the practical limitations of learning algorithms, issues with model initialization and parameter optimization, and the complexities of applying these methods to non-convex systems (Downey, Hefny and Geoffrey Gordon, 2017).

Before the advent of Deep RL, offline POMDP learning has been investigated by the research community. Nonetheless, to the best of our knowledge, no one managed to develop a general approach. The proposed methods need extra expert knowledge, an oracle, or huge approximations when they are up to be applied to real-world scenarios (Fern et al., 2007; Atrash and Pineau, 2010; Taha, Miró and Dissanayake, 2011; Gopalan and Tellex, 2015). Therefore, the current baselines reduce to use-case-specific paradigms that are not applicable in any context, and more often than not, do not even fit in a fully offline learning scheme.

We will now outline how existing methods deal with the problem of the representation choice and with the one of encompassing model uncertainty.

2.6.1 The problem of the representation choice

Choosing the right representation for a POMDP is crucial since it puts at stake the success of the whole approach. A good representation should: (1) make the system abide by the Markov property; (2) be coherent with the (also learned) observation function; (3) be such that the POMDP can be solved or a (sub-)optimal policy can be approximately computed; (4) be representative enough for the learned model to provide a good behavioral policy after resolution.

The majority of existing methods that learn a POMDP offline to address planning for robotics define the representation with expert guidance. In the works from Fern et al. (2007), Atrash and Pineau (2010), Taha, Miró and Dissanayake (2011) and Gopalan and Tellex (2015) the POMDP representation is given by an oracle and only transition and observation functions are learned. Broz, Nourbakhsh and Simmons (2011) assumed that the states are described by discrete variables and the observations can be just a subset of the latter.

In the context where the histories composing the data set have been generated by an unknown POMDP, the work in François-Lavet et al. (2019) showed that choosing a discrete state space with a lower dimensionality than the original POMDP can help reducing overfitting and hence obtaining a more performant policy. The same work also advocates that reducing the discount factor during planning can further benefit the performance of the policy at the time of deployment.

Zheng, Wu and Lin (2018) and Zheng and Lin (2019) use Bayesian non-parametric learning to automatically define the state representation. To do so, the authors resort to a Beta-Process Auto-regressive HMM as a predictive model for the collected data, *i.e.* a non-trivial process with Gaussian co-variate emission and possible infinite states. Notwithstanding, the number of states detected is greatly affected by the hyperparameters used by the approach and they are unlikely to be interpretable.

2.6.2 Model uncertainty

Since learning a POMDP model from data is already arduous, and model uncertainty is a hardship that very few approaches have tried to include and address.

The works in Atrash and Pineau (2010) and Doshi-Velez, Pineau and N. Roy (2012) admitted that a learned trivial model might not be representative enough. For this purpose, they established a Bayesian framework from which models are extracted by a Bayesian prior. However, model uncertainty is then reduced in an online setting by allowing the agent to interact with an oracle.

In an offline context, the work in Zheng and Lin (2019) tackled model uncertainty by extending the POMDP framework to a so-called Vector Autoregressive POMDP. The Vector Autoregressive POMDP considers temporal correlations between observations: the observation at time t does not depend only on the current state s and previous action a but on the whole history of previous observations. Such a non-Markovian dependence in the observation makes it difficult even to call the process a POMDP derivate. The PBVI was also successively extended to deal with Value functions that depend both on the belief and the full history of observations along a trajectory (Zheng and Lin, 2020). Notwithstanding, this approach is not easily interpretable and the extension of the POMDP makes it extremely computationally costly.

Lastly, further research is needed in this field, as current approaches do not provide a universal method that can be applied across a wide range of situations and domains.

2.7 Conclusions

In this chapter, we conducted a comprehensive analysis and review of the literature on the problem of sequential decision-making under uncertainty in an unknown environment, using only pre-collected experiences without further exploration. We note that the success of this approach is heavily dependent on the quality and diversity of the dataset used. Using biased or insufficiently detailed demonstrations can result in policies that fail to perform well in the real environment. Function approximators, for instance, may misevaluate the outcome of actions taken in states not already observed in the dataset. To address this challenge, we examined various methods, including planning with different discounting strategies, leveraging abstractions, and optimizing for risk-sensitive objectives based on the full distribution of returns along a trajectory. However, optimizing for the latter is computationally demanding. With this in mind, we also introduced policy evaluation and selection paradigms to identify the most robust policy from a set of candidate ones that may differ due to a different choice of initialization hyperparameters. Furthermore, we shed light on offline learning in POMDPs, where the presence of supposed hidden states poses a

significant challenge that remains an open question - the choice of representation. In summary, the field of offline reinforcement learning, whether model-based or model-free, continues to demand significant attention from the research community as there is no method yet that can be considered truly satisfactory.

Key Takeaways

- The size and variety of the data set have a great impact on offline learning.
- The outcome of out-of-distribution actions can be misevaluated.
- Planning with a shorter horizon can be beneficial.
- Abstractions can make both learning and planning more data-efficient.
- Risk-sensitive objectives that take into account model uncertainty are coveted.

Part of the content of this chapter gave rise to the following publication:
Giorgio Angelotti, Nicolas Drougard and Caroline Ponzoni Carvalho Chanel (2020).

'Offline Learning for Planning: A Summary'. In: *Proceedings of the 1st Workshop on Bridging the Gap Between AI Planning and Reinforcement Learning at the 30th International Conference on Automated Planning and Scheduling*, pp. 153–161

☞See the proceedings ☞See the arXiv preprint

In the following chapter, we will introduce a field that is poised to become the most significant application case for offline reinforcement learning in the future: sequential decision-making for systems that involve interaction between humans and robots.

Chapter 3

The Firefighter Robot Game use case: towards Mixed-Initiative Human-Robot Interaction

In the first part of this Chapter, we introduce shared autonomy Human-Robot Interaction systems and more specifically the Mixed-Initiative Interaction (MII) paradigm. In the second part, we outline the specific shared autonomy use case scenario, the Firefighter Robot Game (Drougard et al., 2017; Charles et al., 2018), that will serve as a proof of concept environment to test the developed approaches.

3.1 Human-Robot Interaction

HRI is a branch of research devoted to comprehending, designing, and evaluating technologies for use by or with humans (Goodrich, Schultz et al., 2008). Teleoperation can be seen as a kind of HRI, in which human and robot are not co-located (Sheridan, 2016). More specifically, robots are said to be teleoperated when they perform manipulation or mobility tasks in a remote physical environment. This can be: (i) in correspondence with continuous control movements by the remote human; or (ii) when the robot is programmed to execute pieces of the overall task under the supervision of the human operator.

By definition, HRI implies the establishment of a communication channel between humans and robots (Bartneck et al., 2020). The robot is not only accountable for its actions to reach a goal or complete a task but also for its way of properly behaving in a social context populated by humans. To do so, the designer of a HRI should implement in the robot a kind of capability of interpreting and reacting to a not-so-easily forecastable human behavior. A communication channel can be verbal and non-verbal. Non-verbal communication channels are of great importance, particularly in teleoperation scenarios. The most effective non-verbal communication channels that allow a robot to gather information about the human condition and intention in teleoperation cases can be categorized into two broad classes: behavioral and physiological (R. N. Roy et al., 2020).

Behavioral markers provide an intuitive way to capture intention and mission performance. Among behavioral markers, one could infer important information from tracking the gaze (Peysakhovich et al., 2018) or from keystrokes, response time, clicks, and frequency of interaction of a human operator on an interface (Caroline P. C. Chanel et al., 2020).

Physiological markers convey information hidden at a "lower", biological, layer. Numerous works resort to real-time measurements of the electroencephalogram to estimate human satisfaction (Esfahani and Sundararajan, 2011), attention (C.-M. Chen, J.-Y. Wang and C.-M. Yu, 2017), errors (Ricardo, Sobolewski and R. Millán, 2014), and emotions (Dzedzickis, Kaklauskas and Bucinskas, 2020). Recent studies suggest that considering the heart rate and the heart rate variability of a human can provide useful hints regarding her/his cognitive load (Caroline P. C. Chanel et al., 2020; R. N. Roy et al., 2020; G. Singh, Caroline P. C. Chanel and R. N. Roy, 2021).

3.1.1 Autonomy and initiative

HRI does not exclusively involve *information sharing*, but also *task sharing*. The degree of independence of a robot when it comes to deciding not only when and how to communicate, but also when making decisions, is dictated by its *autonomy*. The autonomy of a robot is defined as its ability to make decisions independently of human operators (Mostafa, Ahmad and Mustapha, 2019). Adaptive autonomy refers to autonomous robots that exhibit some degree of flexibility in their behavior assuming that the division of tasks between the human agent and the robotic agent is not fixed, but rather variable. For instance, adaptive autonomy appears in teleoperation cases when both direct remote control and supervisory control are possible.

Interestingly, MII features dynamic autonomy and can be seen as a kind of adaptive autonomy interaction case (S. Jiang and Arkin, 2015).

In a Mixed-initiative HRI, the roles of the human operator and the robots are not predetermined and the status of autonomy features no particular hierarchy. The tasks can be flexibly allocated by the interaction strategy, allowing each agent to switch among them (ibid.). The interaction strategy can enact the allocation and reallocation of tasks among agents at any moment during a mission (Carneiro et al., 2016) to improve the quality of the interaction in terms of both cooperation and independence (S. Jiang and Arkin, 2015) (see Figure 3.1). How can we construct an *optimal* control system that governs interactions in teleoperation scenarios, particularly in accounting for the remote human operator who initiates and subsequently triggers the robot's autonomous behavior? Furthermore, when are such autonomous behaviors relevant? To achieve effective control, a model of the interaction is essential, which ideally should be capable of predicting human behavior.[1] Regrettably, no general mathematical model capable of predicting human behavior currently exists.

Humans, as biological agents, exhibit significant individual differences. In the same situation, each person may behave differently based on factors such as their attitude, background, education, age, gender, and current physical and mental condition. Consequently, a human's actions and reactions can vary greatly depending on the context.

Behavioral modeling One way of modeling human behavior is trying to infer a probability of executing a specific action in a given context in a data-driven fashion, *i.e.* employing a data set. If the dataset is sufficiently large and diverse, and the model benefits from the appropriate degree of expressivity, then hopefully the inferred probability values may accurately represent the likelihood that a metahuman agent, which represents a generalized human agent from the category present in the data set, will act in a particular manner.

In the past, human behavior has been modeled fitting a stochastic process in discrete time (Pentland and A. Liu, 1999). Such a process produces a time series of values representing either the temporal evolution of the system (*e.g.* the mental state of a human, if it can be represented

[1] This is not the case for Model-free Reinforcement Learning approaches that provide black-box solutions.

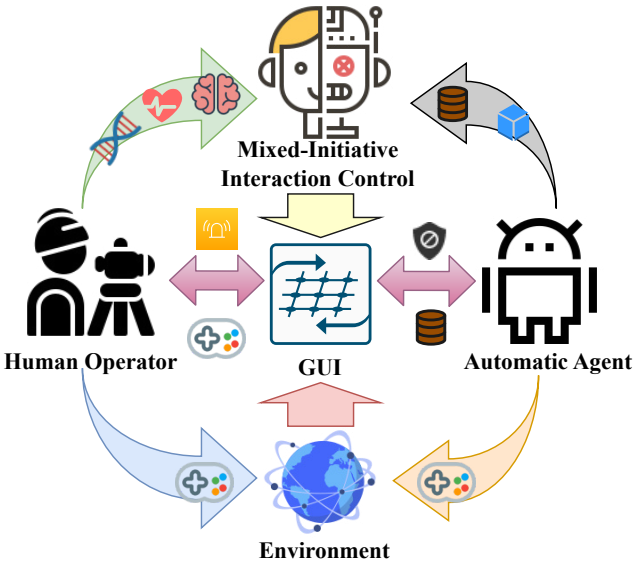


Figure 3.1: Example of a controller for a Mixed-Initiative Interaction System (up). The human operator (left) and the automatic agent (right) feed the controller information about physiological and behavioral markers (up left) or general mission markers (upright). Both operators and the controller interact with a Graphical User Interface (GUI) (middle). The controller allocates the appropriate degree of autonomy to the two agents along with possible alarms to warn the human operator about the state of the mission (middle left and right). Both agents respectively interact within the environment (down).

in some way) or the time evolution of a measurable characteristic generated by a non-directly observable changing state (*e.g.* the time recording of the electrocardiogram or the sequence of actions taken by a human at the teleoperation station). Anyway, inferring human behavior, in general, necessitates a model with a high and unknown expressivity, and thus requires a huge data set encompassing every conceivable situation.
Attempts to learn the stochastic model of human behaviors to optimize planning have been carried out only for specific tasks, *e.g.* assistance during folder navigation in a computer environment (Fern et al., 2007), user-tailored autonomous wheelchairs (Atrash and Pineau, 2010; Taha, Miró and Dissanayake, 2011), calibrating a driving simulator (Broz, Nourbakhsh and Simmons, 2011), and the determination of workload of pilots driving Unmanned Aerial Vehicles (Donath and Schulte, 2015).

Physiological computing Measurements of physiological markers to assess the human's mental states in a general HRI protocol have been investigated (R. N. Roy et al., 2020). In recent studies, Souza, Caroline Ponzoni Carvalho Chanel and Dehais (2015), Drougard et al. (2017) and Caroline P. C. Chanel et al. (2020) showed that including the measurements of both behavioral and physiological features could improve the prediction of team's performance of human-robot MII models by better detecting an *alleged* mental state of the human (or some other quantity functional to the optimization task). The selected features were a) the Heart Rate (HR) and its variability recorded using an electrocardiogram; b) fixations in Areas of Interest (AOI) measured with an eye-tracker; and c) records of the interaction of the user with any interface through a manual controller like a mouse, a keyboard or a joystick.

The study by G. Singh, R. N. Roy and Caroline P. C. Chanel (2022) presented an approach

for designing and solving a POMDP to obtain an adaptive interaction controller for Manned-Unmanned Aerial Vehicles Teaming. The POMDP states were determined by the designer, while the transition function was inferred from a pre-collected dataset of experiences, leveraging variable independence assumptions and expert knowledge. Notably, the resulting interaction controller primarily aimed to enhance the performance of the human operator rather than that of the entire team.

We posit that implementing an adaptive interaction strategy can be especially advantageous in teleoperation scenarios, where one or more human operators must collaborate with one or more mobile robots. In this context, human and artificial agents form a team with a shared objective, and human-related measurements can be accessible to the system responsible for interaction control. That is, the system takes the initiative to initiate autonomous behaviors in the robot(s) when deemed relevant. However, when a human operator is in charge and she/he is interacting with multiple autonomous robots deployed in an environment, she/he may face challenges in interacting with such a complex system. For instance, the capacity of human agents to take over the autonomous system can be strongly affected by several aspects such as poor user interface design, multi-tasking during complex autonomous system operations, high operational pressure, extended time on task, or even emotional commitment. These factors can undermine the human operator's performance and judgment during interaction (Cummings et al., 2013; Régis et al., 2014).

Interestingly, recent literature has demonstrated cases where artificial (software) agents assist human operators as they manage complex systems in a man-machine teaming context. Examples include intelligent ground stations for (multi-)robot deployment (Kaufmann et al., 2021) and power or nuclear power plant control (Marot et al., 2020). The primary goal is to aid the human operator in executing appropriate actions, *i.e.* making the correct decision at the right moment.

The pertinent question then becomes: when, why, and how should the system take the initiative?
Obtaining an adaptive strategy to drive the MII in such cases is complex. Since the (mental) state of the human operator is not directly observable, one could think to design, learn and solve a POMDP (as suggested by Souza, Caroline Ponzoni Carvalho Chanel and Dehais (2015)). Unfortunately, several non-trivial hardships hinder progress in this area. For example, interacting with a system that has a human in the decisional loop can be tedious, costly, and hazardous. Online training is strongly discouraged, whereas offline learning, although preferred, could lead to suboptimal performance policies due to the frugality of available data. Additionally, observations can be derived from various sensors that measure fundamentally different quantities, such as the robot's speed or the human operator's heartbeat rate. On top of this, information fusion must be conducted on data that lack a formal, coherent structure, as they may represent a combination of discrete, continuous, and boolean variables.

These daunting issues make an adaptive interaction application an intriguing use case for examining the theoretical contributions of this thesis. In the following section, we present the Firefighter Robot Game (Drougard et al., 2017; Charles et al., 2018), a playable scenario where a human operator needs to collaborate with a mobile robot to extinguish fires within a confined area. This scenario has been specifically designed to induce deleterious human operator states, as the individual is confronted with multiple tasks under uncertainty and time pressure.

3.2 The Firefighter Robot Game

The Firefighter Robot Game, presented in Drougard et al. (2017) and Charles et al. (2018), was chosen as a use-case because it was willingly designed to be challenging. The game puts the human operator under pressure. Literature showed that in such cases it is relevant to adapt the system (take the initiative) to help the human in order to improve mission performance. Indeed, while dealing with learning an adaptive policy in this setting one must cope with all the hardships of obtaining an optimal strategy for a sequential decision-making process under uncertainty and with a human in the loop:

1. Unknown representation for hidden (mental) states;
2. Unstructured data observations;
3. Information fusion coming from different sensors;
4. Possibly low-data coming from previous experiments in the laboratory in which four missions with eighteen different participants were recorded (Caroline P. C. Chanel et al., 2020);
5. Offline regime, since the learning phase is performed using exclusively a fixed pre-collected data set;
6. High variability in the trajectories formed by the distinct biometric characteristics of human operators;
7. Stochastic unknown environment;
8. Necessity of a robust, risk-aware strategy since the adaptive policy should satisfy a minimum criterium of performance when applied to assist human operators that were never observed in the original batch.

3.2.1 The mission

In a forest with nine trees that can catch fire at any moment, a firefighter human-robot team has the duty of extinguishing as many fires as possible within a 10-minute time frame. The robot is powered by a battery that supplies the necessary electrical energy for movement and has a water tank for extinguishing fires. Additionally, the robot is equipped with a thermometer to check whether its internal temperature does not rise too drastically, a condition necessary for the well-functioning of the machine. The logistics team has established an energy supply zone within the forest where the robot must go to recharge its battery, as well as a water pool to refill the robot's water tank. Unfortunately, the mission is not straightforward. Due to the extreme working conditions, and probably inadequate maintenance, leaks can occur in the water tank. These leaks must be repaired by the human operator, who is the sole agent capable of carrying out this task. Moreover, in order to fill the water tank from the pool, a special nozzle must be carefully balanced in an unstable equilibrium on top of the tank's opening.

The autonomy The robot has two distinct modes of functioning: automatic mode and manual mode.

1. In automatic mode, the robot prioritizes battery recharging and water tank refilling. Consequently, it will autonomously navigate to the energy supplier or water pool when its resources fall below a specified threshold. However, the robot is unable to refill the tank

without the assistance of a human operator, who must operate the nozzle and place it on top of the tank's opening. If the battery and tank levels are above two designated thresholds, the robot calculates the shortest path to the nearest burning tree, drives there, and extinguishes the fire.

2. In manual mode, the robot is entirely remotely operated by the human, meaning that navigation, battery and water recharging, temperature monitoring, and water dispensing must be controlled and initiated by the human operator.

The interface The Graphical User Interface (GUI) is divided into four main sections (see Figure 3.2). In the upper left, there is general information about the mission, including the current score and time remaining in seconds. The upper right section displays a first-person perspective from the robot's primary camera. The lower left section is dedicated to the tasks of water tank refilling and leak repair. In the lower right section, a map of the forest is shown, with trees that may be on fire, as well as a reminder of the robot's status: its operating mode (manual or automatic), battery level, water tank level, and temperature.

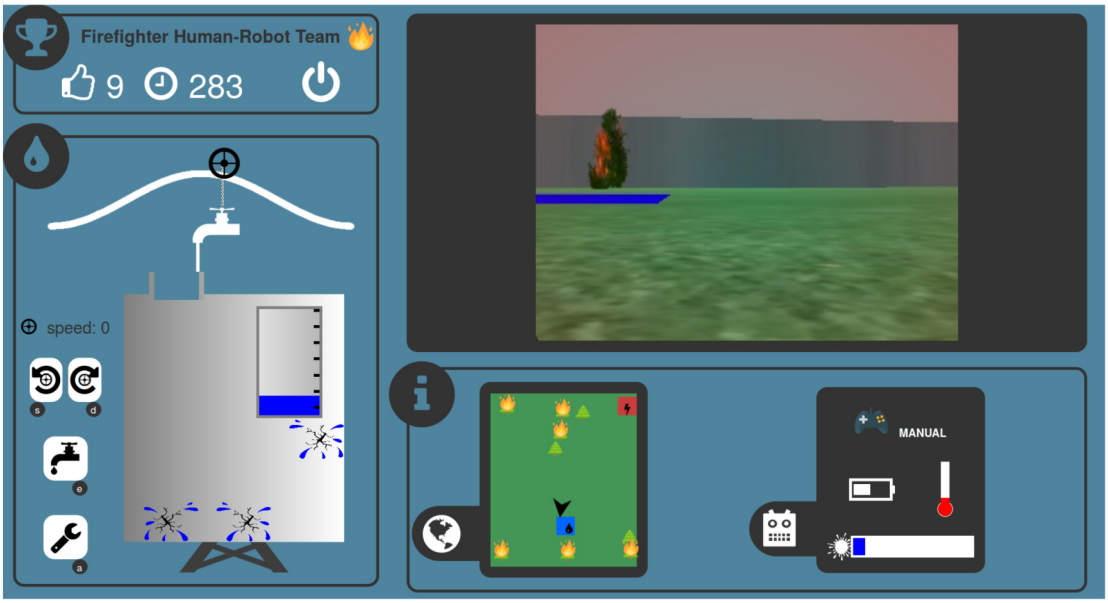


Figure 3.2: Courtesy of Caroline P. C. Chanel et al. (2020). Screenshot of the Graphical User Interface (GUI) during a mission. Top left: the remaining time and score (number of extinguished fires thus far). Top right: robot camera feedback. Bottom left: water stock management task. Bottom right: overall robot status.

Data acquisition Experiments with humans playing the serious game have been carried out in laboratory facilities at Institut Supérieur de l'Aéronautique et de l'Espace (Toulouse, France). The said experiments were run with the aim of collecting data for further studies. Indeed, the collected data will serve as the starting batch for the contribution of this thesis presented in Chapter 6. During the experiments, the functioning of the teleoperated robot firefighter switched between manual and automatic mode uniformly at random every ten seconds. Moreover, alarms concerning important features of the mission were displayed, or not displayed, uniformly at random. A uniformly random control policy is especially appropriate for data collection because it provides the least amount of information, thus ensuring unbiased exploration. Two sets of data collection

campaigns were performed, the first one included the registration of behavioral and physiological markers of eighteen human volunteers along four missions (Caroline P. C. Chanel et al., 2020) (see three samples of processed data in Table 3.1). The last is limited to the collection of actions of the

Table 3.1: Example of post-processed collected data. Each row corresponds to data recorded in a 10-second interval during a mission. In the columns, sub is the subject id; id the mission id; mode and alarm determine the mode of the robot and whether the interface is providing alarms; HRn and HRV are the heart rate and heart rate variabilities normalized with respect to the subject's rest baselines, the procedure is reported in (Caroline P. C. Chanel et al., 2020); nav and space count the number of times the navigation and space keys have been pressed; tank and dtank (or tank_local_score) are features relative to the tank refill task; fires counts the number of fires extinguished; nAOI*i* counts the number of fixations in the *i*-th Area of Interest, while dAOI*j* the cumulative duration of fixations in the *j*-th AOI; score is the total number of fires extinguished during the mission; time displays how much time passed from the start of the mission.

sub	id	mode	alarm	HRn	HRV	nav #	tank	space #	fires #	nAOI1 #	nAOI2 #	nAOI3 #	nAOI4 #	nAOI5 #	dAOI1 (sec)	dAOI2 (sec)	dAOI3 (sec)	dAOI4 (sec)	dAOI5 (sec)	dtank	score #	time (sec)
19	1	M	OFF	6.2	-0.07	14	0	0	0	0	0	4	17	13	0.0	0.0	3.1	4.3	2.6	0.0	27	10
19	1	M	OFF	7.9	-0.09	18	2	0	0	0	5	5	14	8	0.6	1.9	1.2	5.2	1.3	-2.2	27	20
19	1	A	ON	6.9	-0.07	9	5	2	2	0	17	3	6	2	0.1	5.6	0.6	3.1	0.6	-8.5	27	30

human operator on the interface of an online playable version of the game via a crowdsourcing platform (Charles et al., 2018).[2] Due to the ease of playing the online version of the game (no need to access the laboratory) hundreds of missions were recorded.

3.2.2 Discriminative features for performance prediction

In previous research, discriminative features were identified to help predict the performance of the human operator, as demonstrated by Caroline P. C. Chanel et al. (2020). The objective was to determine whether short time windows, specifically 10-second intervals of behavioral and physiological data, could predict the engagement level of the human operator during a mission.

To achieve this, the collected dataset, comprising all 10-second time windows, was categorized into high-performing and low-performing mission groups, facilitating data labeling. The results indicated that behavioral and physiological markers were beneficial for classifying measurements, depending on the robot's automation level (see Figure 3.4). These findings suggest that incorporating physiological and behavioral data of human operators into an interaction control system could lead to improved prediction of their engagement level during a mission, thereby optimizing interactions and enhancing the performance of human-robot teams. The next paragraph outlines the specific physiological and behavioral features employed.

Behavioral and physiological markers During the mission, the Heart Rate (HR) and Heart Rate Variability (HRV) of the human operator are extracted from the Electrocardiogram (ECG) used to collect the cardiac activity at a sampling rate of 500 Hz with the Faros Emotion 360 system. To facilitate comparisons between participants' physiological markers, a normalization process was carried out. This process involved adjusting the live measurements by subtracting each participant's HR and HRV values recorded at rest one minute prior to each mission. By normalizing the data in this manner, the resulting physiological information was believed to become more consistent across participants, enabling more accurate analysis and interpretation. Gaze movements and fixations events are tracked using the SMI Red 250 Hz ET system with a threshold of 80 ms to identify in-stream events. The interface is divided into five AOIs (see Figure 3.4). Key pressed, response time, mouse clicks, and general frequency of interaction with the interface are also recorded.

[2]The game is accessible at https://robot-isae.isae.fr/welcome [Online: accessed 13-04-2023].

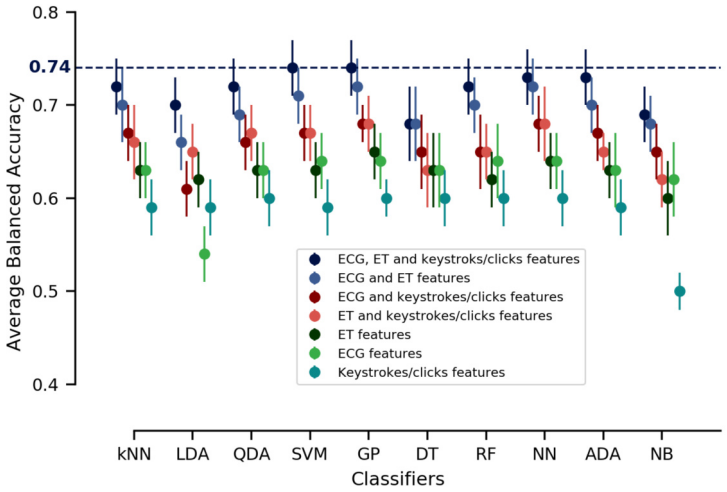


Figure 3.3: Courtesy of Caroline P. C. Chanel et al. (2020). Average balanced accuracy for classification of 10 seconds averaged measurements during a Firefighter Robot Game mission with different human operators. Input features were conditioned on the automation level of the robot, which constitutes an additional common input feature for all tests. The following classifiers were tested. kNN: k-Nearest Neighbors; LDA: Linear Discriminant Analyses; QDA: Quadratic Discriminant Analyses; SVM: Support Vector Machine; GP: Gaussian Process; DT: Decision Trees; RF: Random Forest; NN: Neural Network; ADA: AdaBoost; and NB: Naive Bayes. For every classifier best results were achieved when all the features were used.

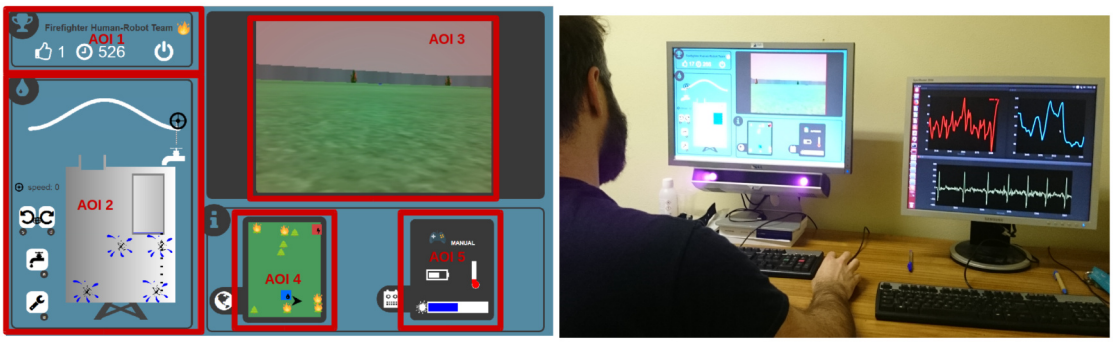
3.3 Conclusions

An adaptive strategy for the quasi[3] MII control system in the Robot Firefighter Game should select the most appropriate action at every 10-second, *i.e.* the discrete decision time step, to maximize the overall performance (the number of extinguished fires). Assuming this time interval, the control system for adaptive autonomy should exploit available observations to decide which action to execute between the following:

1. Set the robot in automatic mode and turn the alarm notifications on;

2. Set the robot in automatic mode and turn the alarm notifications off;

[3]The automated controller can take the initiative from the human operator but not vice versa, therefore the initiative is not *fully* mixed.



(**a**) AOIs definition following the GUI. (**b**) ECG data visualization during experiment.

Figure 3.4: Courtesy of Caroline P. C. Chanel et al. (2020). (a) Screenshot of the graphical user interface showing the areas of interest (AOIs) defined for the study; (b) ECG data acquisition during the experiment.

3. Set the robot in manual mode and turn the alarm notifications on;

4. Set the robot in manual mode and turn the alarm notifications off.

Preliminary studies showed that learning an MDP from demonstrations collected using a random control strategy, could, at least in principle and only in simulation, produce an adaptive optimized policy that benefits from both the mission data and the behavioral markers of the human operator (Charles et al., 2018).

The Firefighter Robot Game exhibits several general properties of a typical human-robot interaction scenario, such as intersubject variability, latent intention, stochastic dynamics, and risky consequences of actions. Considering all aspects, this environment serves as an ideal proof-of-concept use-case for testing Offline Reinforcement Learning and planning methods augmented with physiological computing.

Key Takeaways

- Mixed-initiative interaction refers to a type of Human-Robot interaction where both the human and the automated agent can initiate actions.
- The mixed-initiative interaction setting can be applied to teleoperated scenarios such as the Robot Firefighter Game.
- POMDPs provide a framework to develop adaptive control strategies for mixed-initiative interaction scenarios.
- Recent experiments have shown that incorporating physiological and behavioral markers of the human operator can enhance the quality of adaptive policies in mixed-initiative interaction scenarios.

In what follows, the contributions of this thesis are presented in **Part II**. These contributions are organized across four chapters, as outlined below:

- **Chapter 4** outlines a method to detect alleged symmetries in an MDP in a total offline setting.
- **Chapter 5** proposes a Bayesian method to encompass model uncertainty when a discrete MDP is learned from data, and subsequently exploits this formalism to select a risk-sensitive policy from a set of candidate strategies.
- **Chapter 6** adapts some of the tools established in the previous chapters to first obtain an interpretable model for human-robot Mixed-Initiative Interaction and then to develop a robust behavioral adaptive control system to drive the interaction.
- **Chapter 7** advocates a method to compute not only the contribution of policies in multi-agent systems but also the contribution of agents' features to the final score.

Finally, **Part II** concludes with a chapter that summarizes the entire work and discusses the limitations and perspectives of the research.

Part II

Contributions

Chapter 4

Data augmentation for offline MDP learning

In this chapter, we address one of the numerous issues concerning offline learning for planning, as presented in Chapter 2. Specifically, we focus on the topic of detecting abstractions for more data-efficient learning (refer to Section 2.4). We propose that it is feasible to harness prior information on inherent symmetries in the dynamics of an MDP for their evaluation and subsequent utilization.

First, let us provide a definition of symmetry in an MDP.

Definition 19 (MDP Symmetry)**.** *Given an MDP $\mathcal{M}$, let k be a surjection on $\mathcal{S} \times \mathcal{A} \times \mathcal{S}$ such that $k(s,a,s') = \big(k_\sigma(s,a,s'), k_\alpha(s,a,s'), k_{\sigma'}(s,a,s')\big) \in \mathcal{S} \times \mathcal{A} \times \mathcal{S}$. Let $(T \circ k)(s,a,s') = T\left(k(s,a,s')\right)$. k is a symmetry if $\forall (s,s') \in \mathcal{S}^2$, $a \in \mathcal{A}$ both T and R are invariant with respect to the image of k (an illustrative example will be provided shortly below):*

$$(T \circ k)(s,a,s') = T(s,a,s'), \tag{4.1}$$

$$R\big(k_\sigma(s,a,s'), k_\alpha(s,a,s')\big) = R(s,a). \tag{4.2}$$

If $k_\sigma = k_{\sigma'}$ then an MDP symmetry is also an MDP homomorphism. On one hand, the studies by van der Pol, Kipf et al. (2020) and van der Pol, Worrall et al. (2020) demonstrated that constraining the learning process of the obtainable Q-value function using a valid symmetry results in convergence of the estimate with fewer samples. However, in this approach, valid symmetries were provided by an oracle. On the other hand, it has been established that the fully automatic discovery of homomorphisms in a finite MDP constitutes a graph isomorphism complete problem (Narayanamurthy and Balaraman Ravindran, 2008).

In this context, the research questions that we answer are the following:

1. *Is it possible to develop a method for expert-guided detection of alleged symmetries in the context of offline learning?*
2. *Is Data Augmentation exploiting a detected symmetry beneficial to the learning of an MDP policy in the offline context?*

Contribution The contribution of this chapter is the proposal of a method based on Density Estimation statistical techniques to validate the presence of an *expert-given, alleged* symmetry in an MDP to exploit it to obtain a better solution in the offline setting.

To clarify the motivations, we present an intuitive example below.

One of the most emblematic examples is checking whether or not a dynamical system is symmetric with respect to a specific transformation of the system of reference. Consequently, we aim to exploit this symmetry for more efficient learning. For instance, consider the well-known CartPole domain of the OpenAI's Gym Learning Suite (Brockman et al., 2016). In CartPole, the purpose of the automated agent is to prevent a rotating pole situated on a sliding cart from falling due to gravity. The state of the system is expressed as a tuple $s = (x, v, \alpha, \omega)$ where x is the position of the cart with respect to a horizontal track upon which it can slide, v is its longitudinal velocity, α is the angle between the rotating pole and the axis pointing along the direction of the gravitational acceleration, and ω the angular velocity of the pole. The agent can push the cart left ($\leftarrow$) or right ($\rightarrow$) at every time step (in the negative or positive direction of the track) providing to the system a fixed momentum $|p|$. A pictorial representation of a state-action pair (s_t, a_t) can be found in Figure 4.1.

Let us suppose there exists a function $h : \mathcal{S} \times \mathcal{A} \to \mathcal{S} \times \mathcal{A}$ that maps a state-action pair (s_t, a_t) to $\big(k_\sigma(s_t), k_\alpha(a_t)\big)$, where $k_\sigma : \mathcal{S} \to \mathcal{S}$ and $k_\alpha : \mathcal{A} \to \mathcal{A}$, such that the dynamics of the pair (s, a) is the same as the one of $h(s, a)$. Note that in this example, the system dynamics is symmetric with respect to a flip around the vertical axis. In other words, its dynamics is invariant by multiplication by minus one, assuming that if $a = \leftarrow$ then $k_\alpha(a) = -a = \rightarrow$ and vice versa. Indeed, if the state-action pair (s_t, a_t) leads to the state s_{t+1}, this property will imply that $h(s_t, a_t) = (-s_t, -a_t)$ leads to $k_\sigma(s_{t+1}) = -s_{t+1}$.

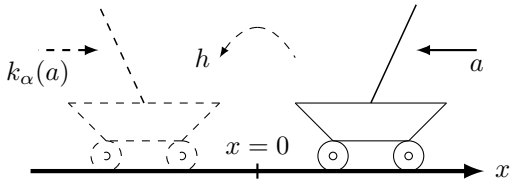


Figure 4.1: The cart in the right is a representation of a CartPole's state s_t with $x_t > 0$ and action $a_t = \leftarrow$. The dashed cart in the left is the image of (s_t, a_t) under the transformation h which inverses state $k_\sigma(s) = -s$ and action $k_\alpha(a) = -a$.

When learning the dynamics from a finite batch of experiences (or trajectories), resulting in a set of transitions $\mathcal{D} = \big\{(s_i, a_i, s_i')\big\}_{i=1}^{n}$ with $n \in \mathbb{N}$ being the size of the batch, we might, for instance, fit a function to predict the next state in a transition $\hat{s}(s, a) = s'$ to minimize a loss, *e.g.* the Mean Squared Error (MSE). However, imagine that in the batch $\mathcal{D}$ there were many transitions regarding the part of the state-action space with $x > 0$ and very few with $x < 0$. Unfortunately, we may learn a good model to forecast what will happen when the cart is at the right of the origin and a very poor model at its left side. We can then suppose that also a control policy will perform well when $x > 0$ and poorly when $x < 0$. Nevertheless, if it were possible to be confident of the existence of the symmetry $k_\sigma(s, a, s') = -s$ and $k_\alpha(s, a, s') = -a$ (where the opposite of the action a is the transformation stated above), we might extend the batch of experiences without additional interaction with the system, and then improve the accuracy of the model also to the regions where $x < 0$.

With this motivation in mind, in the next section, we present density estimation methods that assist in learning the likelihood of any transition given the starting data set.

4.1 Density estimation methods and Normalizing Flows

To verify the presence of a symmetry, a preliminary estimate of the transition model is needed. More specifically, we first perform a Probability Mass Function (pmf) estimation or a pdf estimation of the transitions in the batch $\mathcal{D}$ depending on the typology of the MDP we are tackling (respectively discrete or continuous). In the discrete case, this amounts to learning a set of discrete distributions. In the continuous case, we can use approaches of the literature to approximate the transition function, such as Normalizing Flows (Dinh, Krueger and Bengio, 2015; Kobyzev, Prince and Brubaker, 2020). A Normalizing Flow is a DNN architecture that allows us to approximate a pdf while being able to compute an analytically estimated density value for new samples. In this way, once a pmf/pdf has been estimated from the batch of transitions $\mathcal{D}$, we can compute the probability of an alleged symmetric transition that is supposed to be sampled from the same distribution. When the probability (or the density in the continuous case) is greater than a given threshold for a high fraction of samples, we decide to *trust* in the presence of this alleged symmetry and augment the batch by including the symmetric transitions. In the end, the dynamics of the model is learned over the augmented data set. When the approach detects a symmetry that is present in the true environment, the accuracy of the learned model increases; otherwise, the procedure could also result in detrimental effects.

Probability Mass Function estimation for discrete MDPs. Let $\mathcal{D} = \{(s_i, a_i, s'_i)\}_{i=1}^{n}$ be a batch of recorded transitions. Performing mass estimation over $\mathcal{D}$ amounts to compute the probabilities that define the unknown discrete transition distribution T by estimating the frequencies of transition in $\mathcal{D}$. In other words, we compute Eq. (1.14).

Probability Density Function estimation for continuous MDPs. Performing density estimation over $\mathcal{D}$ amounts to finding an analytical expression for the probability density of a transition (s, a, s') given $\mathcal{D}$: $\mathcal{L}(s, a, s'|\mathcal{D})$. Normalizing flows (Dinh, Krueger and Bengio, 2015; Kobyzev, Prince and Brubaker, 2020) allow defining a parametric flow of continuous transformations that reshapes a known pdf (*e.g.* a multivariate Gaussian) to one that best fits the data. Since the transformations are known, the Jacobians are computable at every step and the probability value can always be assessed (Dinh, Krueger and Bengio, 2015).

4.2 Proposed paradigm and flowchart

We propose a paradigm to check whether the dynamics of a to-be-learned (i) discrete MDP (see Algorithm 5) or (ii) a continuous MDP (see Algorithm 6) are endowed with the invariance with respect to some transformation. A pseudo representation of the flow chart of the algorithms is shown in Figure 4.2.

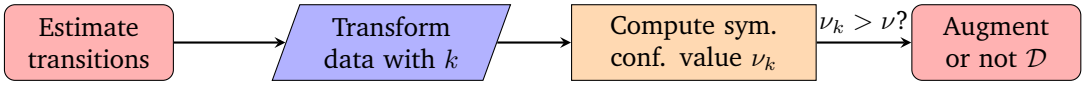


Figure 4.2: Pseudo flow chart of Algorithms 5 and 6.

Paradigm summary The paradigm can be summarized as follows:

1. An expert presumes that a to-be-learned model is endowed with the invariance of T with respect to a transformation k;

2. She/He computes the probability function estimation based on the batch $\mathcal{D}$:
   (a) (discrete case) She/He computes $\hat{T}$, an estimate of T, using the transitions in a batch $\mathcal{D}$ by applying Eq. (1.14);
   (b) (continuous case) She/He performs Density Estimation over $\mathcal{D}$ using Normalizing Flows;
3. She/He applies k to all transitions $(s, a, s') \in \mathcal{D}$ and then checks whether the symmetry confidence value ν_k
   (a) (discrete case) using Eq. (4.9) is smaller than the confidence threshold ν;
   (b) (continuous case) of probability values $\mathcal{L}$ evaluated on $k(\mathcal{D})$ exceeds a threshold θ that corresponds to the $q-$order quantile of the distribution of probability values evaluated on the original batch. The quantile order q is given as an input to the procedure by an expert (see Algorithm 6);
4. If the last condition is fulfilled, then $\mathcal{D}$ is augmented with $k(\mathcal{D})$.

Notice that

$$0 < d_k(s, a, s') \leq 1 \quad \forall (s, a, s') \in \mathcal{S} \times \mathcal{A} \times \mathcal{S}. \tag{4.3}$$

It is worth noting that since $\nu_k \in [0, 1]$, it can be interpreted as the probability that k is a symmetry. Therefore, one could consider ν as a classification threshold and set it, for example, to 0.5 (binary classification). Moreover, keep in mind that once a transformation k is detected as a symmetry, the data set is potentially augmented with transitions that are not present in the original batch, injecting hence unseen and completely novel information into the data set.

Explanation of the procedure

Let $\mathcal{M}$ be a deterministic MDP, let $\mathcal{D}$ be a batch of pre-collected transitions, and let k be an alleged symmetric transformation of $\mathcal{M}$'s dynamics.

Algorithm 5: Symmetry detection and data augmenting in a discrete MDP

Input: Batch of transitions $\mathcal{D}$, k alleged symmetry
Output: Possibly augmented batch $\mathcal{D} \cup \mathcal{D}_k$

1 $\hat{T} \leftarrow$ Most Likely Categorical pmf from $\mathcal{D}$
2 $\nu_k = 1 - \frac{1}{|\mathcal{D}|} \sum_{(s,a,s') \in \mathcal{D}} d_k(s, a, s')$ *(where d_k is defined in Equation 4.8)*
3 **if** $\nu_k > 0.5$ **then**
4 | $\mathcal{D}_k = k(\mathcal{D})$ *(alleged symmetric samples)*
5 | **return** $\mathcal{D} \cup \mathcal{D}_k$ *(the augmented batch)*
6 **else**
7 | **return** $\mathcal{D}$ *(the original batch)*
8 **end**

In order to check whether k can be considered or not as a symmetry of the dynamics, in the discrete case, we will first estimate the most likely set of transition distributions $\hat{T}$ given the batch of transitions (Line 1, Algorithm 5) while in the continuous case, we will estimate the density of transition in the batch obtaining the probability density value $\mathcal{L}(s, a, s'|\mathcal{D})$ (Line 1, Algorithm 6). In the continuous case, we will then compute the density value of every transition $(s, a, s') \in \mathcal{D}$, resulting in a set of real values from $\mathcal{L}$ denoted by Λ (Line 2, Algorithm 6). We will select the q-order quantile of Λ to be a threshold $\theta \in \mathbb{R}$ (Line 3, Algorithm 6) that will determine whether

we can trust the symmetry to be present, and hence to augment or not the starting batch. Next, we will map any $(s, a, s') \in \mathcal{D}$ to its alleged symmetric image $\big(k_\sigma(s, a, s'), k_\alpha(s, a, s'), k_{\sigma'}(s, a, s')\big)$ (Line 4, Algorithm 6). The map of $\mathcal{D}$ under the transformation k will be denoted as $\mathcal{D}_k$. Let then, $\forall (s, a, s') \in \mathcal{D}_k$, $\mathcal{L}(s, a, s'|\mathcal{D})$ be the probability density of the symmetric image of a transition in the batch. We will assume that the system dynamics is invariant under the transformation k if ν_k, the percentage of transitions whose density is greater than θ, is bigger than the percentage threshold ν (Lines 5-10, Algorithm 6). In the discrete case, a complex estimate of a distance between probability distributions is needed to assess if the sampled transitions would be likely to have been generated also after the application of an alleged symmetric transformation (see Subsection 4.3.1 for more details and Line 2 in Algorithm 5). If data augmenting is performed, the boosted batch $\mathcal{D} \cup \mathcal{D}_k$ will be returned as output.

This proposed paradigm offers a systematic way to check for the invariance of dynamics with respect to a given transformation, potentially allowing for improved learning and more efficient exploration of the state and action spaces. By identifying symmetries and augmenting the dataset with novel information, this method can contribute to more effective reinforcement learning algorithms and ultimately better decision-making processes in complex environments.

Algorithm 6: Symmetry detection and data augmenting in a continuous MDP with detection threshold $\nu = 0.5$

Input: Batch of transitions $\mathcal{D}$, $q \in [0, 1)$ order of the quantile, k alleged symmetry
Output: Possibly augmented batch $\mathcal{D} \cup \mathcal{D}_k$

1 $\mathcal{L} \leftarrow$ Density Estimate ($\mathcal{D}$) *(e.g. with Normalizing Flows)*
2 $\Lambda \leftarrow$ Distribution $\mathcal{L}(\mathcal{D})$ *($\mathcal{L}$ evaluated over $\mathcal{D}$)*
3 $\theta = q$-order quantile of Λ
4 $\mathcal{D}_k = k(\mathcal{D})$ *(alleged symmetric samples)*
5 $\nu_k = \dfrac{1}{|\mathcal{D}_k|} \displaystyle\sum_{(s,a,s') \in \mathcal{D}_k} \mathbb{1}_{\{\mathcal{L}(s,a,s'|\mathcal{D}) > \theta\}}$
6 **if** $\nu_k > 0.5$ **then**
7 | **return** $\mathcal{D} \cup \mathcal{D}_k$ *(the augmented batch)*
8 **else**
9 | **return** $\mathcal{D}$ *(the original batch)*
10 **end**

4.3 Expert-guided symmetry discovery

4.3.1 Discrete MDPs

For the method to work in both deterministic and stochastic environments, we need to measure the distance between distributions. Estimating a distance in distribution is considered in the version of the approach that addresses continuous environments, as learning a distribution over transitions represented by their features is independent of the nature of the dynamics. However, when dealing with categorical states, the notion of distance between features cannot be exploited.

We propose to compute the percentage ν_k relying on a distance between discrete probability distributions. Since the transformation k is a surjection on transition tuples, we do not know a-priori the correct mapping $k_{\sigma'}(s, a, s')\ \forall s' \in \mathcal{S}$. In other words, we can compute $k_{\sigma'}$, the symmetric image of s', only when we receive the entire tuple (s, a, s') as input, as an inverse mapping might not exist.

Therefore, we will resort to computing a *pessimistic* approximation of the Total Variational Distance (proportional to the L^1-norm). Specifically, given (s, a, s'), we aim to calculate the Chebyshev distance (the L^∞-norm) between $T(s, a, \cdot)$ and $T\big(k_\sigma(s, a, s'), k_\alpha(s, a, s'), \cdot\big)$. Recall that given two vectors of dimension d, x and y both $\in \mathbb{R}^d$, $||x - y||_\infty \leq ||x - y||_1$.

Let us then define the following four functions:

$$m(s, a, s') = \min_{\overline{s} \in \mathcal{S} \setminus \{s'\} : \hat{T} \neq 0} \hat{T}(s, a, \overline{s}) \tag{4.4}$$

$$M(s, a, s') = \max_{\overline{s} \in \mathcal{S} \setminus \{s'\}} \hat{T}(s, a, \overline{s}), \tag{4.5}$$

$$m_k(s, a, s') = \min_{\substack{\overline{s} \in S \text{ s.t.} \\ \overline{s} \neq k_{\sigma'}(s,a,s') \\ \text{and } \hat{T} \circ k \neq 0}} \hat{T}\big(k_\sigma(s, a, s'), k_\alpha(s, a, s'), \overline{s}\big), \tag{4.6}$$

$$M_k(s, a, s') = \max_{\substack{\overline{s} \in S \text{ s.t.} \\ \overline{s} \neq k_{\sigma'}(s,a,s')}} \hat{T}\big(k_\sigma(s, a, s'), k_\alpha(s, a, s'), \overline{s}\big) \tag{4.7}$$

where m (M) and m_k (M_k) are the minimum (maximum) of the probability mass function (pmf) $\hat{T}$ when evaluated respectively on an initial state and action (s, a) and $\big(k_\sigma(s, a, s'), k_\alpha(s, a, s')\big)$ for which $\hat{T} \neq 0$. To make Equations 4.4 to 4.7 work even on deterministic or sparse $\hat{T}$, instead of excluding zero values we add a small pseudo-count $0 < \varepsilon \leq 10^{-6}$ to all possible transitions before normalization when learning $\hat{T}$ (line 1 of Algorithm 5).

To approximate the Chebyshev distance between $\hat{T}(s, a, \cdot)$ and $\hat{T}(k_\sigma(s, a, s'), k_\alpha(s, a, s'), \cdot)$, we define a pessimistic approximation d_k as follows:

$$\begin{aligned} d_k(s, a, s') = \max\Big\{ & \underbrace{\big|M(s, a, s') - m_k(s, a, s')\big|}_{(I)}, \\ & \underbrace{\big|M_k(s, a, s') - m(s, a, s')\big|}_{(II)}, \\ & \underbrace{\big|\hat{T}(s, a, s') - (\hat{T} \circ k)(s, a, s')\big|}_{(III)} \Big\}. \end{aligned} \tag{4.8}$$

For the moment, consider $\hat{T}(s, a, \cdot)$ and $\hat{T}\big(k_\sigma(s, a, s'), k_\alpha(s, a, s'), \cdot\big)$ just as two sets of numbers. Remove the value corresponding to s' from the first set and the one corresponding to $k_{\sigma'}(s, a, s')$ from the second set. Taking the max between *(I)* and *(II)* just equates to selecting the maximum possible difference between any two values of these modified sets. Equation 4.8 tells us to select the worst possible case since we do not know which permutations of states we should compare when computing the Chebyshev distance.
We remove s' from $\hat{T}(s, a, \cdot)$ and $k_{\sigma'}(s, a, s')$ from $\hat{T}\big(k_\sigma(s, a, s'), k_\alpha(s, a, s'), \cdot\big)$ because we hypothesize that k maps (s, a, s') to $\big(k_\sigma(s, a, s'), k_\alpha(s, a, s'), k_{\sigma'}(s, a, s')\big)$, and thus, we can compare those values directly *(III)*.

In detail, the symmetry confidence value ν_k used in Line 2 of Algorithm 5 is defined as:

$$\nu_k(\mathcal{D}) = 1 - \frac{1}{|\mathcal{D}|} \sum_{(s,a,s') \in \mathcal{D}} d_k(s, a, s'). \tag{4.9}$$

Extreme case scenario *Is Equation 4.8 too pessimistic?* Consider that for a given state action couple $(\overline{s}, \overline{a})$, we have a transition distributed over 3 states $s \in \mathcal{S} = \{\textit{One}, \textit{Two}, \textit{Three}\}$ with probabilities $T(\overline{s}, \overline{a}, One) = 0.01$, $T(\overline{s}, \overline{a}, Two) = 0.01$ and $T(\overline{s}, \overline{a}, Three) = 0.98$. Now, assume the

estimate of the transition function is perfect. Does the distance in Equation 4.8 converge to 0? Not always, but what matters for the detection of symmetries is the average of the distances over the entire batch (Eq. 4.9). Suppose that these probabilities were inferred from a batch with the transition $(\overline{s}, \overline{a}, \textit{One})$ once, $(\overline{s}, \overline{a}, \textit{Two})$ once and $(\overline{s}, \overline{a}, \textit{Three})$ ninety-eight times. Consider $(\overline{s}, \overline{a}, \textit{Three})$. $M(\overline{s}, \overline{a}, \textit{Three}) = M_k(\overline{s}, \overline{a}, \textit{Three}) = m(\overline{s}, \overline{a}, \textit{Three}) = m_k(\overline{s}, \overline{a}, \textit{Three}) = 0.01$. Following Eq. (4.8), $d_k(\overline{s}, \overline{a}, \textit{Three}) = 0$. However, $d_k(\overline{s}, \overline{a}, \textit{One}) = d_k(\overline{s}, \overline{a}, \textit{Two}) = 0.97$, which is a too pessimistic estimate. Nonetheless, let us calculate ν_k (Eq.4.9). For this state-action pair $(\overline{s}, \overline{a})$, the average over the batch is: $(d_k(\overline{s}, \overline{a}, \textit{One}) + d_k(\overline{s}, \overline{a}, \textit{Two}) + 98 d_k(\overline{s}, \overline{a}, \textit{Three}))/100 = 0.0194$. If the estimation is the same for other pairs (s, a), then $\nu_k = 1 - 0.0194 = 0.9806$. This is a value close to 1, suggesting k is a symmetry.

4.3.2 Continuous MDPs

Figure 4.3 provides an intuition about Algorithm 6. The intuition behind the approach in the

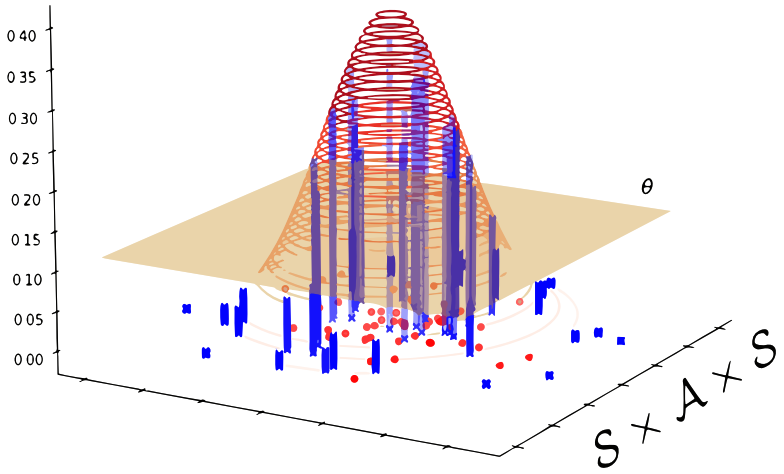


Figure 4.3: Intuition for the continuous case. The xy plane is the space of transitions $\mathcal{S} \times \mathcal{A} \times \mathcal{S}$, the z axis is $\mathcal{L}$, the value of the probability density of a given transition. The red points represent $\mathcal{D}$, the blue crosses $\mathcal{D}_k$ for a given transformation k. We display as a red contour plot the pdf $\mathcal{L}$ learned in Line 1 of Algorithm 6. The orange hyperplane has height θ which is the threshold computed in Line 3 of Algorithm 6. The blue vertical bars have as height the value of $\mathcal{L}$ evaluated for that specific transition. The algorithm counts the fraction ν_k of samples (blue crosses) that have a vertical bar higher than the hyperplane.

continuous case is that if in the original batch $\mathcal{D}$ the density of transitions that are *"not so different"* from some of the symmetric images $\mathcal{D}_k$ of $\mathcal{D}$, then $\mathcal{L}(\mathcal{D}_k|\mathcal{D})$ will not be *"too small"*. How small is small when we are considering real-valued, continuous pdf? In order to insert a comparable scale, we take the threshold θ to be a q-quantile of the set of the estimated density values of the transitions in the original batch $\mathcal{D}$, *i.e.* $\{\mathcal{L}(s, a, s' \mid \mathcal{D}) \mid (s, a, s') \in \mathcal{D}\}$. It goes without saying that since the purpose of Algorithm 6 is to perform data augmentation, it is necessary to select a small q-order quantile; otherwise, the execution of the procedure would serve no practical purpose. It would be pointless to augment the batch with transitions that are already very likely in the original one (see Figure 4.4). In this case, we would not insert any new information.

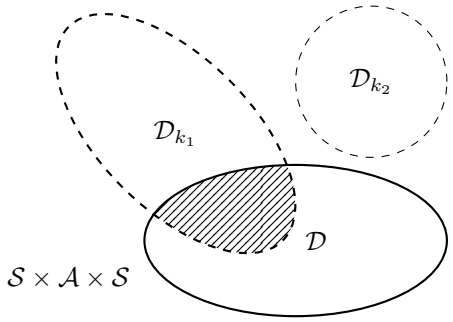


Figure 4.4: Representation of the support in $\mathcal{S}^2 \times \mathcal{A}$ of the of $\mathcal{D}$, $\mathcal{D}_{k_1}$ and $\mathcal{D}_{k_2}$. k_1 and k_2 are two different alleged transformations. The shape and position of the sets are determined by the log-likelihood of the density estimate $\mathcal{L}$ and the quantile threshold θ. It is worth noting that $\mathcal{D}_{k_1} \cap \mathcal{D} \neq \varnothing$ and $\mathcal{D}_{k_2} \cap \mathcal{D} = \varnothing$. If the user-chosen percentage threshold ν is taken into account, then k_1 may be detected as a symmetry while k_2 is not. If k_1 is indeed detected as a symmetry, then augmenting the data in $\mathcal{D}$ with $\mathcal{D}_{k_1}$ involves training the model on the combined dataset $\mathcal{D} \cup \mathcal{D}_{k_1}$. It is important to note that the data in $\mathcal{D}_{k_1} \backslash \mathcal{D}$ are not present in the original batch $\mathcal{D}$.

4.4 Experiments

We test the algorithm in one discrete grid environment, with and without periodic boundary conditions, and in a *stochastic* version of two famous environments of OpenAI's Gym Learning Suite: CartPole and Acrobot.

4.4.1 Environments

In the next subsections, we describe the environments and the proposed transformations k. It is worth noting that sometimes we use a contracted notation to indicate k_α. Let a_1, a_2, a_3, a_4 represent some actions $\in \mathcal{A}$, for semplicity we denote $g(a_1, a_2, \dots) = (a_3, a_4, \dots)$ to indicate $k_\alpha(s, a_1, s') = a_3, k_\alpha(s, a_2, s') = a_4, \dots$, etc.

Discrete environments

Stochastic Toroidal Grid In this environment, the agent can move along fixed directions over a torus by acting with any $a \in \mathcal{A} = \{\uparrow, \downarrow, \leftarrow, \rightarrow\}$. However, when intending to head towards

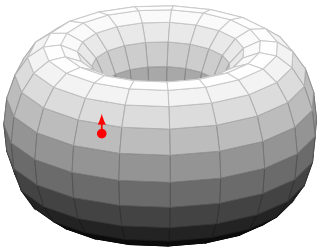


Figure 4.5: Representation of the Grid Environment. The red dot is the position of a state s on the torus. The displacement obtained by acting with action $a = \uparrow$ is shown as a red arrow.

a chosen direction, the agent could slip and end up somewhere else (with a fixed probability), *i.e.* in the opposite direction, to its left, or to its right. We conducted the experiments in four different variants of this environment characterized by different transition probabilities (see Table

4.1). In the first variant, when performing an action, the agent has a 60% chance of moving to the intended direction, a 20% chance to move to the opposite one, and a 10% chance along an orthogonal direction. In the second variant, when acting, the agent has a 75% chance of moving to the intended direction, a 15% chance to move to the opposite one, and a 5% chance along an orthogonal direction. In the third variant, the probabilities are, respectively, 90%, 5%, and $2,5\%$. In the fourth variant, the agent does not slip and always can move to the wanted direction with a 100% probability. For every variant, we collect $z = 10$ sets of $M = 100$ batches, with respectively

Table 4.1: Stochastic Toroidal Grid: transition probabilities for the four variants.

Variant	Intended direction	Opposite direction	Orthogonal direction
1	60%	20%	10%
2	75%	15%	5%
3	90%	5%	2.5%
4	100%	0%	0%

$N = 1000 \times i_z$ steps in each batch (i_z going from 1 to z). The positions on the torus are the states $s = (i, j)$, and the set $\mathcal{S}$ is represented as a grid of fixed side $l = 10$ and periodic boundary conditions (see Figure 4.5). Since there are no obstacles, this environment is endowed with many symmetric transformations and therefore can serve as a useful proof-of-concept.
We tested Algorithm 5 with six different alleged transformations k in a Grid with size $l^2 = 100$ over $N = 50$ different simulations.

1. **Time reversal symmetry with action inversion (TRSAI).** Assuming that $\downarrow$ is the reverse of $\uparrow$ and $\leftarrow$ is the reverse of $\rightarrow$, we proposed the following transformation: $k = \big(k_\sigma(s, a, s') = s'$, $g(\uparrow, \downarrow, \leftarrow, \rightarrow) = (\downarrow, \uparrow, \rightarrow, \leftarrow)$, $k_{\sigma'}(s, a, s') = s\big)$.

2. **Same dynamics with action inversion (SDAI).** $k = \big(k_\sigma(s, a, s') = s,\ g(\uparrow, \downarrow, \leftarrow, \rightarrow) = (\downarrow, \uparrow, \rightarrow, \leftarrow)$, $k_{\sigma'}(s, a, s') = s'\big)$.

3. **Opposite dynamics and action inversion (ODAI):** $k = \big(k_\sigma(s, a, s') = s,\ g(\uparrow, \downarrow, \leftarrow, \rightarrow) = (\downarrow, \uparrow, \rightarrow, \leftarrow)$, $k_{\sigma'}(s, a, s') = s' \mp (2, 0) \vee (0, 2)\big)$. In other words, we revert the action but also the final state is changed to reproduce the correct destination.

4. **Opposite dynamics but wrong action (ODWA).** The alleged transformation is like the one of Point 3, but the action is switched on the wrong axis, *e.g.* $g(\uparrow) = \rightarrow$).

5. **Translation invariance (TI).** $k = \big(k_\sigma(s, a, s') = s', g(\uparrow, \downarrow, \leftarrow, \rightarrow) = (\uparrow, \downarrow, \leftarrow, \rightarrow), k_{\sigma'}(s, a, s') = s' \pm (1, 0) \vee (0, 1)\big)$.

6. **Translation invariance with opposite dynamics (TIOD).** In this case, the action is the same as Point 5, but the agent returns to the previous state.

The said transformations are resumed in Table 4.2.

Deterministic Grid with boundaries This environment is a classic deterministic Grid with side $l = 10$ and $l^2 = 100$ the number of cells. As in the previous case, the agent can move along fixed directions by acting with any $a \in \mathcal{A} = \{\uparrow, \downarrow, \leftarrow, \rightarrow\}$. The outcome of the actions is deterministic, *i.e.* every action has a 100% probability of success. It is worth noting that this grid has boundaries, and therefore when the agent will try to cross any boundaries, it will instead stay in the same spot. The valid symmetries proposed for the Stochastic Grid environment are, in this scenario, only approximately valid, since they are broken for states next to the boundaries, but valid anywhere

Table 4.2: Stochastic Toroidal Grid: proposed transformations and label.

k	Label
$k_\sigma(s,a,s') = s'$ $k_\alpha\big(s, a=(\uparrow,\downarrow,\leftarrow,\rightarrow), s'\big) = (\downarrow,\uparrow,\rightarrow,\leftarrow)$ $k_{\sigma'}(s,a,s') = s$	**TRSAI**
$k_\sigma(s,a,s') = s$ $k_\alpha\big(s, a=(\uparrow,\downarrow,\leftarrow,\rightarrow), s'\big) = (\downarrow,\uparrow,\rightarrow,\leftarrow)$ $k_{\sigma'}(s,a,s') = s'$	SDAI
$k_\sigma(s,a,s') = s$ $k_\alpha\big(s, a=(\uparrow,\downarrow,\leftarrow,\rightarrow), s'\big) = (\downarrow,\uparrow,\rightarrow,\leftarrow)$ $k_{\sigma'}(s, a=(\uparrow,\downarrow,\leftarrow,\rightarrow), s') =$ $\big(s'-(0,2), s'+(0,2), s'+(2,0), s'-(2,0)\big)$	**ODAI**
$k_\sigma(s,a,s') = s$ $k_\alpha(s, a=(\uparrow,\downarrow,\leftarrow,\rightarrow), s') = (\rightarrow,\leftarrow,\uparrow,\downarrow)$ $k_{\sigma'}(s, a=(\uparrow,\downarrow,\leftarrow,\rightarrow), s') =$ $\big(s'-(0,2), s'+(0,2), s'+(2,0), s'-(2,0)\big)$	ODWA
$k_\sigma(s,a,s') = s'$ $k_\alpha(s,a,s') = a$ $k_{\sigma'}\big(s, a=(\uparrow,\downarrow,\leftarrow,\rightarrow), s'\big) =$ $\big(s'+(0,1), s'-(0,1), s'-(1,0), s'+(1,0)\big)$	**TI**
$k_\sigma(s,a,s') = s'$ $k_\alpha(s,a,s') = a$ $k_{\sigma'}(s,a,s') = s$	TIOD

else. After collecting data in the same fashion as for the Stochastic Toroidal Grid environment, we tested Algorithm 5 with the very same six different alleged transformations k over $N = 50$ different simulations.

Continuous environments

Stochastic CartPole As previously stated, the dynamics of CartPole is invariant with respect to the transformation $k = (k_\sigma(s,a,s') = -s, k_\alpha(s,a,s') = -a, k_{\sigma'}(s,a,s') = -s) \forall (s,a,s') \in \mathcal{S}\times\mathcal{A}\times\mathcal{S}$. In order to use Algorithm 6, we first map the actions to real numbers: $\leftarrow = -1.5$ and $\rightarrow = 1.5$. We then normalize every state feature in the range $[-1.5, 1.5]$. The dynamics of Stochastic CartPole is similar to that of CartPole (Brockman et al., 2016), however, the force that the agent uses to push the cart is sampled from a normal distribution with mean f (the force defined in the deterministic version) and standard deviation $\tilde{\sigma} = 2$. We tested Algorithm 6 with four different alleged transformations k over $N = 5$ different simulations, a batch of size $|\mathcal{D}| = 10^3$ collected with a random policy, and a quantile order to compute the thresholds $q = 0.1$.

1. **State and action reflection with respect to an axis in $x = 0$ (SAR).** Assuming that $\leftarrow$ is the reverse of $\rightarrow$ we proposed the following transformation: $k = \big(k_\sigma(s,a,s') = -s,\ g(\leftarrow,\rightarrow) = (\rightarrow,\leftarrow),\ k_{\sigma'}(s,a,s') = -s'\big)$.

2. **Initial State Reflection (ISR).** We then tried the same transformation as before but without reflecting the next state s': $k = \big(k_\sigma(s,a,s') = -s,\ g(\leftarrow,\rightarrow) = (\rightarrow,\leftarrow),\ k_{\sigma'}(s,a,s') = s'\big)$.

3. **Action Inversion (AI).** What about reversing only the actions? $k = \big(k_\sigma(s,a,s') = s,\ g = (\leftarrow,\rightarrow) = (\rightarrow,\leftarrow),\ k_{\sigma'}(s,a,s') = s'\big)$.

4. **Single Feature Inversion (SFI).** We also tried to reverse only one single feature of the starting state: $k = \big(k_\sigma(s=(x,v,\alpha,\omega),a,s') = (-x,v,\alpha,\omega),\ g(\leftarrow,\rightarrow) = (\leftarrow,\rightarrow),\ k_{\sigma'}(s,a,s') = s'\big)$.

5. **Translation Invariance (TI).** We translated the position of the initial state x and that of the final state x' by an arbitrary value (0.3): $k = \big(k_\sigma(s = (x, v, \alpha, \omega), a, s') = (x + 0.3, v, \alpha, \omega)$, $g(\leftarrow, \rightarrow) = (\leftarrow, \rightarrow)$, $k_{\sigma'}(s, a, s' = (x', v', \alpha', \omega')) = (x' + 0.3, v', \alpha', \omega')\big)$.

The proposed transformations are resumed in Table 4.3.

Table 4.3: Stochastic CartPole. Proposed transformations and labels.

k	Label
$k_\sigma(s, a, s') = -s$ $k_\alpha\big(s, a = (\leftarrow, \rightarrow), s'\big) = (\rightarrow, \leftarrow)$ $k_{\sigma'}(s, a, s') = -s'$	**SAR**
$k_\sigma(s, a, s') = -s$ $k_\alpha(s, a, s') = a$ $k_{\sigma'}(s, a, s') = s'$	ISR
$k_\sigma(s, a, s') = s$ $k_\alpha\big(s, a = (\leftarrow, \rightarrow), s'\big) = (\rightarrow, \leftarrow)$ $k_{\sigma'}(s, a, s') = s'$	AI
$k_\sigma\big(s = (x, ...), a, s'\big) = (-x, ...)$ $k_\alpha(s, a, s') = a$ $k_{\sigma'}(s, a, s') = s'$	SFI
$k_\sigma\big(s = (x, ...), a, s'\big) = (x + 0.3, ...)$ $k_\alpha(s, a, s') = a$ $k_{\sigma'}\big(s, a, s' = (x', ...)\big) = (x' + 0.3, ...)$	**TI**

Stochastic Acrobot The Acrobot environment consists of two poles linked with a rotating joint at one end. One of the poles is pinned to a wall with a second rotating joint (see Figure 4.6). The system is affected by gravity and hence the poles are hanging down. An agent can apply

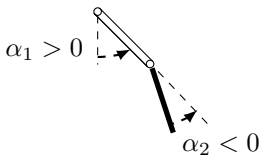


Figure 4.6: Representation of a state of the Acrobot environment.

a negative torque to the lower pole ($a = -1$), a positive one ($a = 1$), or do nothing ($a = 0$). The goal is to push the lower pole as high as possible. The state consists of the sine and cosine of the two rotational joint angles (α_1, α_2) and the joint angular velocities (ω_1, ω_2) : $s = (\sin\alpha_1, \cos\alpha_1, \sin\alpha_2, \cos\alpha_2, \omega_1, \omega_2)$. The dynamics is invariant under the transformation $k = (k_\sigma(s = (\alpha_1, \alpha_2, \omega_1, \omega_2), a, s') = (-\alpha_1, -\alpha_2, -\omega_1, -\omega_2)$ and $k_\alpha(s, a, s') = -a$,

$$k_{\sigma'}(s, a, s' = (\alpha_1', \alpha_2', \omega_1', \omega_2')) = (-\alpha_1', -\alpha_2', -\omega_1', -\omega_2'))\forall(s, a, s') \in \mathcal{S} \times \mathcal{A} \times \mathcal{S}.$$

To apply Algorithm 6 we first normalize the state features and the action in the interval $[-3, 3]$. Stochastic Acrobot is the very same Acrobot of (ibid.) but at every time step a noise ϵ is sampled from a uniform distribution on the interval $[-0.5, 0.5]$ and added to the torque. We tested Algorithm 6 with four different alleged transformations k over $N = 5$ different simulations, a batch of size $|\mathcal{D}| = 10^3$ collected with a random policy, and a quantile order to compute the thresholds $q = 0.1$. The label of the transformations hereafter explained are resumed in Table 4.4.

1. **Angles and Angular Velocities Inversion (AAVI).**
   $k = \big(k_\sigma(s = (\sin\alpha_1, \sin\alpha_2, \cos\alpha_1, \cos\alpha_2, \omega_1, \omega_2), a, s') = (-\sin\alpha_1, -\sin\alpha_2, \cos\alpha_1, \cos\alpha_2, -\omega_1, -\omega_2)$,

Table 4.4: Acrobot. Proposed transformations and labels.

k	Label
$k_\sigma\big(s=(s_1,s_2,\omega_1,\omega_2,\dots),a,s'\big)$ $=(-s_1,-s_2,-\omega_1,-\omega_2,\dots)$ $k_\alpha(s,a=(-1,0,1),s')=(1,0,-1)$ $k_{\sigma'}\big(s,a,s'=(s'_1,s'_2,\omega'_1,\omega'_2,\dots)\big)$ $=(-s'_1,-s'_2,-\omega'_1,-\omega'_2,\dots)$	**AAVI**
$k_\sigma\big(s=(c_1,c_2,\omega_1,\omega_2,\dots),a,s'\big)$ $=(-c_1,-c_2,-\omega_1,-\omega_2,\dots)$ $k_\alpha\big(s,a=(-1,0,1),s'\big)=(1,0,-1)$ $k_{\sigma'}\big(s,a,s'=(c'_1,c'_2,\omega'_1,\omega'_2,\dots)\big)$ $=(-c'_1,-c'_2,-\omega'_1,-\omega'_2,\dots)$	CAVI
$k_\sigma(s,a,s')=s$ $k_\alpha\big(s,a=(-1,0,1),s'\big)=(1,0,-1)$ $k_{\sigma'}(s,a,s')=s'$	AI
$k_\sigma(s,a,s')=-s$ $k_\alpha\big(s,a,s'\big)=a$ $k_{\sigma'}(s,a,s')=s'$	SSI

$k_\alpha(s,a,s')=-a, k_{\sigma'}(s,a,s'=(\sin\alpha'_1,\sin\alpha'_2,\cos\alpha'_1,\cos\alpha'_2,\omega'_1,\omega'_2))$
$=(-sin\alpha'_1,-\sin\alpha'_2,\cos\alpha'_1,\cos\alpha'_2,-\omega'_1,-\omega'_2)\big)$.

2. **Cosines and Angular Velocities Inversion (CAVI).**
$k=\big(k_\sigma(s=(\sin\alpha_1,\sin\alpha_2,\cos\alpha_1,\cos\alpha_2,\omega_1,\omega_2),a,s')=(\sin\alpha_1,\sin\alpha_2,-\cos\alpha_1,-\cos\alpha_2,-\omega_1,-\omega_2),$
$k_\alpha(s,a,s')=-a, k_{\sigma'}(s,a,s'=(\sin\alpha'_1,\sin\alpha'_2,\cos\alpha'_1,\cos\alpha'_2,\omega'_1,\omega'_2))$
$=(\sin\alpha'_1,\sin\alpha'_2,-cos\alpha'_1,-\cos\alpha'_2,-\omega'_1,-\omega'_2)\big)$.

3. **Action Inversion (AI).** $k=\big(k_\sigma(s,a,s')=s, k_\alpha(s,a,s')=-a, k_{\sigma'}(s,a,s')=s'\big)$.

4. **Starting State Inversion (SSI).** $k=\big(k_\sigma(s,a,s')=-s, k_\alpha(s,a,s')=a, k_{\sigma'}(s,a,s')=s'\big)$.

4.4.2 Setup

We first collect a batch of transitions $\mathcal{D}$ by acting in the environment with a uniform random policy. We suppose the presence of a symmetry k and we try to detect it using Algorithm 5 or Algorithm 6. We report $\overline{\nu}_k$, the average value plus or minus the standard deviation of the quantity ν_k, computed over an ensemble of N different iterations of the procedure (using N distinct batches $\mathcal{D}$). We set the confidence threshold $\nu = 0.5$ since ν_k is normalized between 0 and 1, and can be interpreted as the binary probability of detecting or not a symmetry. However, distinct thresholds ν may yield varying results.

We show the real gain in performance when the policies computed for the model learned using the (augmented) data set eventually deployed in the real environment. The performance metrics are defined in the next paragraphs.

Evaluation of the performance (discrete case) In the end, let ρ be the distribution of initial states $s_0 \in S$ and let the performance u^π of a policy π be $u^\pi = \mathbb{E}_{s\sim\rho}[V^\pi(s)]$. We consider as performance difference the quantity $\Delta U = u^{\hat{\pi}_k} - u^{\hat{\pi}}$. In *discrete* environments, the policies are obtained with Policy Iteration and evaluated with Policy Evaluation.

Evaluation of the performance (continuous case) In *continuous* environments, Offline Learning is not trivial. With the aim of showing that batch augmentation through symmetry detection

is beneficial, we resort to two Model-Free Deep RL architectures: DQN (Mnih, Kavukcuoglu et al., 2015) and CQL (Kumar, Zhou et al., 2020) of the d3rlpy learning suite (Takuma Seno, 2021) to obtain a policy starting from the batches. The first method originally established the validity of Deep RL and is used in online RL, while the second was specifically developed to address offline RL problems. Convergence of Deep RL training is heavily dependent on hyperparameter tuning, which in turn relies on both the environment and the batch (Paine et al., 2020). Therefore, we apply DQN and CQL using the default parameters provided by d3rlpy, adhering more closely to an offline learning setting. This implies that the learning may not always converge to a good policy. We find this philosophy more honest than showing the results obtained with the best seed or the finest-tuned hyperparameters. Each architecture is trained for a number of steps equal to fifty times the number of transitions present in the batch.

4.4.3 Results

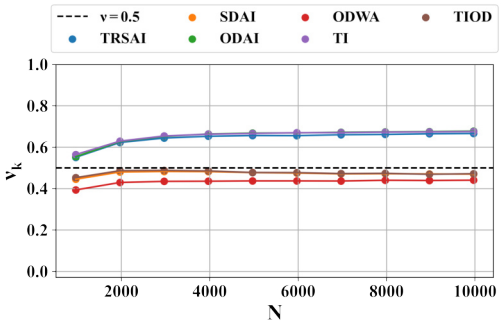
Discrete environments

Stochastic Toroidal Grid *Detection phase (ν_k).* The proposed algorithm perfectly manages to identify the real symmetries of the environment (see Figure 4.7): $\nu_k > 0.5, \forall k \in \{\text{TRSAI}, \text{ODAI}, \text{TI}\}$. Moreover, there are no false positives: $\nu_k < 0.5, \forall k \in \{\text{SDAI}, \text{ODWA}, \text{TIOD}\}$. We observe that in the first variant, ν_k for a true symmetry saturates around 0.7 as the batch size increases, while ν_k is slightly less than 0.5 for incorrect symmetries (see Figure 4.7a). Increasing the probability of moving in the right direction raises the saturation level of ν_k for correct symmetries: 0.75 for the second variant (see Figure 4.7b), 0.8 for the third variant (see Figure 4.7c), and 1 in the case of a deterministic environment (see Figure 4.7d). Similarly, ν_k for an incorrect symmetric transformation decreases when the probability of moving in the right direction increases. In the deterministic variant, $\nu_k = 0$ for all k that are not symmetries.

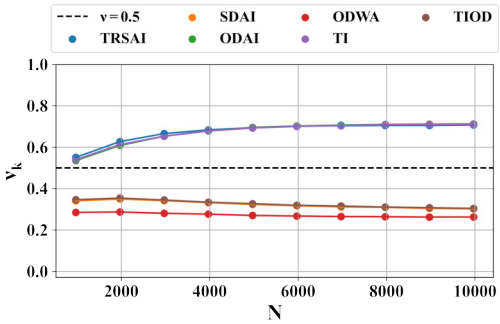
Evaluation of performance gain (ΔU). The difference in the performance of the deployed policies, ΔU, aligns well with the expected behavior. When k is a symmetry, $\Delta U > 0$ and approaches 0 as N increases. Conversely, when k is not a symmetric transformation of the dynamics, $\Delta U < 0$ and continues to decrease with N (see Figure 4.8a). In the low samples regime, the performance gain is the highest for the environment with the less deterministic transformations, as more samples would be needed to estimate the correct transition function. In this case, data augmentation using valid symmetric samples has the most significant impact on the performance of the learned policy.

Deterministic Grid with boundaries *Detection phase (ν_k).* Figure 4.9a demonstrates that the symmetries which are only approximately valid, such as TRSAI, ODAI, and TI, are detected with a lower ν_k compared to their "deterministic toroidal counterpart" (see Figure 4.7d). This result is not surprising, considering the manner in which ν_k is computed.

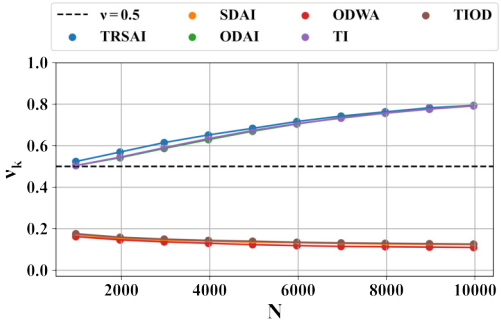
Evaluation of performance gain (ΔU). Although the algorithm identifies TRSAI, ODAI, and TI as valid symmetries, Figure 4.9b shows that using data augmentation with symmetric transitions to enhance planning performance in this environment is, on average, detrimental. We hypothesize that while data augmentation reduces the distributional shift for state-action pairs far from the grid borders, it actually increases the error for state-action pairs at the border. This effect leads to an overall decrease in performance if the computed policy relies on the negatively affected state-action pairs.



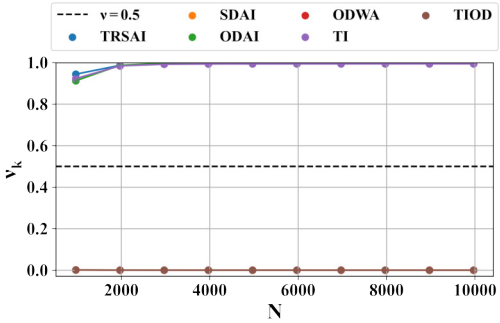
(a) First variant: movement in the intended direction with 60% probability.



(b) Second variant: movement in the intended direction with 75% probability.

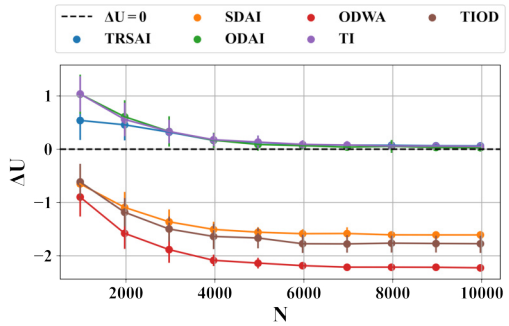


(c) Third variant: movement in the intended direction with 90% probability.

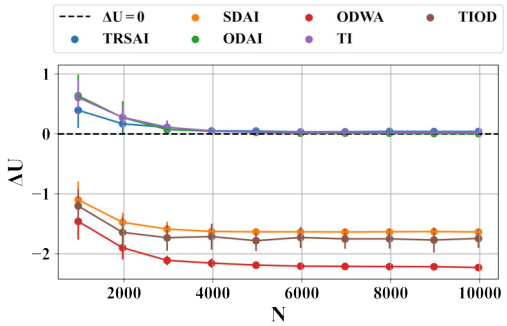


(d) Fourth variant: deterministic environment.

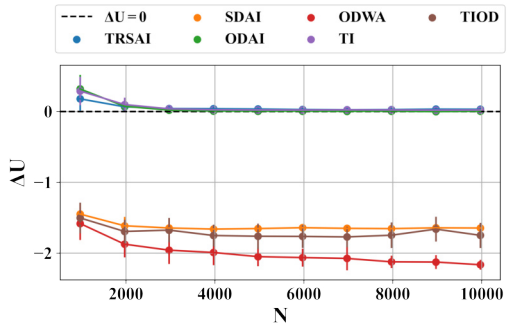
Figure 4.7: Stochastic Toroidal Grid Environment. ν_k for every variant and for every transformations k computed over sets of 100 different batches of size N. Points are mean values and bars standard deviations.



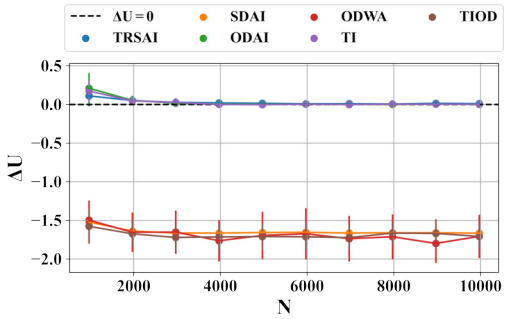
(a) First variant: movement in the intended direction with 60% probability.



(b) Second variant: movement in the intended direction with 75% probability.

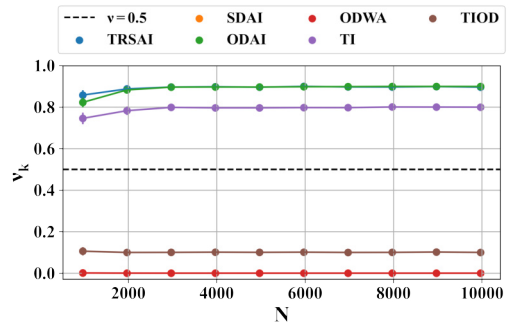


(c) Third variant: movement in the intended direction with 90% probability.

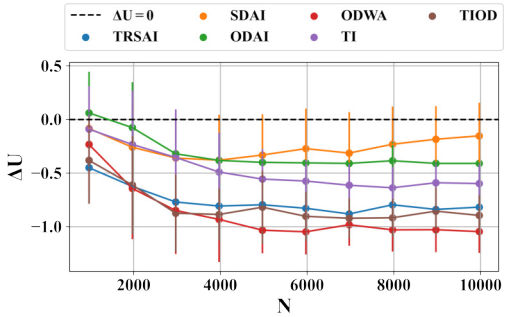


(d) Fourth variant: deterministic environment.

Figure 4.8: Stochastic Toroidal Grid Environment. Performance difference ΔU. The threshold at $\Delta U = 0$ is displayed as a dashed line. $\Delta U > 0$ means that data augmenting leads to better policies.



(a) Deterministic Grid with boundaries. Probability of symmetry ν_k. The threshold at $\nu = 0.5$ is displayed as a dashed line. $\nu_k > 0.5$ means that the transformation is detected as a symmetry.



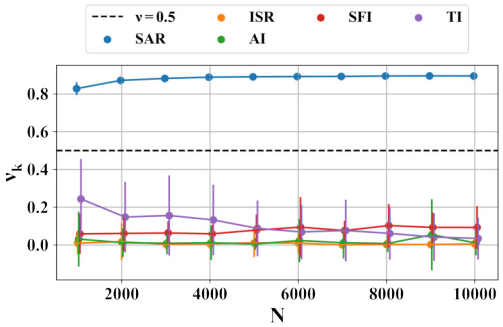
(b) Performance difference ΔU. The threshold at $\Delta U = 0$ is displayed as a dashed line. $\Delta U > 0$ means that data augmenting leads to better policies.

Figure 4.9: Deterministic Grid with boundaries. Detection phase and evaluation of performance gain.

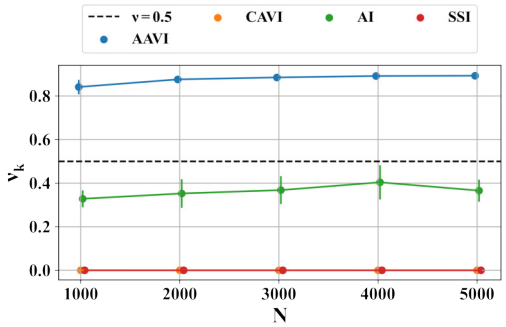
Continuous environments

Stochastic CartPole *Detection phase (ν_k)* In Stochastic CartPole, the algorithm fails to detect the symmetry $k =$ TI. This could be because the translation invariance symmetry in this case is fixed for a specific value (see TI in Table 4.3 where the translation is set at 0.3). If the translation is too small, the neural network struggles to discern the transformation from the noise. The algorithm correctly classifies $k =$ SAR as a symmetry and the remaining transformations as non-symmetries (see Figure 4.10a).

Evaluation of performance gain (ΔU). Results are displayed in Table 4.5. Offline RL is highly unstable and sensitive to the choice of hyperparameters. Furthermore, the training is conducted for a fixed number of epochs. We observe that on average, across different batch sizes, $\Delta U > 0$ for DQN and SAR, and SFI transformations. While SAR is a valid symmetry, SFI is not. A more conservative algorithm like CQL more readily exploits SAR. The performance difference for TI, both for DQN and CQL, is so close to zero that we believe augmenting the data set with this symmetry might not provide a significant advantage over using only the information contained in the original batch.



(a) Stochastic CartPole. Probability of symmetry ν_k. The threshold at $\nu = 0.5$ is displayed as a dashed line. $\nu_k > 0.5$ means that the transformation is detected as a symmetry.



(b) Stochastic Acrobot. Probability of symmetry ν_k. The threshold at $\nu = 0.5$ is displayed as a dashed line. $\nu_k > 0$ means that the transformation is detected as a symmetry.

Figure 4.10: ν_k, for the transformations k computed over sets of different batches of size N in Stochastic CartPole (up) and Stochastic Acrobot (down). Points are mean values and are a bit shifted horizontally for the sake of display. Standard deviation is displayed as a vertical error bar.

Stochastic Acrobot *Detection phase (ν_k).* In this environment, the only real symmetry of the dynamics, *AAVI*, gets successfully detected by the algorithm with $q = 0.1$. Non-symmetries yield a $\nu_k < 0.5$ (Figure 4.10b).

Evaluation of performance gain (ΔU). Results are displayed in Table 4.6 and indicate that the training in Stochastic Acrobot is more challenging than in Stochastic CartPole. Even with a large data set, the algorithms sometimes fail to learn a good policy. In particular, while CQL manages to learn appropriate behavior in the environment by exploiting the **AAVI** symmetry (average $\Delta U = 52.9$), DQN still struggles with every k, both correct and incorrect. However, CQL also benefits from augmenting the data set with wrong symmetries, albeit to a lesser extent. We suspect that this effect is due to the instability in Offline RL training.

Table 4.5: ΔU for every alleged symmetry in Stochastic CartPole with two baselines and different batch sizes N.

		N (number of transitions in the original batch)						
k	*Baseline*	5000	10000	15000	20000	25000	30000	*Average* ΔU
SAR	*DQN*	-7.3	25.4	41.8	7.2	9.0	3.4	**13.3**
	CQL	37.4	-2.5	-4.1	20.1	17.9	-9.0	**10.0**
ISR	*DQN*	-1.3	-48.5	-29.9	-78.7	-107.8	-29.1	-49.2
	CQL	6.4	1.6	-2.2	-22.3	-10.3	-25.9	-8.8
AI	*DQN*	26.9	-48.5	-43.7	-74.6	-41.3	-84.6	-44.3
	CQL	-13.1	-7.6	-29.8	-6.5	-22.3	-15.3	-15.8
SFI	*DQN*	-33.4	17.9	21.4	45.4	-6.9	-0.1	7.4
	CQL	-5.5	-2.1	7.4	-3.9	-3.6	-18.5	-4.4
TI	*DQN*	36.9	-28.1	34.5	15.7	6.1	-9.1	-0.2
	CQL	7.6	-1.3	-2.1	11.8	-16.5	5.2	**0.8**

Table 4.6: ΔU for every alleged symmetry in Stochastic Acrobot with two baselines and different batch sizes N.

		N				
k	*Baseline*	10000	20000	30000	40000	*Average* ΔU
AAVI	*DQN*	24.7	-17.5	-63.4	-10.6	-16.7
	CQL	-2.8	10.5	-9.5	213.3	**52.9**
CAVI	*DQN*	8.9	-9.3	-24.6	-48.0	-12.2
	CQL	-8.8	0.5	4.4	1.1	-0.7
AI	*DQN*	-377.3	-399.3	-386.8	-388.5	-388.0
	CQL	-25.6	235.3	-88.2	-49.9	17.9
SSI	*DQN*	265.7	-408.2	-334.9	-396.3	-218.4
	CQL	35.8	4.0	11.9	-22.8	7.2

4.5 Conclusions

Data efficiency in the offline learning of MDPs is highly coveted. Exploiting the intuition of an expert about the nature of the model can help to learn dynamics that better represent reality.

In this chapter, we built a semi-automated tool that can aid an expert in providing a statistical data-driven validation of her/his intuition about some properties of the environment. Correct deployment of the tool could improve the performance of the optimal policy obtained by solving the learned MDP. Indeed, our results suggest that the proposed algorithm can effectively detect a symmetry of the dynamics of an MDP with high accuracy and that exploiting this knowledge can not only reduce the distributional shift but also provide performance gain in an envisaged optimal control of the system. However, when applied to Offline RL environments with DNN, all the prescriptions (and issues) about hyperparameter fine-tuning well known to Offline RL practitioners persist.

Besides its pros, the proposed paradigm is still constrained by several limitations. We note

that the quality of the approach in continuous MDPs is greatly affected by the architecture of the Normalizing Flow used for Density Estimation and, more generally, by the state-action space preprocessing. In detail, sometimes an environment is endowed by symmetries that an expert can not straightforwardly perceive in the default representation of the state-action space and a transformation would be required (imagine the very same CartPole, but with also the linear speed and position of the car expressed in polar coordinates).

Future perspectives could include: (i) expanding this approach by trying out more recent Normalizing Flow architectures like FFJORD (Grathwohl et al., 2019); (ii) considering combinations of multiple symmetries.

Key Takeaways

- Offline estimation of the dynamics of a MDP can be challenging but is made easier if the model's dynamics is invariant with respect to certain state and action transformations called symmetries
- A pipeline using density estimation methods, such as Normalizing Flows, is presented to effectively detect expert-proposed symmetries and exploit them to augment the original data set
- This data augmentation technique can be used to improve the performance of offline reinforcement learning architectures and can tackle both deterministic and non-deterministic MDPs.

The content of this chapter gave rise to the following publications:

Giorgio Angelotti, Nicolas Drougard and Caroline Ponzoni Carvalho Chanel (2022). 'Expert-guided Symmetry Detection in Markov Decision Processes'. In: *Proceedings of the 14th International Conference on Agents and Artificial Intelligence - Volume 2: ICAART,*. INSTICC. SciTePress, pp. 88–98

☞See the proceedings ☞See the arXiv preprint

Giorgio Angelotti, Nicolas Drougard and Caroline Ponzoni Carvalho Chanel (2023b). 'Data Augmentation Through Expert-Guided Symmetry Detection to Improve Performance in Offline Reinforcement Learning'. In: *Proceedings of the 15th International Conference on Agents and Artificial Intelligence - Volume 2: ICAART,*. INSTICC. SciTePress, pp. 115–124

☞See the proceedings ☞See the arXiv preprint

In the next chapter, we will contribute to addressing another well-known issue among offline reinforcement learning practitioners: offline risk-sensitive policy selection.

Chapter 5

Bayesian Policy Selection

In the field of offline model learning for planning and Reinforcement Learning, limited data availability can hinder the accuracy of the value function estimate for the corresponding MDP. This can lead to suboptimal or even risky policies when deployed in the real world, where the consequences of incorrect decisions can be catastrophic. As a result, researchers have explored various approaches to reduce model error and develop risk-aware solutions that take into account model uncertainty. In this chapter, we introduce Exploitation vs Caution (EvC), a paradigm that elegantly incorporates model uncertainty using the Bayesian formalism, and selects the policy that maximizes a risk-aware objective over the Bayesian posterior among a fixed set of candidate policies. We validate EvC against state-of-the-art approaches in a variety of simple environments, demonstrating its ability to select robust policies that outperform the current baselines. EvC is thus a valuable tool for practitioners aiming to apply offline planning and Reinforcement Learning in real-world scenarios.

Research question The focus of this chapter is policy selection (or also hyperparameter selection) within a set of strategies obtained offline from a fixed data set of demonstration. The relative state-of-the-art has been discussed in Chapter 2, and more specifically, the part about evaluation and selection of risk-sensitive policies for offline MDPs has been outlined in Section 2.5.

Recall that recently, Chandak et al. (2021) proposed UnO, a method for evaluating policies offline using risk-sensitive metrics. However, this approach is not expected to work effectively for deterministic policies, as it relies on Importance Sampling.

The research question that we answer is the following:
Is it possible to develop to offline evaluate and select deterministic policies for MDPs in a risk-sensitive way?

We show that developing such a technique is feasible, at least for *small finite states and actions* MDPs. Before presenting the results, let use provide some necessary definitions.

5.1 Recalling definitions

Let us call the definition of performance in an MDP.

Definition 20. *The performance of a policy π in an MDP M with value function V_M^π is defined as:*

$$u^\pi(M) = \mathbb{E}_{s\sim\mu_0}\left[V_M^\pi(s)\right]. \tag{5.1}$$

Then, let us better formalize the Bayesian MDP (BMDP) briefly introduced in Section 2.2.

Definition 21 (BMDP). *A BMDP is an 8-tuple* $\beta \stackrel{\text{def}}{=} \langle \mathcal{S}, \mathcal{A}, \mathcal{T}, \mathcal{R}, \hat{\tau}, \rho, \gamma, \mathcal{B} \rangle$ *where* $\mathcal{S}$ *is the set of states;* $\mathcal{A}$ *the set of actions;* $\mathcal{T}$ *is a parametric family of transition functions* T *of any MDP compatible with* $\mathcal{S}$ *and* $\mathcal{A}$*:* $\mathcal{T} = \left\{T : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to [0,1] \text{ s.t. } \sum_{s' \in \mathcal{S}} T(s,a,s') = 1\right\}$*;* $\mathcal{R}$ *is a parametric family of reward functions* R *of any MDP compatible with* $\mathcal{S}$ *and* $\mathcal{A}$*:* $\mathcal{R} = \left\{R : \mathcal{S} \times \mathcal{A} \to [r_{min}, r_{max}]\right\}$*;* $\hat{\tau}$ *is a non-informative prior distribution uniform over* $\mathcal{T}$*:* $\int_{T \in \mathcal{T}} d\hat{\tau} = 1$ *with* $\hat{\tau} \geq 0$*;* ρ *is a non-informative prior distribution uniform over* $\mathcal{R}$*:* $\int_{r \in \mathcal{R}} d\rho = 1$ *with* $\rho \geq 0$*;* $\gamma \in [0,1)$ *is the discount factor, and* $\mathcal{B} = \{(s_t, a_t, r_t, s_{t+1})\}$ *is a batch of transitions generated by acting in a fixed, unknown MDP compatible with* $\mathcal{S}$ *and* $\mathcal{A}$*and initial state distribution* μ_0*.*

Definition 22. *A solution to a BMDP* β *is a policy* π *which maximizes the following utility function:*

$$\mathcal{U}_\beta^\pi \stackrel{\text{def}}{=} \mathbb{E}_{M \sim \hat{\tau}_p}\left[u^\pi(M)\right] \tag{5.2}$$

where, $U_M^\pi \stackrel{\text{def}}{=} \mathbb{E}_{s \sim \mu_0}\left[V_M^\pi(s)\right]$ *is the expected value of an BMDP, averaged on the initial state, with transition function sampled from a Bayesian prior or posterior* $\hat{\tau}_p$*, e.g. defined by Eq. (2.5).*

The optimal performance with respect to Eq. (5.2) will be the one that, on average, works the best on the BMDP β when the model is distributed according to the Bayesian posterior:

$$\mathcal{U}_\beta^* = \max_\pi \mathcal{U}_\beta^\pi. \tag{5.3}$$

Definition 23. *Let* β *be a BMDP and let* $V_M^\pi(s)$ *be the value function at state* s *while following a policy* π *in the MDP* M *with transitions distributed according to the posterior* $\hat{\tau}_p$*. Let also* $Pr\left(u^\pi(M)|\mathcal{B}\right)$ *be the pdf over the possible values assumed by* $u^\pi(M) = \mathbb{E}_{s \sim \mu_0}\left[V_M^\pi(s)\right]$ *given the batch* $\mathcal{B}$*. Then a risk-aware utility function is defined as:*

$$\mathcal{U}_{\beta,\sigma}^\pi \stackrel{\text{def}}{=} \sigma_{M \sim \hat{\tau}_p}\left[u^\pi(M)\right] \tag{5.4}$$

where, σ *is a risk measure, e.g. the VaR (Definition 16) or the CVaR (Definition 17).*

We refer to a BMDP whose performance is computed with respect to a risk-aware utility function as a Risk-aware BMDP.

5.2 Solving offline a Risk-aware BMDP

The expectation over the distribution of models makes the resolution of a BMDP an intractable computational task. Moreover, a Risk-aware BMDP also presents an additional difficulty: the risk measure requires an estimate of the quantiles of the unknown value distribution given a policy. Analytical maximization of the performance defined in Eq. (5.4)) is often either impossible or too computationally demanding. To tackle the maximization problem, a valuable choice is to resort to a Monte Carlo estimate of the performance. We will then look for a sub-optimal policy, rather than an optimal one, by constraining the search to a set of candidate policies Π. However, what number $L_\pi \in \mathbb{N}$ of models would be necessary to be sampled in order to have an accurate estimate of the performance of a policy within a chosen confidence interval? Ideally, L_π should be as small as possible because Policy Evaluation has to be performed L_π times to obtain the Bayesian posterior distribution of values assumed by the Value Functions.

Fortunately, this problem has been addressed by the work of Briggs and Ying (2018). It proposes a procedure that allows iteratively sample values from a distribution whose quantile is required until the estimate of the said quantile will fall within a confidence interval with a required

probabilistic significance.

In the present chapter, we exploit the idea of estimating a quantile through sampling to propose the Monte Carlo Confident Policy Selection (MC2PS) algorithm. MC2PS is presented in Algorithm 7. MC2PS identifies a robust policy for a Risk-aware Bayesian MDP among a set of candidate policies.

Algorithm 7: Monte Carlo Confident Policy Selection (MC2PS)

Input: set of policies Π, significance level $\alpha \in [0,1]$, sampling batch size $k \in \mathbb{N}$, relative error tolerance $\varepsilon_{rel} \in [0,1]$, posterior distribution $\hat{\tau}_p$, risk level $q \in (0,1)$, risk measure σ, initial state distribution μ_0, evaluation discount factor γ

Output: Best policy π^*

1 **for** $\pi \in \Pi$ **do**
2 $\quad \mathcal{U}^{\pi}_{\beta,\sigma} \leftarrow$ RISKEVALUATION $(\pi, \sigma, \hat{\tau}_p, \mu_0, \varepsilon_{rel}, \alpha, q, k, \gamma)$
3 **end**
4 **return** $\pi^* = arg\max_{\pi \in \Pi} \mathcal{U}^{\pi}_{\beta,\sigma}$
5 **procedure** RiskEvaluation
Input: policy π, risk measure $\sigma = \{VaR, CVaR\}$, posterior distribution $\hat{\tau}_p$, initial state distribution μ_0, relative error threshold $\varepsilon_{rel} \in [0,1]$, significance level $\alpha \in [0,1]$, risk level $q \in (0,1)$, sampling batch size $k \in \mathbb{N}$, evaluation discount factor γ.
6 Initialize $u^{\pi} = \varnothing$
7 *(the loop estimates the quantile needed in Eq. (5.4))*
8 **repeat**
9 **for** $j \in \{1, \ldots, k\}$ ***in parallel*** **do**
10 $\quad$ Sample $\mathcal{M}_j \sim \hat{\tau}_p$
11 $\quad V^{\pi}_{\mathcal{M}_j}(s) \leftarrow$ Policy Evaluation on model $\mathcal{M}_j$
12 $\quad u^{\pi}(\mathcal{M}_j) \leftarrow \mathbb{E}_{s\sim\mu_0}[V^{\pi}_{\mathcal{M}_j}(s)]$ Eq. (5.1)
13 $\quad u^{\pi} \leftarrow$ **append** $u^{\pi}(\mathcal{M}_j)$
14 **end**
15 $L_{\pi} \leftarrow |u^{\pi}|$
16 Sort u^{π} in increasing order
17 Find $(g,h) \in \mathbb{N}^2$ such that $|h-g|$ is minimal and:
18 $\quad Pr(u^{\pi}_g \leq a_q < u^{\pi}_h) = \left(\sum_{i=g}^{h-1} \binom{L_{\pi}}{i} q^i (1-q)^{L_{\pi}-i}\right) > 1-\alpha$
19 **until** $u^{\pi}_h - u^{\pi}_g < \varepsilon_{rel} \cdot \left(u^{\pi}_{L_{\pi}} - u^{\pi}_1\right)$
20 **if** $\sigma =$ *VaR* **then**
21 $\quad \widehat{a_q} \leftarrow u^{\pi}_g$
22 $\quad$ **return** $\widehat{a_q}$
23 **end**
24 **if** $\sigma =$ *CVaR* **then**
25 $\quad \widehat{\phi_q} \leftarrow \frac{1}{g}\sum_{i=1}^{g} u^{\pi}_i$
26 $\quad$ **return** $\widehat{\phi_q}$
27 **end**
28 **end procedure**

In detail, for a given set of policies Π and for every policy $\pi \in \Pi$, the algorithm incrementally samples k transition models from $\hat{\tau}_p$ and performs Policy Evaluation in parallel for each one of them until the stopping criterion is reached (see the RISKEVALUATION procedure in Alg. 7). The stopping criterion guarantees that the estimate of the q-quantile is statistically well approximated with a significance level α within a dynamically sampled confidence interval whose width is smaller than ε_{rel} (lines 17-19) given the total L_{π} models sampled.

Indeed, being $u^{\pi}_i := u^{\pi}(M_i)$ the list of ordered performance values obtained from the sampled L_{π} models with $i \in \{1, \ldots, L_{\pi}\}$, the probability that the elements with indices h and g of this list are bounding a_q, will be given by the probability of the union of all the (incompatible) events that lead

to $u_g^\pi \leq a_q \leq u_h^\pi$ (see Figure 5.1). In detail, let a_q be the theoretical Value at Risk of $u^\pi(M)$ at risk level q. Let us denote sampled utility values in increasing order by $u^\pi(M_1) \leq \ldots \leq u^\pi(M_{L_\pi})$, and suppose that the utility distribution has no probability atom at a_q: $\forall 1 \leq i \leq L_\pi$, $\Psi(a_q) := Pr\big(u^\pi(M_i) \leq a_q\big) = q$. Let us introduce the random variables

$$B_i = \mathbb{1}_{\left\{u^\pi(M_i) \leq a_q\right\}} = \begin{cases} 1, & \text{if } u^\pi(M_i) \leq a_q, \\ 0, & \text{otherwise.} \end{cases} \tag{5.5}$$

The random variables B_i are drawn by a Bernoulli distribution with parameter q. The random variable $B = \sum_{i=1}^{L_\pi} B_i$ is the number of sampled utilities that are lower than a_q, drawn by a binomial distribution with parameters L_π and q. The event $\left\{u^\pi(M_g) \leq a_q < u^\pi(M_h)\right\}$ is $\cup_{i=g}^{h-1}\{B = i\}$, *i.e.* the event "there are exactly g, $g+1$, ..., or $h-1$ sampled utility values that are lower than a_q". Using the binomial distribution formula, the probability of this event is

$$Pr\Big(u^\pi(M_g) \leq a_q < u^\pi(M_h)\Big) = \sum_{i=g}^{h-1} Pr(B = i) = \sum_{i=g}^{h-1} \binom{L_\pi}{i} q^i (1-q)^{L_\pi - i}. \tag{5.6}$$

Hence, by imposing constraint $\sum_{i=g}^{h-1} \binom{L_\pi}{i} q^i (1-q)^{L_\pi - i} > 1 - \alpha$ when selecting indices r and s, we ensure that

$$Pr\Big(u^\pi(M_g) \leq a_q < u^\pi(M_h)\Big) > 1 - \alpha, \tag{5.7}$$

i.e. we get probabilistic bounds computed from the sampled utility values. Note that, if there is a probability atom at a_q, *i.e.* $Pr\big(u^\pi(M_i) = a_q\big) > 0$, the previous reasoning cannot be applied directly in the case where $q < \Psi(a_q) := Pr\big(u^\pi(M_i) \leq a_q\big)$. However, we can write

$$q_- := Pr\big(u^\pi(M_i) < a_q\big) \leq q < q_+ := Pr\big(u^\pi(M_i) \leq a_q\big), \tag{5.8}$$

and one can show that selected indices g and h are non decreasing with q. Thus, using the risk level q, selected indices are higher than those that would be selected using q_-, and lower than those that would be selected using q_+, both corresponding to utility values bounding the location of the probability atom, *i.e.* the Value at Risk a_q, with probability $1 - \alpha$. Eventually, the algorithm leverages the estimate of both the Value at Risk and of the policy value achieved on sampled models to obtain an estimate of the utility function $\mathcal{U}_{\beta,\sigma}^\pi$ for a specific risk measure σ and risk level q. For instance, return the estimate of a_q if $\sigma = VaR$ or ϕ_q if $\sigma = CVaR$ (lines 23-28). Finally, once the utility function has been estimated for every policy, it outputs the one that maximizes it (line 4).

Let Λ be the total number of models sampled to estimate the quantile of the performance distribution among policies: $\Lambda = \sum_{\pi \in \Pi} L_\pi$.

MC2PS performs Policy Evaluation Λ times. The size of the space of all applicable policies of a finite state and action space MDP is $|\Pi| = |\mathcal{A}|^{|\mathcal{S}|}$. Looking over the whole policy space can be practically intractable even for not-so-big MDPs. Nevertheless, restricting the research to a subset of policies could be a viable solution also for big MDPs, also considering that the Policy Evaluations are carried out in parallel.

5.2.1 Exploitation vs Caution (EvC)

The work in N. Jiang, Kulesza et al. (2015) showed that the policy obtained by solving an MDP $\hat{M} = (\mathcal{S}, \mathcal{A}, \hat{T}, R, \gamma^*)$, trivially learned from a batch of experiences gathered from another MDP $M = (\mathcal{S}, \mathcal{A}, T, R, \gamma_{ev})$, with γ^* a discount factor such as $\gamma^* \leq \gamma_{ev}$, is more effective in M than

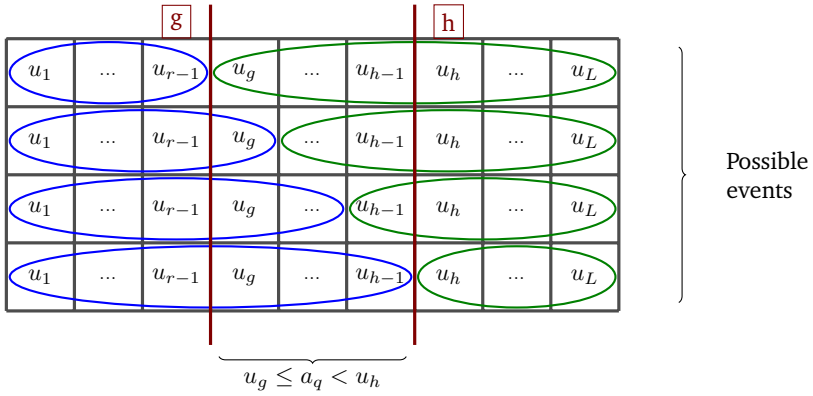


Figure 5.1: Example of how the estimate in Algorithm 7 works: imagine you have an ordered list with L values u_i, $i \in \{1, \ldots, L\}$ represented in the figure as rows. The probability of the event in Eq. (5.7) is $\binom{L}{g} q^g (1-q)^{L-g}$ and corresponds to the probability of the random variable B defined in Eq. (5.5) to assume all integer values between g and $h-1$. The said probability is the sum of the probability of the events $B = i$ with $g \leq i < h$. In the figure, every addend is represented as a row. In blue are encircled the values of u_i smaller than a_q and in green the ones bigger. The algorithm looks for the indices $(g, h) = \textit{argmin}_{(g,h)}|g - h|$ such that $Pr(u_g \leq a_q < u_h) > 1 - \alpha$ and $u_h - u_g < \varepsilon$ where ε is an error term dictating the maximum acceptable size of the confidence interval.

the policy obtained by solving $\hat{M}$ using γ_{ev}. The rationale behind this observation is that $\hat{T}$ is an approximation of T, and may not be trusted for long-term planning horizons. Choosing the optimal γ^* balances the *exploitation* of information contained in the batch and the need for *caution*, given that the model estimate is imperfect.

Motivated by the findings in N. Jiang, Kulesza et al. (ibid.) and the intuition that the model M that generated the data will differ from $\hat{M}$ but is expected to be *"close"* to it, we hypothesize that policies obtained by solving an MDP $\tilde{M} = (\mathcal{S}, \mathcal{A}, \tilde{T}, R, \gamma)$, where $\tilde{T}$ is *close* to $\hat{T}$ and $\gamma \leq \gamma_{ev}$, can serve as viable solutions for the Risk-aware Bayesian MDP.

In the following, we present the Exploitation vs Caution (EvC) algorithm, depicted in Algorithm 8. EvC aims to identify a promising risk-aware policy by concentrating the search on a set of candidate policies Π, computed using several baseline algorithms $\mathbb{A}$. This set is further enriched by solving different MDPs $\tilde{M}$ with $\gamma \leq \gamma_{ev}$ values. It is important to note that the goal is to find a policy that performs well in the model M, considering that the agent does not have access to the probability values defining the model, but only to the pre-collected batch. Model uncertainty is addressed within a Bayesian MDP framework. As previously mentioned, our objective is not to find the optimal solution to the Bayesian MDP, but rather to select the most robust or risk-aware policy from the candidate set.

In detail, EvC first generates candidate policies that constitute the set Π by calling the GENERATEPOLICIES procedure (line 1). For this purpose, the problem is solved using a portfolio of state-of-the-art algorithms, starting from the batch (line 6). Additionally, the trivial MDP[4] $\hat{M}$ and l supplementary MDPs are sampled from the Bayesian posterior $\hat{\tau}_p$ obtained from the batch (line 4). These MDPs are then solved using different values of $\{\gamma \in G | \gamma \leq \gamma_{ev}\}$ (lines 7-9), where γ_{ev} is the discount factor of the Risk-aware Bayesian MDP. It is important to note that the resulting set Π contains unique solutions (line 9 and line 12). Finally, MC2PS is executed with the obtained set of candidate policies Π, returning the best risk-aware solution $\pi^* \in \Pi$ (line 2).

[4]The trivial MDP $\hat{M}$ is a straightforward MDP estimate using the batch $\mathcal{B}$. For instance, in the case of a discrete MDP, this is equivalent to the model that maximizes the likelihood of $\mathcal{B}$, i.e., the one whose transition probabilities are obtained from the frequencies of transitions in the batch.

Algorithm 8: EvC

Input: risk level $q \in [0,1]$, significance level $\alpha \in [0,1]$, sampling batch size $k \in \mathbb{N}$, relative error tolerance $\varepsilon_{rel} \in [0,1]$, posterior distribution $\hat{\tau}_p$, risk measure σ, initial state distribution μ_0, set of discount factors G, number of models to solve $l \in \mathbb{N}$, $\mathcal{B}$ batch of transitions, evaluation discount factor γ_{ev}

Output: best policy π^*.

$\Pi \leftarrow$ GENERATEPOLICIES $(\hat{\tau}_p, G, l)$

return $\pi^* =$ *MC2PS* $(q, \alpha, k, \varepsilon_{rel}, \hat{\tau}_p, \Pi, \sigma, \mu_0, \gamma_{ev})$

procedure GeneratePolicies

Input: posterior distribution $\hat{\tau}_p$, set of discount factors G, number of models $l \in \mathbb{N}$ to be solved, $\mathcal{B}$ batch of transitions.

Output: policy set Π.

Initialize $\mathbb{M} = \{l$ transition models $\sim \hat{\tau}_p\} \cup \{\hat{T}\}$

Initialize $\Pi = \varnothing$ (an empty set)

Initialize $\mathbb{A} = \{$*SPIBB*, *BOPAH*, *BCR*, *NORBU*$\}$ (examples of baseline algorithms)

for $(\gamma \in G, T \in \mathbb{M})$ **do**

  $\pi_{(T,\gamma)} =$ solution to the MDP with T and γ

  Append $\pi_{(T,\gamma)}$ to Π if $\pi_{(T,\gamma)} \notin \Pi$

for *algorithm* $\in \mathbb{A}$ **do**

  $\pi_{algorithm} =$ solution to the Offline MDP with $\mathcal{B}$ and *algorithm*

  Append $\pi_{algorithm}$ to Π if $\pi_{algorithm} \notin \Pi$

return Π

end procedure

For instance, if we test with 9 different discount factors, such as $G = \{0.1, 0.2, \ldots, \gamma_{ev} = 0.9\}$, and 5 different MDPs $\tilde{M}$ (including $\hat{M}$), with $l = 5$, then we solve $|\Pi| \leq 9l = 45$ MDPs to enrich the set of candidate policies using this approach.

5.2.2 Theoretical guarantees

Since EvC searches for the policy $\pi \in \Pi$ that maximizes the criterion of Eq. (5.4), the Algorithm 8, rather than yielding a sub-optimal solution to the Risk-aware BMDP, can be seen as a policy selection approach. Assuming that the Bayesian posterior $\hat{\tau}_p$ efficiently encodes the model uncertainty, EvC outputs a policy whose performance in the real environment is guaranteed in probability to be greater than some value that changes with respect to the chosen risk-aware measure. More simply, we can provide theoretical guarantees on the estimate of the quantile needed to compute the risk-aware utility function that will be eventually maximized over the set of candidate policies.

Theorem 1. *Let $\pi \in \Pi$ be a candidate policy and $u^\pi(M_g)$ be an estimate of the Value at Risk of $u^\pi(M)$ at risk level q calculated through EvC. Let $u^\pi(M)$ be the performance of π with M distributed according to the Bayesian posterior $\hat{\tau}_p$. The performance of π in this MDP M is greater than the estimate of a_q with probability:*

$$Pr\big(u^\pi(M) \geq u^\pi(M_g)\big) \geq (1-q)(1-\alpha). \tag{5.9}$$

Proof. Note that $\big\{u^\pi(M) \geq a_q\big\} \cap \big\{a_q \geq u^\pi(M_g)\big\} \subseteq \big\{u^\pi(M) \geq u^\pi(M_g)\big\}$, where a_q denotes the Value at Risk of $u^\pi(M)$ at risk level q. The two events of the intersection respectively depend on two independent random variables – a future performance $u^\pi(M)$, that could be obtained by acting according to the policy π, and a Value at Risk estimate $u^\pi(M_g)$, whose randomness is the result of the sampling procedure in the Algorithm 7. The previous inclusion allows writing $Pr\big(u^\pi(M) \geq$

$u^\pi(M_g)\big) \geq Pr\big(u^\pi(M) \geq a_q\big) \cdot Pr\big(a_q \geq u^\pi(M_r)\big) \geq (1-q) \cdot Pr\big(a_q \geq u^\pi(M_g)\big) \geq (1-q)(1-\alpha)$. The last inequality is ensured by the quantile estimation (lines 19-22 in Algorithm 7), and the previous one by the definition of a_q. Therefore, we get the Equation 5.9. □

When the risk-aware measure used in EvC is *VaR* the lower bound on $u^\pi(M)$ ($u^\pi(M_g)$) in the Proof of Theorem 1 is maximized over the policies. If the risk-aware measure is *CVaR* the empirical expected value over the q-fraction of low-performant policies is maximized. Since $u^\pi(M_1) \leq u^\pi(M_2) \leq \ldots \leq u^\pi(M_g)$, then $\frac{1}{g}\sum_{i=1}^{g} u^\pi(M_i) \leq u^\pi(M_g)$, therefore the same lower bound in probability is also valid for the *CVaR* utility function:

$$Pr\left(u^\pi(M) \geq \frac{1}{g}\sum_{i=1}^{g} u^\pi(M_i)\right) \geq Pr\big(u^\pi(M) \geq u^\pi(M_g)\big) \geq (1-q)(1-\alpha). \tag{5.10}$$

Theorem 2. *Let $\pi \in \Pi$ be a candidate policy, $N_i \in \mathbb{M}$ be one of the n new sampled models from the posterior $\hat{\tau}_p$ such that $\forall i, u^\pi(N_i) \leq u^\pi(M_g)$, with $u^\pi(M_g)$ calculated through EvC, and $\overline{U} = \frac{1}{n}\sum_{i=1}^{n} u^\pi(N_i)$ be the new estimate of ϕ_q. This new estimate of the Conditional Value at Risk of u^π at risk level q respects the following inequality:*

$$Pr\left(|\overline{U} - \phi_q| \geq t\right) \leq 2\exp\left(-\frac{2nt^2}{\big(u^\pi(M_g) - \xi\big)^2}\right) + \alpha, \tag{5.11}$$

with $\xi = \inf_{m\in\mathbb{M}} u^\pi(m)$, or any other lower bound of u^π as, for instance, $\frac{r_{min}}{1-\gamma}$. Note that $\xi \geq 0$ if the reward values are known to be non-negative.

Proof. By using the law of total probability, and upper bounding some probability values by 1,

$$\begin{aligned} Pr\Big(|\overline{U} - \phi_q| \geq t\Big) &= Pr\Big(|\overline{U} - \phi_q| \geq t \big| \forall i, u^\pi(N_i) \leq a_q\Big) Pr(\forall i, u^\pi(N_i) \leq a_q) \\ &\quad + Pr\Big(|\overline{U} - \phi_q| \geq t \big| \exists i \text{ s.t. } u^\pi(N_i) > a_q\Big) Pr\big(\exists i \text{ s.t. } u^\pi(N_i) > a_q\big) \\ &\leq Pr\Big(|\overline{U} - \phi_q| \geq t \big| \forall i, u^\pi(N_i) \leq a_q\Big) + Pr\big(\exists i \text{ s.t. } u^\pi(N_i) > a_q\big). \end{aligned}$$

The probability value on the right is lower than $Pr\big(u^\pi(M_g) > a_q\big) \leq Pr\Big(a_q \notin \big[u^\pi(M_g), u^\pi(M_h)\big)\Big) \leq \alpha$ using the inequality $Pr(u^\pi(M_g) \leq a_q) > 1-\alpha$ from lines 19-22 of Algorithm 7. What follows only depends on the definition of ϕ_q as the expected value up to a_q, and Hoeffding's inequality. □

It should be noted that in Theorem 2, the models $(N_i)_{i=1}^n$ are independent random variables when conditioned on $\forall i, u^\pi(N_i) \leq a_q$. This is due to the fact that a_q represents the theoretical Value at Risk at risk level q, and as such, it is not a random variable.

5.3 Experiments

To evaluate the proposed approach, we selected three small MDPs, which are easy-to-study stochastic environments with diverse characteristics: two planning environments without absorbing states, Ring (5 states, 3 actions) and Chain (5 states, 2 actions). The former involves stabilizing the agent in a specific non-absorbing goal with stochastic drift, while the latter presents cycles. Additionally, we considered the Random Frozen Lake (RFL) environment, a re-adaptation of the Frozen Lake from the OpenAI Gym suite (Brockman et al., 2016), which is an 8×8 grid world with fatal absorbing states.

5.3.1 Description of the environments

Ring This environment consists of five states: $0, \ldots, 4$, forming a single loop. There are three possible actions: *a*, *b*, and *c*. The agent starts in state 0. Action *a* moves the agent to state $s-1$ with probability 1.0 (e.g., from state 4 to 3) if $s = 0, 1, 3$, and with probability 0.5 if it is elsewhere. With action *b*, the agent remains in the same state with probability 0.8 and moves to the left or right with probability 0.1 if in state $s = 0, 1, 3$. If in state 2 or 4, the agent moves with probability 1. Action *c* moves the agent to the right with probability 0.9 and remains stationary with probability 0.1 if in state $s = 0, 1, 3$. In other states, the same effects occur but with probability 0.5. The agent earns an immediate reward $r = 0.5$ when moving from $2 \to 3$ or $4 \to 3$, and $r = 1$ for any transition $3 \to 3$. In all other cases, $r = 0$. A graphical representation is shown in Figure 5.2a.

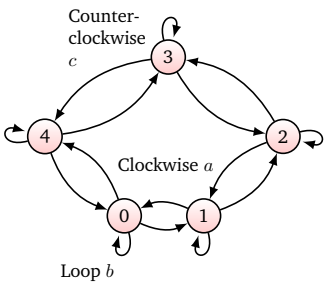
Chain This environment, proposed by Strens (2000), was adapted for this study. It consists of five states with an open chain topology and two actions a and b. The agent starts in the leftmost state. With action a, the agent moves to the right and receives an immediate reward $r = 0$ with probability 0.8. Once in the rightmost state, performing action a allows the agent to stay and receive a reward $r = 10$ with probability 0.8. The agent slips back to the origin and earns a reward $r = 2$ with probability 0.2. Action b moves the agent to the origin state with probability 0.8, receiving a reward $r = 2$, or lets it go right with probability 0.2, earning $r = 0$. The optimal policy involves applying action b in the first state and action a in the others. A representation is shown in Figure 5.2b.

Random Frozen Lake (RFL) The Frozen Lake Environment of the Open AI Gym suite (Brockman et al., 2016) was edited for this study. The agent moves in a grid world (8×8). It starts in the utmost left corner and it must reach a distant absorbing goal state that yields a reward $r = 1$. In the grid, there are some holes. If it falls into a hole it is blocked there and it can not move anymore, obtaining from that moment an immediate reward $r = 0$. Unfortunately, the field is covered with ice and hence it is slippery. When the agent wants to move towards a nearby state it can slip with fixed probability p and ends up in an unintended place. The grid is generated randomly assuring that there always exists a hole-free path connecting the start and the goal. Moreover, to each couple of actions and non-terminal state (a, s) is assigned a different immediate reward r sampled at random between $(0, 0.8)$ at the moment of the generation of the MDP problem. The MDP itself does not have a stochastic reward, but the map and the rewards are randomly generated. A graphical representation (for a 3×3 grid) is shown in Figure 5.2c.

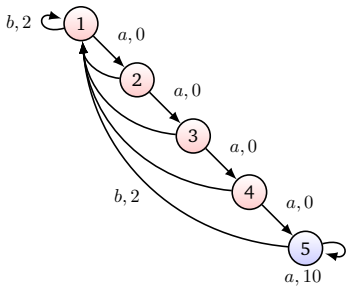
5.3.2 Setup

Given $(n, m) \in \mathbf{N}^2$, m trajectories, with n steps each, are generated following a uniform random policy in each environment. We opted for a random data collecting procedure because we imagine using EvC in a scenario where both the developers and the autonomous agent are completely agnostic about the model dynamics and have no prior knowledge.

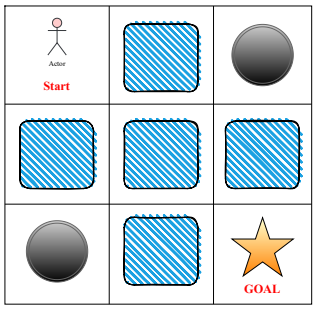
The true environment is assumed to be known for the a posteriori evaluation. The most likely transition model is inferred from the batch. The trivial MDP was then solved with the Policy Iteration algorithm and its relative performance in the true environment is obtained by Policy Evaluation. EvC data was computed with $a_{0.25}$ and $\phi_{0.25}$ (the first quartile). For each of these risk-aware measures, the following parameters (see Algorithm 8) were used: the set of discount factors $G = \{0.2, 0.4, 0.6, 0.8, 0.9\}$, the significance level $\alpha = 0.01$, the relative tolerance error $\varepsilon_{rel} = 0.01$, and the number l of different models sampled from the prior is given in Table 5.1.



(a) Representation of the Ring environment.



(b) Representation of the Chain environment. Each circle is a state, each arrow is a transition labeled by action, reward.



(c) Frozen Lake environment example with a grid of size 3×3. The agent has to reach the goal, paying attention to slippery (blue) states and avoiding holes (black).

Figure 5.2: Environments illustration.

In the experiments, for a given batch size $N = nm \in \mathbf{N}$, 50 different batches were generated containing fixed size trajectories. The trajectory sizes used are also given in Table 5.1.

The chosen state-of-the-art algorithms that provide the baselines for the set of candidate policies are the following:

1. *Deterministic policies:* output by the following baselines[5], please notice that the quantile used for the robust and soft robust objectives in the algorithms is the same provided as general input for the estimate of EvC: **BCR** (Petrik and Russel, 2019), **NORBU - Soft Robust *CVaR*** (Lobo, Ghavamzadeh and Petrik, 2021) (soft robust hyperparameter $\lambda = 0.5$);

2. *Stochastic policies:* output by the following algorithms [6]: **SPIBB** (Laroche, Trichelair and Des Combes, 2019) receiving as input the batch collector policy, and, **BOPAH** (B. Lee et al., 2020) receiving as input the batch collector policy.

In our implementation of these baselines we only used intuitively tunable parameters (*e.g.* the discount factor).

We did not use MOPO (T. Yu et al., 2020) and MOReL (Kidambi et al., 2020) since: (1) they have usually been tested on continuous state MDPs driven by a deterministic dynamics, while here we are tackling non-deterministic environments; (2) they highly rely on hyperparameter domain-dependent fine tuning which we did not do to fulfill the offline learning obligation.

In the evaluation phase, the discount factor is defined as $\gamma_{\text{ev}} = 0.9$. The other simulation parameters are provided in Table 5.1. Eventually, we also compared EvC with UnO by performing the risk-sensitive off-policy evaluation with UnO over the same set of candidate policies provided to EvC and then selected the one that maximized the risk-sensitive objective. Although it is true that UnO, like other Importance Sampling-based off-policy evaluation methods, may not accurately evaluate deterministic policies, we still compare our approach to it because, to our knowledge, there are no other risk-sensitive off-policy evaluation approaches available.

5.3.3 Metrics

We report metrics about the performance differences $\Delta U = u^{\pi}_{\beta,\sigma} - u^{\pi_{trivial}}_{\beta,\sigma}$ of the policies π obtained with a specific algorithm (Eq. (5.1) using the utility function defined in Eq. (5.4)) and the

[5]The code was taken from the authors' GitHub repository: https://github.com/marekpetrik/craam2/tree/master/examples/evaluation/algorithms and readapted.

[6]The code was taken from the GitHub repository: https://github.com/KAIST-AILab/BOPAH and readapted.

Table 5.1: Parameters and hyperparameters used during the simulations: n is the number of steps in each trajectory contained in a batch; l is the number of different models sampled from the prior in EvC (Algorithm 8); $\{N_\wedge\}$ is the set of different thresholds used in SPIBB; **fold** and **DOF** are the fold and degree of freedom hyper-parameters used in BOPAH; λ is the soft robust hyper-parameter of NORBU. Bold values are displayed in the plots.

Environment	n	l	$\{N_\wedge\}$	**fold**	**DOF**	λ
Ring	8	3	$\{\mathbf{1}, 2, 3, 5, 7, 10, 20\}$	2	20	0.5
Chain	8	3	$\{\mathbf{1}, 2, 3, 5, 7, 10, 20\}$	2	20	0.5
RFL (8×8)	15	10	$\{\mathbf{1}, 2, 3, 5, 7, 10, 20\}$	2	20	0.5

performance obtained by solving the trivial model in the same setting and using the same batch of trajectories. This last value is normalized by the performance of the optimal policy. In particular, we consider: (1) the maximal ΔU obtained, (2) the mean value over all the different simulations, (3) the median over all simulations, and (4) the minimal ΔU. The selected metrics provide insight into the validity of the approaches. We consider only the extrema of the distributions of the results (min, max), their median, and mean values since trying to estimate the whole distributions, and hence their quantiles could result in wrong conclusions if we are not sampling enough batches. For instance, to correctly estimate a quantile of order $q = 0.25$ with a $\alpha = 0.01$ significance usually tens of thousands of samples are required. However, we are performing only hundreds of simulations with a fixed batch size N, which are enough for the selected metrics but insufficient for the study of the whole distributions.

Please notice that the distribution whose statistics are displayed in the tables is not the one used to maximize Eq. (5.4) since it is a distribution over different starting batches collected with the same random policy and not the distribution that encodes the model uncertainty using the same starting dataset. Indeed, from a bayesian point of view the results are distributed along:

$$Pr\left(u^\pi(M), \mathcal{B} | \pi_{random}\right) = Pr\left(u^\pi(M) | \hat{\tau}_p\right) Pr\left(\hat{\tau}_p | \mathcal{B}\right) Pr\left(\mathcal{B} | \pi_{random}\right), \tag{5.12}$$

that represents the probability of collecting a batch $\mathcal{B}$ by collecting transitions using a random policy π_{random} and hence observing the performance $u^\pi(M)$ by deploying a policy π. Note that there is a deterministic mapping among the posterior $\hat{\tau}_p$ and the batch, therefore $Pr\left(\hat{\tau}_p | \mathcal{B}\right)$ is a delta function.

5.3.4 Results and discussion

For Ring and Chain, the results averaged over 100 different batches for each batch size $N \in \{8, 16, 24, 32, 40, 48, 56\}$ are displayed in Table 5.2. While for RFL the results averaged over 50 different batches for every batch size $N \in \{15, 30, 45, 60, 75, 90, 105\}$ are reported in Table 5.3.

Even if the datasets are composed of relatively short trajectories (Ring and Chain $n = 8$ time steps each, Random Frozen Lake $n = 15$ time steps each), in most cases UnO did not manage to accurately evaluate the deterministic policies. Please note that UnO computes the Importance Sampling ratio for a trajectory h, a policy π, and a behavioral policy π_β as

$$\rho_h^\pi = \prod_{i=1}^{n_h} \frac{\pi\left(s_i, a_i\right)}{\pi_\beta\left(s_i, a_i\right)} \tag{5.13}$$

where n_h is the number of time steps of the trajectory h. However, this formulation assumes that both π and π_β are stochastic. In our formulation $\pi_\beta(s, a) = |\mathcal{A}|^{-1}\ \forall (s, a) \in \mathcal{S} \times \mathcal{A}$, but π is stochastic only when it is the output of SPIBB or BOPAH. When π is deterministic, the former

equation can be rewritten as

$$\rho_h^\pi = \prod_{i=1}^{n_h} \frac{\delta_{\pi(s_i),a_i}}{\pi_\beta(s_i,a_i)} = |\mathcal{A}|^{n_h} \prod_{i=1}^{n_h} \delta_{\pi(s_i),a_i}. \tag{5.14}$$

This means that $\rho_h^\pi = |\mathcal{A}|^{n_h}$ if and only if the all sequence of actions and states is consistent with the deterministic policy π, otherwise $\rho_h^\pi = 0$. The probability that the ratio will be zero grows as a power of $|\mathcal{A}|$ and exponentially in n_h. In particular, the probability that a sequence will be generated by the deterministic policy is in Ring $|\mathcal{A}|^{-n_h} = 3^{-8} \approx 1.5 \times 10^{-4}$, in Chain $2^{-8} \approx 3.9 \times 10^{-3}$ and in RFL $4^{-15} \approx 9.3 \times 10^{-10}$. Therefore, UnO almost always chooses a policy between SPIBB and BOPAH, as the Importance Sampling ratio will be zero for other policies. If both the outputs of SPIBB and BOPAH result in a zero Importance Sampling ratio, then the first policy in the candidate set (the trivial policy) is chosen. This phenomenon occurs most of the time. As a result, UnO alternates between the Trivial Policy, one of SPIBB or BOPAH, and occasionally selects another approach.

Ring Using $q = 0.25$ the best method according to the *Max*, *Mean* and the *Median* is NORBU with the *CVaR* Soft Robust objective (see Table 5.2). However, the most robust baseline in terms of worst-case performance is BOPAH. The distributions of results are asymmetric around ΔU. In the cases of BCR and NORBU, the *Mean* and the *Median* are approximately zero. In the cases of SPIBB and BOPAH, the *Median* and the *Mean* are less than zero. Regarding the off-policy evaluation and selection methods, EvC with the *VaR* is the most performing one with respect to all the considered metrics.

Chain In this environment, every baseline except for SPIBB worked the same with SPIBB being the worst in terms of *Min* (see Table 5.2). Regarding the off-policy evaluation and selection, all algorithms performed well since there is not a substantial difference between the approaches (except for SPIBB).

Random Frozen Lake (RFL) We tested the approaches in 4 different RFLs. The best approach in terms of overall metrics in 3 out of 4 environments is again NORBU with the Soft Robust *CVaR* (see Table 5.3). SPIBB is the best in Environment 4. The best selection method is EvC with *VaR/CVaR*, which provided identical performances.

Table 5.2: Statistics of the normalized performance difference ΔU between the reported algorithm (risk level used $q = 0.25$) and the trivial policy averaged over batch size $N \in \{8, 16, 24, 32, 40, 48, 56\}$ with 100 different batches for size in Ring and Chain. On the right ΔU with the algorithm selected by EvC and UnO with $a_{0.25}$ and $\phi_{0.25}$. Notice that both EvC and UnO can pick also a policy obtained with a model solved with a different discount factor.

		Baseline				**Selection Method**			
Environment	*Metrics*	SPIBB	BOPAH	BCR	NORBU	$EvC_{a_{0.25}}$	$EvC_{\phi_{0.25}}$	$UnO_{a_{0.25}}$	$UnO_{\phi_{0.25}}$
Ring	*Max*	0.61	0.48	0.74	**0.84**	**0.82**	0.71	**0.82**	0.72
	Mean	-0.29	-0.28	-0.01	**0.03**	**0.01**	-0.04	-0.26	-0.27
	Median	-0.31	-0.34	**0.0**	**0.0**	**0.0**	**0.0**	-0.27	-0.33
	Min	-0.78	**-0.68**	-0.82	-0.71	**-0.82**	**-0.82**	-0.96	-0.96
Chain	*Max*	**0.55**	0.54	**0.55**	**0.55**	**0.55**	**0.55**	0.54	0.54
	Mean	0.0	0.01	0.01	**0.02**	**0.01**	**0.01**	**0.01**	**0.01**
	Median	**-0.01**	**-0.01**	**-0.01**	**-0.01**	**-0.01**	**-0.01**	**-0.01**	**-0.01**
	Min	-0.38	-0.16	**-0.15**	**-0.15**	**-0.16**	**-0.16**	**-0.16**	**-0.16**

Table 5.3: Statistics of the normalized performance difference ΔU between the reported algorithm (risk level used $q = 0.25$) and the trivial policy averaged over batch size $N \in \{15, 30, 45, 60, 75, 90, 105, 120, 135\}$ with 50 different batches for size in different Random Frozen Lake environments.

		Baseline				**Selection Method**			
Environment	*Metrics*	SPIBB	BOPAH	BCR	NORBU	$EvC_{a_{0.25}}$	$EvC_{\phi_{0.25}}$	$UnO_{a_{0.25}}$	$UnO_{\phi_{0.25}}$
RFL Env. 1	*Max*	**0.32**	0.31	0.31	**0.32**	0.3	0.32	**0.37**	0.32
	Mean	**0.05**	-0.04	-0.04	**0.05**	**0.05**	**0.05**	-0.02	-0.05
	Median	**0.04**	-0.07	-0.04	**0.04**	**0.04**	**0.04**	-0.01	-0.08
	Min	-0.25	**-0.22**	-0.39	-0.33	-0.33	-0.33	-0.31	**-0.22**
RFL Env. 2	*Max*	0.33	0.22	0.3	**0.34**	**0.34**	**0.34**	0.28	0.18
	Mean	0.02	-0.07	-0.05	**0.06**	**0.06**	**0.06**	0.0	-0.07
	Median	0.01	-0.08	-0.06	**0.06**	**0.06**	**0.06**	-0.01	-0.08
	Min	-0.21	-0.22	-0.29	**-0.12**	**-0.12**	**-0.12**	-0.28	-0.26
RFL Env. 3	*Max*	0.3	0.23	**0.43**	0.36	**0.36**	**0.36**	0.35	0.18
	Mean	0.01	-0.08	0.0	**0.04**	**0.04**	**0.04**	-0.03	-0.08
	Median	-0.0	-0.09	0.01	**0.02**	**0.02**	**0.02**	-0.03	-0.09
	Min	**-0.16**	-0.3	-0.36	-0.27	-0.27	-0.27	-0.29	**-0.26**
RFL Env. 4	*Max*	0.32	0.22	**0.36**	0.31	**0.31**	**0.31**	0.27	0.22
	Mean	0.02	-0.06	0.01	**0.05**	**0.05**	**0.05**	-0.05	-0.06
	Median	0.02	-0.06	0.01	**0.05**	**0.05**	**0.05**	-0.05	-0.06
	Min	-0.32	-0.3	-0.4	**-0.29**	**-0.29**	**-0.29**	-0.39	-0.3

In the following, we comment on the results obtained with EvC. Note this algorithm selects the policy that optimizes the (Conditional) Value at Risk over the first quartile ($q = 0.25$) starting from the set of candidate policies discussed in the previous section.

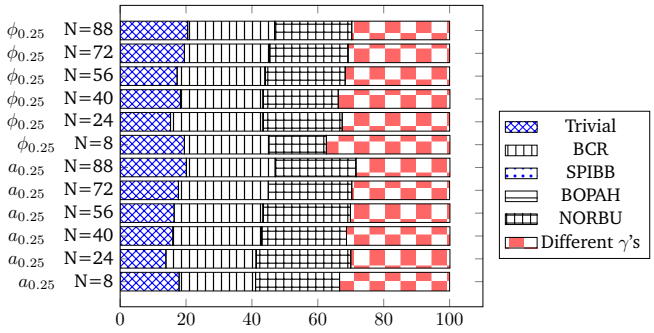


Figure 5.3: Policy selection rate by EvC $a_{0.25}$ and EvC $\phi_{0.25}$ in Ring for different batch sizes.

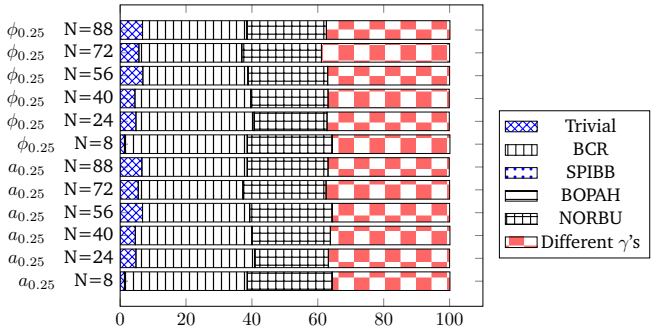


Figure 5.4: Policy selection rate by EvC $a_{0.25}$ and EvC $\phi_{0.25}$ in Chain for different batch sizes.

In terms of risk awareness, after a global study over different batch sizes, EvC does not select the policy that produces the best values concerning the considered metrics. Nevertheless, the policy selected by EvC is among the more robust ones. These results are shown in Figures 5.3,

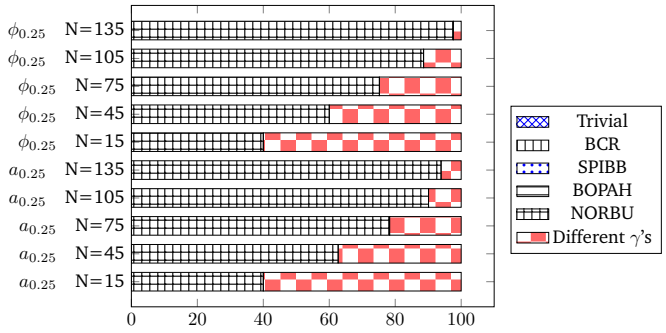


Figure 5.5: Policy selection rate by EvC $a_{0.25}$ and EvC $\phi_{0.25}$ in RFL (aggregate of Env. 1, 2, 3 and 4) for different batch sizes.

5.4, and 5.5. In particular, our approach tends to opt for a policy from the ones obtained by solving several models with different discount factors γ when the batch is small. The number of times such a policy is selected decreases to the benefit of 1) the trivial policy when the batch size N increases (Ring and Chain) or 2) NORBU (in the RFL environment). This is reasonable since model uncertainty decreases with N and the trivial model will be closer and closer to the true one. We suppose that for not-so-small environments (RFL), the trivial policy cannot be trusted for small batch sizes, while NORBU manages to cut the posterior space into ambiguity sets that are efficiently optimized over. The policies computed through SPIBB and BOPAH are never selected. Remember that those are stochastic policies that were obtained by improving the batch collector policy, which was uniformly random over the actions. Stochastic policies seem not to provide good risk-aware estimates with respect to risk-aware BMDP criteria defined in Eq. (5.4) and also require sampling more models in order for the method to estimate a quantile with the needed accuracy.

Another interesting effect reported in Ring is that for $N = 8$, the trivial policy is picked a considerable number of times. Both in Ring and in Chain, EvC selects more often the output of BCR rather than that of NORBU, even though NORBU is slightly the most performing according to Table 5.2. In RFL, only the policies computed through solving different models sampled from the posterior with different γ's and NORBU are selected. The first kind of policies are preferred when the batch is very small ($N = 15$), however, the ratio inverts already for $N = 45$ with NORBU, which gets more and more chosen as N grows. Both the Trivial policy and the one returned by BCR are always discarded, stressing the superiority of NORBU in this environment typology. Surprisingly, EvC never selects neither SPIBB nor BOPAH, not even in RFL despite its good performance. This is likely due to the difficulty in estimating the quantiles of the performance of a non-deterministic policy, such as the output of SPIBB. The algorithm would require several sampled models higher than the bail-out hyperparameter.

5.4 Conclusions

This chapter presented Exploitation vs Caution (EvC), a method to first evaluate and then select the best risk-aware policies within a set of candidate policies in the context of offline solutions to Risk-aware Bayesian MDPs. The Risk-aware BMDP defines an elegant mathematical framework that balances the exploitation-caution trade-off in offline model-based sequential decision-making under uncertainty.

The set of candidate policies exploited by EvC contains the strategies obtained by solving not only the trivially learned MDP but also other MDPs with transition dynamics sampled from the Bayesian posterior (*e.g.* the one shown in Eq. (2.5)) using different discount factors and the

solutions of current offline MDP and RL solvers (SPIBB, BOPAH, BCR, NORBU). The estimate of risk in the presented algorithm provides a probabilistic guarantee for the actual performance of the resulting policy described in Theorem 1 (and possibly Theorem 2). The selected solution maximizes the risk-aware utility function of Eq. (5.4).

Since EvC is based on the parallel resolution of a great number of models sampled from the Bayesian posterior, we doubt that it could efficiently scale to select policies for MDPs with a large number of states and actions. However, the presented approach should be considered a valuable tool to be exploited for real-world problem-solving through MDP modeling. In such a case, time is an affordable resource since the safety of possible humans in the loop would be the priority.

In the future, we aim to improve EvC's method of generating the set of candidate policies. An interesting direction consists of incrementally enriching the set of candidate policies following some heuristics, *e.g.* policy improvement by genetic algorithms. An extension to compute robust policies for data-driven POMDPs could be envisaged if a consistent representation of the model uncertainty can be formalized.

Key Takeaways

- Bayesian MDP is a framework for incorporating model uncertainty in offline model learning and Offline RL
- The EvC paradigm is a robust baseline for offline policy selection that elegantly incorporates model uncertainty through the Bayesian formalism and selects the policy that maximizes a risk-aware objective over the Bayesian posterior of a set of candidate policies.
- EvC has been shown to be effective in selecting robust policies in a variety of MDPs and can be a useful tool for practitioners looking to apply offline planning and reinforcement learning in the real world.

The content of this chapter gave rise to the following work:

Giorgio Angelotti, Nicolas Drougard and Caroline Ponzoni Carvalho Chanel (2023a). 'An Offline Risk-aware Policy Selection Method for Bayesian Markov Decision Processes'. arXiv:2105.13431

☞See the arXiv preprint

In the upcoming chapter, we will focus on a specific application case of offline reinforcement learning that highlights the various limitations and challenges of the field. The chosen environment involves the interaction between a human operator and an automated system during a mixed-initiative teleoperated mission.

Chapter 6

Firefighter Robot Game Study Case: Robust POMDP model learning and solving

In this chapter, we address the problem of obtaining an adaptive interaction policy for the system described in Chapter 3, the Firefighter Robot game. This chapter can be seen as an extension of previous studies (Charles et al., 2018; G. Singh, Caroline P. C. Chanel and R. N. Roy, 2021), which modeled an interaction controller as a sequential decision-making process under uncertainty, taking into account both physiological and behavioral data of the human operator.

Charles et al. (2018) utilized a factored MDP to reduce the complexity of the environment, while G. Singh, Caroline P. C. Chanel and R. N. Roy (2021) employed a POMDP representation, assuming the human operator's mental state to be unobservable. Although Charles et al. (2018) did not test their model with real-world simulations, G. Singh, Caroline P. C. Chanel and R. N. Roy (2021) focused on optimizing the human operator's performance without considering the overall mission outcome.

We argue that the optimization criterion maximized by G. Singh, Caroline P. C. Chanel and R. N. Roy (ibid.) was oversimplistic, as enhancing the performance of only a subset of team members does not necessarily imply improved team performance. For example, a football coach aiming to optimize the defense's performance according to specific criteria might only focus on preventing goals, neglecting the importance of scoring goals and winning the match. Thus, in this chapter, our goal is to optimize the entire team's performance.

With this objective, we first attempt to infer a POMDP model from pre-collected demonstrations. In the study by G. Singh, Caroline P. C. Chanel and R. N. Roy (ibid.), a "trivial" POMDP was used without considering model uncertainty, and crucially, it assumed that an expert provided the optimal hyperparameters (*e.g.* the discount factor γ). However, we note that such an approach can be heavily influenced by the initial data set and model uncertainty. Consequently, inspired by the literature on risk-sensitive offline learning and hyperparameter selection discussed in Chapter 2, as well as our contribution to risk-sensitive offline policy selection for discrete Bayesian MDPs in Chapter 5, we aim to extend the EvC method to address the issue of risk-sensitive offline policy selection for the POMDP related to the Firefighter Robot Game that we will learn from demonstrations.

More explicitly, we seek answers to the following research questions: *Can we represent the dynamics of the Firefighter Robot Game as a POMDP? Can we learn it from demonstrations? Which*

hyperparameters should the practitioner use to deploy a robust, risk-sensitive policy?

Our robustness criterion aims to avoid low-score missions with any human operator. Consequently, the selected policy should be robust against both inter-subject variability and stochastic environmental deviations, such as multiple trees catching fire simultaneously.

Lastly, we will compare the safe policy selected by different approaches: Offline POMDP modeling with Risk-sensitive offline Bayesian policy selection, Offline MDP modeling with Risk-sensitive offline Bayesian policy selection, the batch collector policy, and a common-sense robust policy.

6.1 Sketch: EvC for POMDPs

Before discussing the specific application to the Firefighter Robot Game, it is essential to introduce the problem of offline risk-sensitive policy selection for a discrete POMDP learned offline from a batch of experiences. In Chapter 5, we proposed EvC to select the policy that maximizes a distributional risk-sensitive criterion (Equation 5.4) relying on a Bayesian representation of the model uncertainty (Eq. (2.5)). While for a discrete MDP, the model's Bayesian posterior obtained from data is directly represented as a Dirichlet distribution (see Equation 2.5), such practice is not directly applicable to POMDPs. However, to derive and deploy a risk-sensitive policy selection approach following the same idea as EvC, we must first find a way to encompass model uncertainty in the POMDP.

To address our specific proof-of-concept environment, we propose a method to include model uncertainty while representing the observation function of the POMDP. Thus, instead of sampling various transition functions from a transition Bayesian posterior (as in Algorithm 7), we will sample distinct observation functions from an observation Bayesian posterior and then learn a different transition function using Expectation-maximization (EM), keeping the sampled observation function fixed.

Furthermore, instead of maximizing the risk-sensitive utility function defined as the performance of a policy distributed according to model uncertainty (see Equation 5.4), we will maximize a risk-sensitive utility function describing the full cumulative return over a trajectory obtained using a policy and distributed according to model uncertainty. In simpler terms, we do not select the policy that maximizes the expected value over possible trajectories generated using a specific model but rather a policy that performs better in the worst $n\%$ cases, *e.g.* $n = 50$. Formally, we aim to find

$$\pi^* = \text{argmax}_{\Pi} \mathit{VaR}_{0.50}[G_\pi], \tag{6.1}$$

$$G_\pi = \sum_t R_t^\pi, \tag{6.2}$$

where R_t^π is the random variable representing the immediate return after each time step gained along a history while following the policy π, Π is the set of policies among which π^* is selected, and $\mathit{VaR}_{0.50}$ is an empirical estimate of the Value at Risk at risk level $q = 0.50$ ($\widehat{a_{0.50}}$ or the median). Refer to Definition 16 for the definition of *VaR*.

We believe that such modeling is more suitable for obtaining a robust policy since, according to model uncertainty, we want to reduce the chances of obtaining a low reward history and not the chance of obtaining a low reward history *averaged* over the model stochastic dynamics.

In the next section, we outline the proposed pipeline to first represent the problem as a POMDP, and also, as an MDP. Eventually, we solve the obtained decision processes with risk-sensitive baselines. Finally, we select the most robust policy using EvC. For the POMDP case, we develop an extension of EvC. It is worth remembering that the starting batch is composed of pre-collected

experiences in the form of features of both the game and the human operator processed over 10-second time windows. Therefore, a time step of the stochastic processes is what happens in a time interval of ten seconds during the mission. The features considered were described in Chapter 3 (see 3.2.2).

6.2 Model learning and solving

Partially following the spirit of the work in Caroline P. C. Chanel et al. (2020), we use a classifier to reduce the dimensionality of the measurements. However, instead of predicting the performance through behavioral and physiological markers related to a correct level of engagement of the human operator, which is a strong claim, we aim to detect whether a particular collection of measurements relative to a 10-second time interval[7] is likely to be part of a high-performant or a low-performant mission.

It is worth recalling that to make physiological markers such as HR and HRV comparable across participants, a normalization processing was performed. This involved subtracting the per-subject HR and HRV measured at rest one minute before each mission from the relative live (in mission) measurements. This normalization was an attempt to ensure that the inter-participant physiological data were standardized, allowing for more meaningful comparisons and analysis in the context of our human-robot system dynamics modeling.

In the following subsection, we describe how the human-robot system dynamics is modeled as a POMDP and as an MDP, taking into account model uncertainty, how these models are solved, and how a risk-sensitive policy is selected.

6.2.1 POMDP learning and solving

Our target is to learn a POMDP using a set of trajectories that were collected using a fixed, known policy (see Chap. 3). In particular, the policy used to collect the batch is uniformly random over the actions. In the specific case of the Firefighter Robot game, the set of actions $\mathcal{A}$ that the interaction controller can apply is $\mathcal{A} = \{$*manual,off; manual,on; auto,off; auto,on*$\}$ where *manual* or *auto* refers to the autonomy level of the robot and *off* or *on* refers to whether alarms concerning the mission are displayed on the Graphical User Interface (GUI) to the human operator. Concretely, at each time step, the controller decides the autonomy level of the system and its policy regarding notifications. It is important to note that when the robot's autonomy level is set to automatic (*e.g.* after the controller deploys action $a =$ *auto, off*), the human operator is relieved from active control, but she/he must still concentrate on tasks such as refilling the water tank and repairing leaks. Consequently, in the subsequent sections of this chapter, when we mention a fully automatic policy, *i.e.* a policy that consistently sets the autonomy level to automatic (with alarms), the human operator continues to play a crucial role in the mission's success.
To learn a POMDP that represents the human-robot team mission, we first propose considering the system as an HMM. Then, we transform this HMM into a POMDP by allowing the policy to change.

Nevertheless, when there is not a strong prior knowledge neither of $\mathcal{S}$, nor of the transition and observation probability functions, the shape of $\mathcal{S}$ is the biggest obstacle that hinders the learning of an HMM starting from a set of trajectories. Indeed, in the context of the Firefighter Robot Game, the observations are described by a high-dimensional feature vector with entries that can assume continuous, discrete, or boolean values.

[7]The 10-second time interval was specifically chosen to accommodate the processing needs of physiological features, such as Heart Rate (HR) and Heart Rate Variability (HRV).

With this in mind, rather than learning the true underlying HMM, we propose a method to fit a batch of histories onto an HMM characterized by low-dimensional and discrete states and observations. Subsequently, we transform this HMM into a POMDP that is easily solvable by state-of-the-art solvers.

To achieve this, it is necessary for $|\mathcal{S}| \in \mathbb{N}$. If $|\mathcal{S}|$ is too small, there is a risk of neglecting important information. Nevertheless, following the literature in the offline context, it seems preferable to use a more coarse-grained representation to avoid overfitting the batch (François-Lavet et al., 2019).

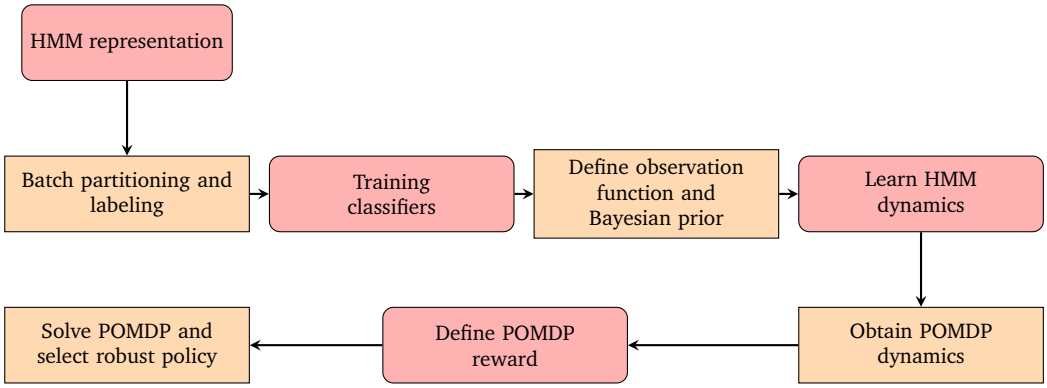


Figure 6.1: Steps of the method applied to represent, learn and solve a POMDP for the interaction controller of the Firefighter Robot Game.

The methodology applied follows eight steps (see Figure 6.1):

Step 1 (HMM representation):

To control the interaction mode (automatic or manual, and notifications on or off) in order to improve the performance of the team, we consider a latent state space such as

$$\mathcal{S} = \{m_{np,off}, m_{np,on}, m_{p,off}, m_{np,on}, a_{np,off}, a_{np,on}, a_{p,off}, a_{p,on}, g\}.$$

$m_{np,off}$ indicates that the robot's operation mode is manual, the measurement is part of a performant mission and the alarms are off; where $m_{np,on}$, indicates that the robot's operation mode is manual, the measurement is part of a non-performant mission and the alarms are on; $m_{p,off}$ for manual mode, performant, alarms off; $m_{p,on}$ for manual mode, performant, alarms on; $a_{np,off}$ automatic mode, non-performant, alarms off; $a_{np,off}$ automatic mode, non-performant, alarms on; $a_{p,off}$ automatic mode, performant, alarms off; $a_{p,on}$ automatic mode, performant, alarms on; g is an artificial absorbing state indicating the game is over or a premature end of a mission.

Furthermore, we consider an observation space such as

$$\Omega = \{\underline{m}_{np,off}, \underline{m}_{np,on}, \underline{m}_{p,off}, \underline{m}_{p,on}, \underline{a}_{np,off}, \underline{a}_{np,on}, \underline{a}_{p,off}, \underline{a}_{p,on}, \underline{g}\},$$

where each element of the set corresponds to the observation of a specific state. For example, the observation $\underline{m}_{np,off}$ could imply that the agent "observed" the state $m_{np,off}$, but its observation may be imperfect, meaning it cannot be certain that this is the actual state of the system. It is worth reminding that, in this context, the agent is the controller of the human-robot interacting system (see Figure 3.1).

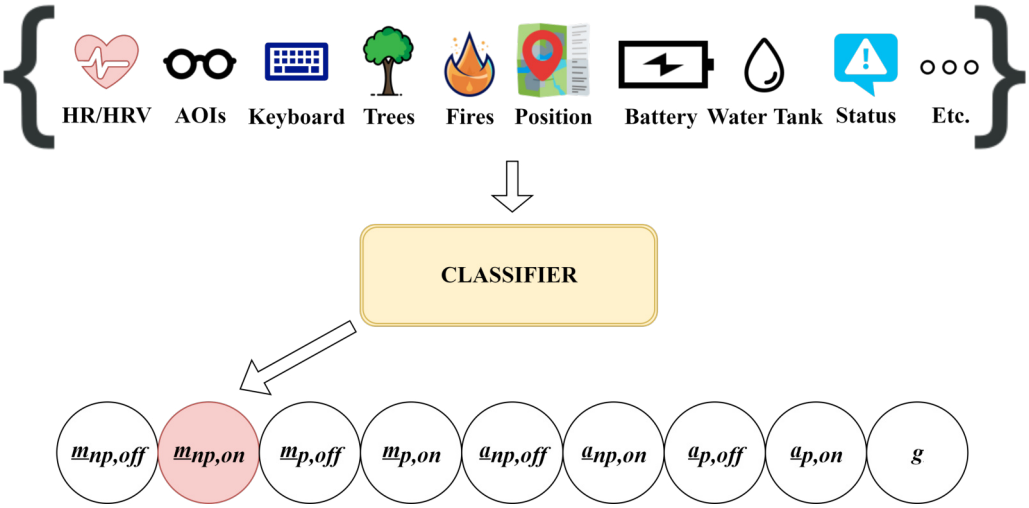


Figure 6.2: Example of the procedure to map a high dimensional array of measurements regarding the human operator, the robot, the mission, and the interaction to one of the nine possible HMM/POMDP observations $\in \Omega$ through a classifier. In this case, the classifier maps the high dimensional array of measurements (*i.e.* the features Y processed each 10-second time window) to the observation $\underline{m}_{np,on}$.

Step 2 (batch partitioning and labeling):

How to map the high-dimensional measurements obtained from the robot's sensors and from the devices used to measure the human's operator behavioral and physiological features to elements of Ω? Following the approach suggested by R. N. Roy et al. (2020), we propose performing this dimensionality reduction using multiple classifiers trained in a supervised manner (see Figure 6.2).

Supervised training requires ground truth labels, which we lack. This is because we do not know which 10-second time window (*i.e.* a decision step) high-dimensional observation corresponds to which state $s \in S$. Thus, we will first define a problem-specific method for assigning labels.

In the batch, 72 mission data, consisting of several feature vectors (i.e., multiple feature vectors Y_t with $t \in 0, \cdots, \mathrm{N}$, where N is the final mission time step), are labeled by the mission's global score (mission performance measured as the total number of extinguished fires). However, to infer the system state (including the mental engagement of a human operator) concerning performance, we need to evaluate a local measure of efficiency. The global score represents the cumulative sum of extinguished fires throughout an entire mission. Nevertheless, there may be time steps (10-second intervals) when a human operator performs well, even though no trees are on fire. How can we estimate whether a time step is associated with a latent state of performance or not?

We propose dividing the batch of missions into three parts. Initially, we assign a global score to each time step within a mission, equal to the entire trajectory's score. The first and last quartiles of missions according to the global score are selected as the subset $\mathcal{D}$. This assumption is reasonable because if a policy's quality is defined as the expected linear additive utility, low-scoring missions are likely characterized by low-scoring rewards at each time step throughout the whole history, and vice versa for high-scoring missions. Refer to Table 6.1 for descriptive statistics and Figure 6.3 for a graphical representation of the process.

We then select the best 20% of the first quartile and the worst 20% of the best quartile as the test set. This is because the better the performance of the whole mission, the more an overall low-score history is contaminated by high-performant single observations. Likewise, the worse the

Table 6.1: Descriptive Statistics of 72 mission scores in $\mathcal{D}$. During the 72 missions, the interaction controller deployed a uniform random policy. We report also the statistics of the global score of 3591 time steps composing the 72 missions. Notice that since low-score missions have fewer time steps, the quartiles are shifted toward higher values.

	Score						
	mean	std	min	25%	Median	75%	max
Type							
Mission	22.1	10.0	1.0	12.8	26.5	29.3	36.0
Time Step	24.9	8.0	1.0	21.0	28.0	30.0	36.0

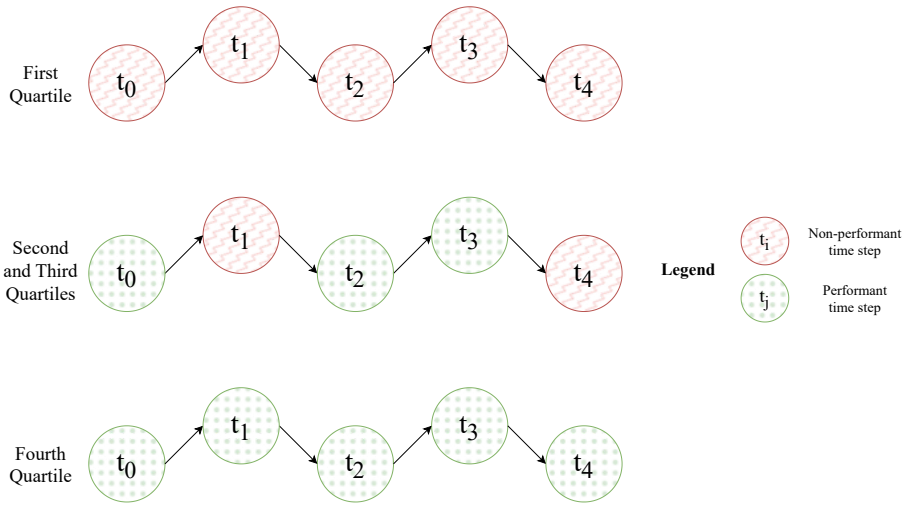


Figure 6.3: Example of how the batch is split. Missions in the batch are sorted according to the final score: the total number of fires extinguished. We suppose that during missions with a low score, the team has constant non-performant behavior and, vice versa, during missions with a very high score the team has constant performant behavior. Hence, the time steps of the missions in the First Quartile (with respect to the final score) are labeled as *non-performant* (red in the figure) and the ones in the top 25% (the Fourth Quartile) as *performant* (green). The time steps in the missions in the second and third quartiles form the set $\overline{\mathcal{D}}_{\mathcal{B}}$ that will be used to learn the transition dynamics since we suppose that there might be a frequent change in performance status during an average score mission.

performance of a high-score history, the more it could be contaminated by single observations of low-performant time steps.

In addition to this, we slightly rearrange the missions in the data sets to have in the test and the training sets missions related to different participants. The latter is very important for a generalization of the approach to a completely new participant since we want to extract a behavior that is as general as possible. Then, we labeled the observations present in the first quartile as *non-performant* according to the observed operation mode of the robot $m_{np,off}$ or $m_{np,on}$ or $a_{np,off}$ or $a_{np,on}$. Conversely, the observations that compose the last quartile are labeled as *performant*: $m_{p,off}$ or $m_{p,on}$ or $a_{p,off}$ or $a_{p,on}$.

Step 3 (training a classifier for dimensionality reduction and to describe model uncertainty):

Using the data set partitioned and labeled as mentioned above, we then train four different classifiers:

1. one that gives an observation of the mission performance, translating the features related

to the time step for which the robot is in manual mode and alarms (*i.e.* notifications) are activated;

2. one that gives an observation of the mission performance, translating the features related to the time step for which the robot is in manual mode and alarms are not activated;
3. one that gives an observation of the mission performance, translating the features related to the time step for which the robot is in automatic mode and alarms are activated;
4. one that gives an observation of the mission performance, translating the features related to the time step for which the robot is in automatic mode and alarms are not being provided.

More specifically, we train four Extra Tree Classifiers (Geurts, Ernst and Wehenkel, 2006). We choose Extra Tree Classifiers over Random Forests because they produce low variance outputs at the expense of bias. This choice fits like a glove to our problem, as we want to use a classifier that generalizes between different participants (who might yield measures that are differently distributed). For this purpose, we need a classifier that is less robust to outliers. However, the right trade-off between variance and bias is non-trivial. The classifiers are trained using a 10-fold Group (participant-wise) Shuffle Split cross-validation approach to select the best hyperparameters with respect to the balanced accuracy metric, initially with a Random Search and then with a Grid Search method. The validation set contains 20% of the data from the entire training set, and both sets include only missions carried out by different groups of participants. In doing so, we obtain the hyperparameters of the Extra Tree Classifiers that, on average, minimize the generalization error on the validation set.

Step 4 (observation function and Bayesian prior for model uncertainty):

The diagonal elements of the confusion matrix associated with each classifier enumerate the number of samples that were correctly classified by the respective algorithm. Conversely, the off-diagonal elements of the confusion matrix represent the number of samples that were classified as pertaining to a particular class (as determined by the column index), while their actual label belongs to a different class (as determined by the row index). We propose using the confusion matrices of the classifiers as Bayesian Dirichlet priors for the HMM observation function, in the same spirit as Eq. (2.5). In fact, normalizing a confusion matrix by row results in another matrix whose elements $O_{ij} = \Pr(O_t = o_j | S_t = s_i)$. These matrices will be referred to as the *trivial observation functions*. In Table 6.2, we present the trivial observation functions for each autonomy level and alarm notification state. It is important to note that for the artificial absorbing state g, we impose $\Pr\left(O_t = \underline{g} | S_t = g\right) = 1$ by construction.

Interpretability. We employ SHAP (S. M. Lundberg and S.-I. Lee, 2017) to calculate the Shapley Values of the observation features on the test set for each classifier. Specifically, the global Shapley values indicate how the model generally utilizes a feature to generate its output. Our results, as displayed in Figure 6.4, reveal that the physiological data of the human operator play a crucial role in the final prediction of the latent state. This suggests that monitoring the normalized HR, HRV, as well as the duration and number of fixations in different AOIs, is essential for inferring the performance level during a mission. Interestingly, these findings align with the work by Caroline P. C. Chanel et al. (2020), which claimed improved classification accuracy when physiological features are combined with behavioral ones.

Table 6.2: Confusion matrices computed on the test set for the different classifiers.

(a) Manual, Alarms ON. Sensitivity: 0.60, Specificity: 0.72, Balanced Accuracy: 0.66.

	Obs. Non-perf.	Obs. Perf.
Non-perf.	113 (0.72)	43 (0.28)
Perf.	71 (0.40)	105 (0.60)

(b) Manual, Alarms OFF. Sensitivity: 0.71, Specificity: 0.67, Balanced Accuracy: 0.69.

	Obs. Non-perf.	Obs. Perf.
Non-perf.	105 (0.67)	51 (0.33)
Perf.	51 (0.29)	125 (0.71)

(c) Auto, Alarms ON. Sensitivity: 0.75, Specificity: 0.59, Balanced Accuracy: 0.67.

	Obs. Non-perf.	Obs. Perf.
Non-perf.	99 (0.59)	70 (0.41)
Perf.	46 (0.25)	140 (0.75)

(d) Auto, Alarms OFF. Sensitivity: 0.64, Specificity: 0.70, Balanced Accuracy: 0.67.

	Obs. Non-perf.	Obs. Perf.
Non-perf.	91 (0.70)	39 (0.30)
Perf.	55 (0.36)	97 (0.64)

Step 5 (learning the HMM transition function accounting for model uncertainty):

Having used the worst and best quartiles concerning the global score of a mission to learn a Dirichlet distribution of observation functions, we use all the missions characterized by a global score that is in the middle between the two extremes to define $\overline{\mathcal{D}}_{\mathcal{B}}$ and to learn the latent dynamics of the HMM. Indeed, for this choice of $\overline{\mathcal{D}}_{\mathcal{B}}$ the assumption that the global score can be used as a representative of the local performance is rather speculative and not reliable. However, this dataset appears perfectly suited for learning the transition probabilities between latent states because we expect that changes in human engagement conditions are likely to occur along average score trajectories.

We start by using the learned classifier to accordingly retrieve the time series of HMM's observation O using the time series of features $Y \in \overline{\mathcal{D}}_{\mathcal{B}}$. Since the probability of observing a transition in a POMDP is

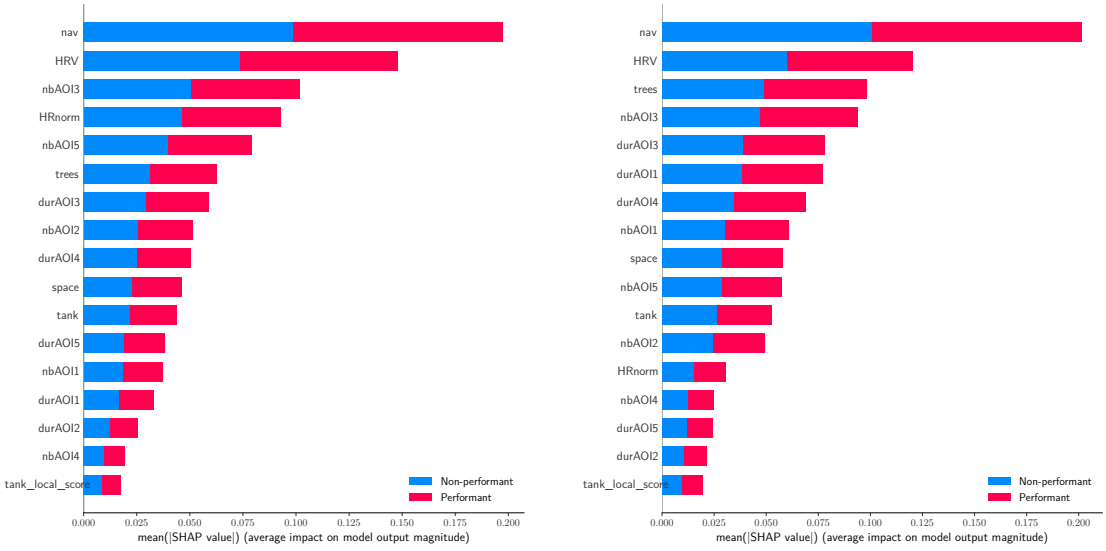
$$\begin{aligned}\Pr\big(O_{t+1}=o, S_{t+1}=s'|S_t=s\big) &= \sum_{a\in\mathcal{A}} \pi(a|s)\Pr\big(O_{t+1}=o, S_{t+1}=s'|S_t=s, A_t=a\big)\\ &= \sum_{a\in\mathcal{A}} \pi(a|s)\mathcal{O}(a,s',o)T(s,a,s') = \mathcal{O}(s',o)\sum_{a\in\mathcal{A}} \pi(a|s)T(s,a,s'),\end{aligned} \tag{6.3}$$

because the observation function $\mathcal{O}$ does not depend on the action here, we can learn the transition probability function of the HMM. The latter can be expressed as

$$\Pr\left(S_{t+1}=s'|S_t=s\right) = \sum_{a\in\mathcal{A}} \pi\left(a|s\right) T\left(s,a,s'\right) \tag{6.4}$$

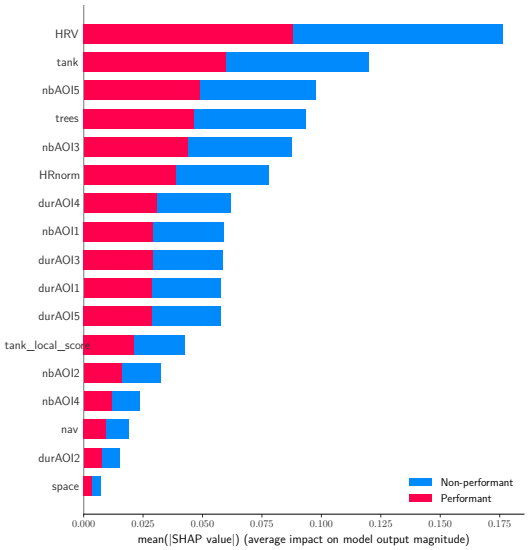
using the EM algorithm and imposing the normalization with respect to the known π with the Bayes' rule at each iteration. The transition function of the HMM, before the EM algorithm is launched, is initialized with the frequencies of transitions between the observations in the same data set. This is due to the empirical findings that, when possible, if EM starts from transition matrices that reflect the changes in the observed data set, then it will most likely converge to a meaningful result.

Inspired by EvC, we sample N_M observation functions from the Dirichlet prior and keep them fixed during the execution of EM. We also introduce a small pseudo count of 0.5 to every transition to avoid overfitting the data set. In this way, we obtain a set of distinct HMM models characterized

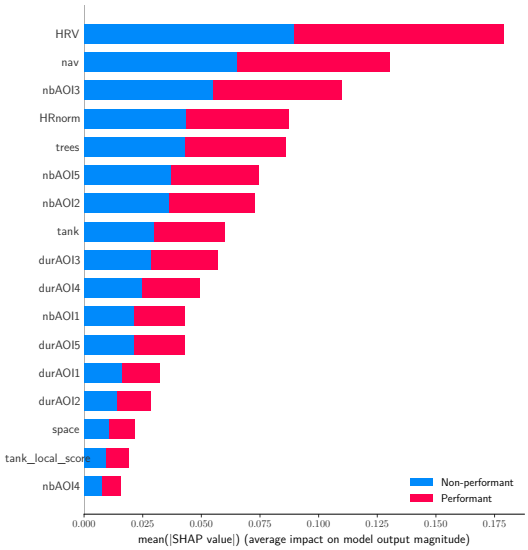


(a) Manual Mode - Alarms Off

(b) Manual Mode - Alarms On



(c) Automatic Mode - Alarms Off



(d) Automatic Mode - Alarms On

Figure 6.4: Shapley values for the features of each classifier (autonomy mode/notification status) obtained via SHAP's Tree Explainer. The features include *HRV* (Normalized Heart Rate Variability), *HRnorm* (Normalized Heart Rate), $durAOI_i$ (duration of fixations in AOI i), $nbAOI_i$ (number of fixations in AOI i), *nav* (number of times a navigation key has been pressed), *space* (number of times the space bar has been pressed), *trees* (number of extinguished fires), *tank* (level of the water tank), and *tank_local_score* (changes in the level of the water tank).

by pairs of a sampled observation function and a learned transition function.

Step 6 (obtaining the POMDP transition function from the HMM one):

For each HMM and its corresponding learned transition function, we can develop an action-wise transition function T for a POMDP by trying to find a T that solves Eq. (6.4). We are going to do it by incorporating prior knowledge of the chosen representation. This approach is guided more by intuition than mathematical rigor because, ideally, the problem of obtaining the single addends of $T(s, a, s')$ in the sum in Eq. (6.4) does not have a unique solution. Obtaining $T(s, a, s')$ is impossible without other constraints. Given that the data collector policy $\pi(a|s)$ is uniformly

random, we have:

$$\pi(a|s) = \frac{1}{|\mathcal{A}|}, \tag{6.5}$$

for all $(a, s) \in \mathcal{A} \times \mathcal{S}$, which allows us to express:

$$\Pr(S_{t+1} = s'|S_t = s) = \frac{1}{|\mathcal{A}|} \sum_{a \in \mathcal{A}} T(s, a, s'). \tag{6.6}$$

The given expression is now more manageable, as the $\pi(a|s)$ terms are constant. However, determining the unknown $T(s'|a, s)$ remains impossible, as there could be an infinite number of valid combinations.

At this stage, we rely on our intuition and prior knowledge to establish a rule for T. We propose to set T as follows:

$$T(s, a, s') = |\mathcal{A}|\Pr(S_{t+1} = s'|S_t = s)\frac{\delta^a_{s,s'}}{\sum_{a'} \delta^{a'}_{s,s'}}, \tag{6.7}$$

where $\delta^a_{s,s'} = 1$ if the transition $(s, a) \rightarrow s'$ can occur, and 0 otherwise. For example, the transition $T(m_{np,off}, (auto; on), m_{np,off})$ cannot occur as the next state can only be $a_{np,on}$, $a_{p,on}$, or g. Moreover, since g is an absorbing state artificially created and transitions from it are never seen in the data, we set:

$$T(g, a, g) = 1 \tag{6.8}$$

and

$$T(g, a, s' \neq g) = 0. \tag{6.9}$$

In Tables 6.3-6.6 we show the transition probabilities for the POMDP learned from the "trivial" observation functions.

Table 6.3: Transition probabilities for action: put mode manual and activate alarms. Rows are current states and columns are next states.

	$m_{np,off}$	$m_{np,on}$	$m_{p,off}$	$m_{p,on}$	$a_{np,off}$	$a_{np,on}$	$a_{p,off}$	$a_{p,on}$	g
$m_{np,off}$	0	0.917	0	0.067	0	0	0	0	0.017
$m_{np,on}$	0	0.917	0	0.065	0	0	0	0	0.019
$m_{p,off}$	0	0.027	0	0.956	0	0	0	0	0.017
$m_{p,on}$	0	0.032	0	0.956	0	0	0	0	0.012
$a_{np,off}$	0	0.945	0	0.051	0	0	0	0	0.005
$a_{np,on}$	0	0.931	0	0.063	0	0	0	0	0.006
$a_{p,off}$	0	0.018	0	0.973	0	0	0	0	0.009
$a_{p,on}$	0	0.016	0	0.965	0	0	0	0	0.020
g	0	0	0	0	0	0	0	0	1

Step 7 (definition of the immediate reward of the POMDP):

Since the global score of a mission is the sum of the number of extinguished fires along a trajectory, a rational choice for the expected reward function is the average number of fires extinguished by the human-robot team while being in the latent state s.

Our procedure assumes that the latent state is perfectly known only for the transitions in $\mathcal{D}$. To begin with, we recover a histogram (distribution) for the number of fires extinguished in each state using this dataset (see Figure 6.5). These distributions define a stochastic reward function. In addition, we also establish a deterministic reward function that assigns a per-state reward based on the expected value of the associated distribution. This deterministic reward function will be

Table 6.4: Transition probabilities for action: put manual mode and deactivate alarms. Rows are current states and columns are next states.

	$m_{np,off}$	$m_{np,on}$	$m_{p,off}$	$m_{p,on}$	$a_{np,off}$	$a_{np,on}$	$a_{p,off}$	$a_{p,on}$	g
$m_{np,off}$	0.914	0	0.069	0	0	0	0	0	0.017
$m_{np,on}$	0.914	0	0.067	0	0	0	0	0	0.018
$m_{p,off}$	0.020	0	0.963	0	0	0	0	0	0.017
$m_{p,on}$	0.024	0	0.964	0	0	0	0	0	0.012
$a_{np,off}$	0.940	0	0.055	0	0	0	0	0	0.005
$a_{np,on}$	0.923	0	0.071	0	0	0	0	0	0.006
$a_{p,off}$	0.015	0	0.976	0	0	0	0	0	0.009
$a_{p,on}$	0.013	0	0.967	0	0	0	0	0	0.020
g	0	0	0	0	0	0	0	0	1

Table 6.5: Transition probabilities for action: put automatic mode and activate alarms. Rows are current states and columns are next states.

	$m_{np,off}$	$m_{np,on}$	$m_{p,off}$	$m_{p,on}$	$a_{np,off}$	$a_{np,on}$	$a_{p,off}$	$a_{p,on}$	g
$m_{np,off}$	0	0	0	0	0.941	0	0.042	0	0.017
$m_{np,on}$	0	0	0	0	0.940	0	0.042	0	0.018
$m_{p,off}$	0	0	0	0	0.020	0	0.963	0	0.017
$m_{p,on}$	0	0	0	0	0.025	0	0.963	0	0.012
$a_{np,off}$	0	0	0	0	0.921	0	0.074	0	0.005
$a_{np,on}$	0	0	0	0	0.916	0	0.079	0	0.006
$a_{p,off}$	0	0	0	0	0.023	0	0.968	0	0.009
$a_{p,on}$	0	0	0	0	0.019	0	0.961	0	0.020
g	0	0	0	0	0	0	0	0	1

Table 6.6: Transition probabilities for action: put automatic mode and deactivate alarms. Rows are current states and columns are next states.

	$m_{np,off}$	$m_{np,on}$	$m_{p,off}$	$m_{p,on}$	$a_{np,off}$	$a_{np,on}$	$a_{p,off}$	$a_{p,on}$	g
$m_{np,off}$	0	0	0	0	0.944	0	0.039	0	0.017
$m_{np,on}$	0	0	0	0	0.943	0	0.038	0	0.018
$m_{p,off}$	0	0	0	0	0.021	0	0.963	0	0.017
$m_{p,on}$	0	0	0	0	0.025	0	0.962	0	0.012
$a_{np,off}$	0	0	0	0	0.932	0	0.064	0	0.005
$a_{np,on}$	0	0	0	0	0.929	0	0.066	0	0.006
$a_{p,off}$	0	0	0	0	0.031	0	0.961	0	0.009
$a_{p,on}$	0	0	0	0	0.026	0	0.954	0	0.020
g	0	0	0	0	0	0	0	0	1

utilized during the policy evaluation and selection stages of the methodology (Step 8, detailed later). Table 6.7 summarizes the reward function. This precaution is performed since, in order to use a state-of-the-art POMDP solver (*i.e.* SARSOP), we must employ a non-stochastic reward function.

Step 8 (solving the POMDPs and maximizing a risk-sensitive criterion):

Ultimately, the trivial POMDP (the one learned starting from the trivial observation function) is solved using SARSOP for several discount factors γ, yielding various optimal policies. It is worth noting that, as discussed in Chapters 2 and 5, solving a sequential decision-making problem with a shorter planning horizon could lead to more performant policies if the model was learned offline.

Algorithm 9 outlines the proposed solving procedure. Lines 2 to 5 implement the sampling process described in Step 5, in which $\mathrm{N}_M = 10^4$ observation functions are sampled from the POMDP

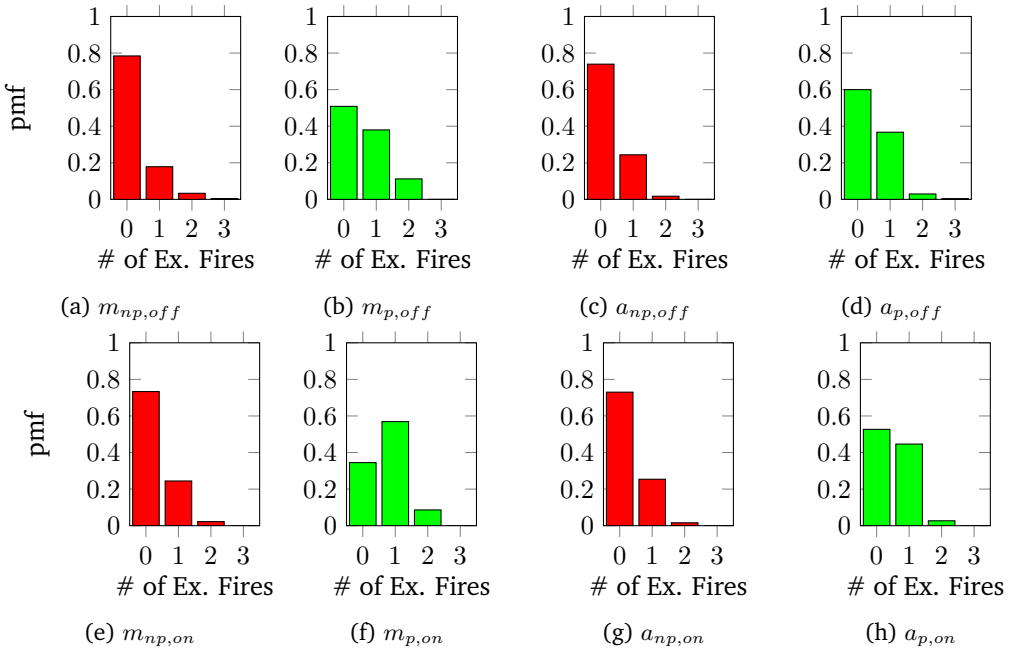


Figure 6.5: Stochastic Reward Function. Histogram representing the Probability Mass Function (pmf) of the number of extinguished fires in a 10 seconds window (1 time step of the discrete-time decision process) for a given state (autonomy level, performance, and alarm setting). The maximum probability in the case of $m_{p,on}$ corresponds to 1 extinguished tree per time step (0.57). For the other states the maximum probability corresponds to 0 extinguished trees. During performant time steps the human-robot team tends to extinguish a higher number of fires.

Table 6.7: POMDP deterministic rewards for each hidden state, independently of the action.

State	$m_{np,off}$	$m_{np,on}$	$m_{p,off}$	$m_{p,on}$	$a_{np,off}$	$a_{np,on}$	$a_{p,off}$	$a_{p,on}$	g
Reward	0.257	0.289	0.603	0.741	0.257	0.333	0.432	0.493	0.0

posterior distributions and used to learn the transition functions through EM. Subsequently, Lines 6 to 9 of Algorithm 9 solve the trivial POMDP for different values of γ. Finally, each computed policy is evaluated across N_M different learned POMDPs, and, as in EvC, the one that maximizes a risk-sensitive criterion is selected. We decided to maximize the $\mathit{VaR}_{0.5}$ metric (the median). Although this choice may not seem very risk-sensitive, there is an experimental rationale behind it: we will later validate our results using non-parametric statistical tests that examine discrepancies in the median of the distributions. Most importantly, as explained at the beginning of the chapter, the criterion used to select the POMDP adaptive policy differs from the one used in Chapter 5. Instead of maximizing the $\mathit{VaR}_{0.5}$ with respect to the distribution of expected cumulative rewards across different models, we maximize the $\mathit{VaR}_{0.50}$ (see Equations 6.1 and 6.2) with respect to the distribution of cumulative rewards over $\mathrm{N}_E = 10^4$ trajectories for each of the $\mathrm{N}_M = 10^4$ different models. In other words, we do not perform the mean over trajectories. We believe that this protocol adheres even more closely to the risk-sensitive optimization goal. This policy evaluation and selection procedure is illustrated in Lines 10 to 17 of Algorithm 9.

Robust POMDP policy In Table 6.8, we present various robustness and performance metrics calculated from the empirical distribution of returns for each tested policy (refer to Equation 6.2 and Lines 11-15 of Algorithm 9). To provide a comprehensive analysis, we not only simulated trajectories for policies derived from SARSOP and different discount factors (Lines 7-9 of Algorithm 9), but also included results from a random strategy, a fully automated strategy with and without alarms,

Algorithm 9: Pseudocode Algorithm for POMDP solving and Risk-sensitive Policy Selection

Input: Trivial POMDP, Γ set of discount factors, $\overline{\mathcal{D}}_{\mathcal{B}}$, N_M number of models to sample/learn, N_E number of histories per model
Output: π^*

```
1  Initialization: M ← ∅ (empty list)
2  for i from 1 to N_M do
3      Sample observation functions from the POMDP posterior distributions
4      Learn POMDP transitions using EM(observation functions, D̄_B)
5      Append new POMDP model to M
6  Π ← ∅ (empty list)
7  forall γ in Γ do
8      π ← Solve(Trivial POMDP, γ)
9      Append π to Π
10 forall π in Π do
11     G_π ← ∅ (empty list)
12     forall pomdp in M do
13         for i from 1 to N_E do
14             R ← Cumulative reward of a generated trajectory
15             Append R to G_π
16 π* ← arg max_{π∈Π} VaR_0.5[G_π]
17 return π*
```

and a fully manual policy with and without alarms.

There are a few points worth noting. Firstly, the minimum value of each distribution is approximately 0.3. This outcome results from a transition to the game-over state after the initial ten seconds, during which the autonomy level is set to manual with alarms activated. Although this event is clearly unrealistic, it stems from an assumption in our modeling. We introduced a game-over state to account for the possibility that various states and performance levels could prematurely terminate the simulation with different probabilities. However, in reality, it is highly improbable that this would occur after only ten seconds.

The maximum value for both the fully manual strategy with alarms and the SARSOP adaptive policies is 44.5. This figure corresponds to a scenario in which the autonomy level remains fixed at fully manual with alarms activated, and the team consistently performs well. According to the empirical VaR at a risk level of 0.25, the most robust policy is the fully automated strategy without alarms. This policy also exhibits the least variability, as indicated by its standard deviation. On the other hand, the fully manual policy with alarms results in a distribution with the highest third quartile, suggesting that this strategy might be the most effective when working with high-performing human operators.

Ultimately, the procedure selected the policy obtained by solving the trivial POMDP with a discount factor (γ) of 0.98, as it maximizes the $\mathit{VaR}_{0.50}$ (see Equation 6.1 and Line 16 of Algorithm 9). Coincidentally, this chosen policy also has the highest mean value.

Note that the state is characterized by directly observable features (*e.g.* autonomy level of the robot and notification status of the GUI) and a non directly observable features (Is the 10-

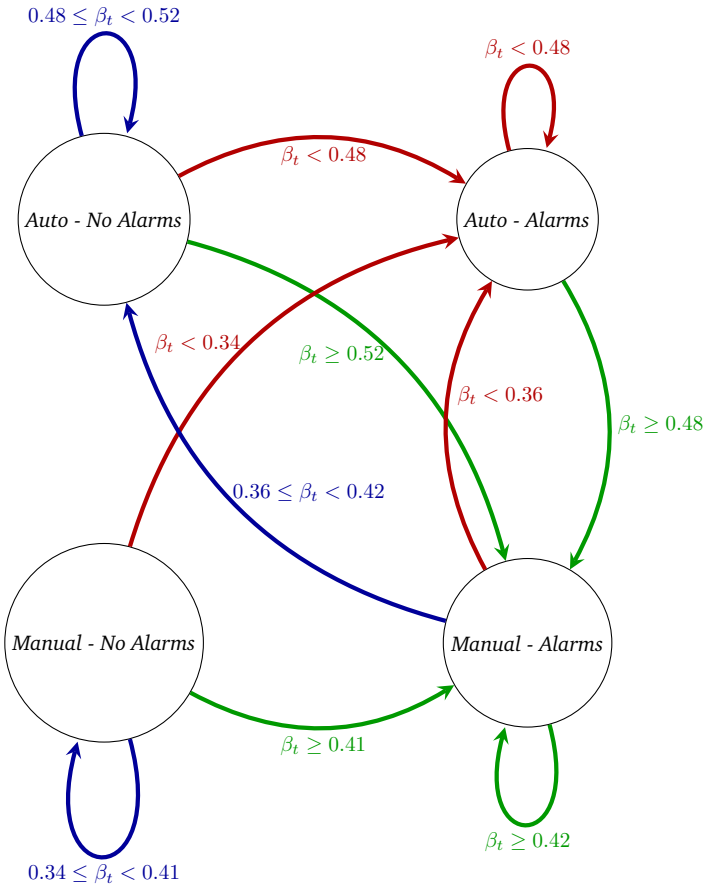


Figure 6.6: Selected robust POMDP policy ($\gamma = 0.98$). Which action is taken and when? Let β_t be the belief of performance for an observed state (Manual/Auto - Alarms/No Alarms) and time step t. Different values of β_t lead to transitions to different autonomy and alarm modes. It is worth noting that *Auto - Alarms* is an "absorbing" modality for non performant missions, while *Manual - Alarms* is an "absorbing" modality for performant missions. Moreover, since no arrows lead to *Manual - No Alarms* and the initial modality *Manual - Alarms*, the former modality is never reached during a mission that follows the robust POMDP policy.

second time interval typical of a performant or a non-performant mission?). Let β_t be a belief state marginalized over performance states for a given autonomy level and alarm modality. For example, if the modality of the robot at time t is x and the status of the notifications of the alarms is z, then

$$\beta_t = \sum_{s \in \mathcal{S}_{perf.}} b_t(s) = b_t(x_{p,z}). \tag{6.10}$$

with $\mathcal{S}_{perf.} = \{m_{p,on}, m_{p,off}, a_{p,on}, a_{p,off}\}$. The belief state is computed (*i.e.* updated) using the trivial POMDP. Please remember that the modality of the robot is fully observable, but not the status of performance. The next action is chosen according to the value of β_t (see Figure 6.6). When β_t is estimated to be around 50%, hence when the controller is doubtful whether that period of 10 seconds is typical of a performant or a non-performant mission, the autonomy level might be switched to automatic and the notifications might not be provided in the GUI. Since alarms are believed to provide useful information to the human operator, but more to a distracted player than to a focused one, removing the alarms could be interpreted as an action to *"gather information"* to assess the degree of engagement of the human operator and hence to better predict the performance degree of the next state. When the estimated performance probability is

Table 6.8: Comparison of different policies ran in simulation. Different robust and performance metrics are computed on the empirical distribution of returns. The best values in each column are in **bold**.

Policy	Mean	Std. Dev.	Min	25%	Median	75%	Max
Random	19.6	10.7	**0.3**	9.9	21.7	29.0	38.3
Manual/Alarms	23.2	13.9	**0.3**	10.2	24.6	**35.4**	**44.5**
Manual/No Alarms	19.1	11.7	**0.3**	8.0	19.9	30.0	36.3
Auto/Alarms	17.8	9.5	**0.3**	9.0	20.1	26.5	29.8
Auto/No Alarms	17.9	**7.3**	**0.3**	**14.1**	20.3	23.3	26.2
SARSOP $\gamma = 0.70$	23.3	13.8	**0.3**	10.4	25.0	35.4	**44.5**
SARSOP $\gamma = 0.80$	23.4	13.8	**0.3**	10.5	25.1	35.4	**44.5**
SARSOP $\gamma = 0.90$	23.6	13.6	**0.3**	11.1	25.5	35.4	**44.5**
SARSOP $\gamma = 0.97$	23.7	13.3	**0.3**	11.8	26.0	35.2	**44.5**
SARSOP $\gamma = 0.98$	**23.8**	13.1	**0.3**	12.0	**26.1**	34.5	**44.5**
SARSOP $\gamma = 0.99$	23.3	12.0	**0.3**	13.6	25.1	33.0	**44.5**
SARSOP $\gamma = 0.999$	21.4	10.8	**0.3**	13.6	22.8	29.9	**44.5**

lower than 35%, the robot is pessimistically set to automatic mode, and alarms are provided to the human operator. This approach ensures that as much information as possible is provided, and the robot is driven automatically to minimize casualties.

In the next section, we present an alternative modeling that relaxes the assumption of partial observability. Eventually, we will evaluate the policies yielded by both approaches in laboratory experiments.

6.2.2 MDP learning and solving

We are going to compare the results given by modeling the system as a POMDP against an MDP representation. To do so, we keep the split of the data set described in the previous section and also the classifiers used for dimensionality reduction. However, in this case, we will assume that the output of the classification is the state of the system and not just an observation. Therefore, we model the system as an MDP with nine states

$$\mathcal{S} = \{m_{np,off}, m_{np,on}, m_{p,off}, m_{np,on}, a_{np,off}, a_{np,on}, a_{p,off}, a_{p,on}, g\}$$

and four actions. The time series Y of measurements in $\overline{\mathcal{D}}_{\mathcal{B}}$ will be accordingly classified. The transitions between states will be counted and integrated into a Dirichlet posterior Eq. (2.5). The same reward function as the POMDP model will be used. However, the MDP will be solved using the deterministic policy risk-sensitive algorithms adopted in Chapter 5: BCR and NORBU (Petrik and Russel, 2019; Lobo, Ghavamzadeh and Petrik, 2021). Likewise, the trivial MDP will be also solved with several discount factors γ (*e.g.* 0.1, 0.2, 0.3, 0.4, $\ldots$, 0.9, and 0.99).

Robust MDP policy In this study, we employ the EvC method to select the most robust policy based on the VaR at risk level 0.5 (the median). It is important to note that we utilize the original EvC implementation, which calculates risk-sensitive metrics using the distribution of performance (refer to Equation 5.4), rather than returns. As for the EvC hyperparameters, we choose a $\gamma_{ev} = 0.99$.

Interestingly, the selected policy, NORBU with $\lambda = 0.5$, opts to consistently set the autonomy level to manual mode and enable alarms. To provide a comprehensive analysis, we also simulate

the distribution of returns for this policy using the trivial MDP. The initial state is randomly selected from $m_{np,on}, m_{p,on}$ with equal probability. The simulation results are presented in Table 6.9.

It should be noted that simulating in the trivial MDP yields a distribution that is more concentrated on lower values. This is because the probability of reaching a game-over state (with an average transition probability of 0.063 per time step from $s \neq g$) is higher in the trivial MDP compared to the trivial POMDP (with a probability of 0.013) and a majority of sampled POMDPs.

Table 6.9: Descriptive statistics of the simulated distribution of returns in the trivial MDP environment using the MDP robust adaptive policy.

Policy	Mean	Std. Dev.	Min	25%	Median	75%	Max
MDP	16.0	12.3	0.3	5.3	12.9	25.7	44.5

6.3 Experiments

In order to evaluate the robust policies obtained, experiments were conducted in laboratory settings following a specific protocol. The robust POMDP policy was compared against the robust MDP policy, a random policy, and a fully automatic (with alarms on) policy. The goals of these experiments were:

- to assess the generalization capability of the classifier on both low and high-performing missions;
- to evaluate the accuracy of the POMDP model by studying the correlation between the marginalized belief of performance β and the score of a mission;
- to demonstrate that the policies obtained via the POMDP and the MDP are more performant and robust than the data collector policy;
- to show that a model with hidden states better describes the problem of controlling the interaction in the Firefighter Robot Game by illustrating that the policy obtained with the POMDP is more performant and robust than that obtained via an MDP;
- to evaluate if an adaptive policy is as robust as the fully automatic policy;
- to assess whether an adaptive policy decreases the mental workload of the human operator;
- to examine whether the interaction is perceived as more fluent when the adaptive (POMDP) policy is deployed.

6.3.1 Materials and methods

Participants

A total of 26 participants (mean age 28.6, sd. 5.7; 7 females), all students from ISAE-SUPAERO, were recruited through electronic mail announcements and oral advertising. All participants subscribed to social security and civil responsibility insurance and provided their informed consent. The experiment was approved by the local ethical committee of the University of Toulouse[8]. Figure 6.7 shows a participant taking part in the experiment.

[8] CER-2018-070.

Figure 6.7: A participant tanking part in the experiment at ISAE-SUPAERO lab facilities.

Experimental protocol

The experimental procedure, illustrated in Figure 6.8, was enacted for each participant as follows:

1. (5 minutes) - Participants read the experiment goals information and provided approval and consent for data recording and analysis.
2. (5 minutes) - The Firefighter Robot Game was explained to the participants, and any questions they had were answered.
3. (4 minutes) - Participants wore devices to measure physiological markers (ECG). The Eye Tracker was calibrated by the practitioner, and the participant stared at a black cross on the screen for 1 minute to measure their physiological baseline markers.
4. (10 minutes) - Participants trained on a test mission. The robot was in automatic mode for the first 5 minutes and then in manual mode for the next 5 minutes. The interaction controller provided notifications and alarms during the entire mission.
5. (few seconds) - Participants were asked to complete the Karolinska Sleepiness Scale (KSS) questionnaire (Åkerstedt and Gillberg, 1990). The KSS is a survey that is generally used to measure subjective fatigue.
6. The following loop was repeated 4 times with 4 different interaction control policies (and pseudo-randomized between participants) to be tested: the policy obtained by robustly solving the POMDP, the one given by a robust resolution of the MDP, a random policy (*i.e.* the same used to collect data in the previous experiment, see the work in Caroline P. C. Chanel et al. (2020)), and lastly, the policy which is trivially believed to be the most robust, fully automatic mode with alarms.
   (a) (1 min) - The human operator stared at a black cross on the screen for 1 minute to measure their physiological baseline markers.
   (b) (10 min) - The user performed a mission for which the policy of the controller was randomly sampled (without repetition) from the 4 possible options. This approach ensured that different users likely experienced distinct policies in different orders.
   (c) (3 min) - After each mission, users completed the NASA Task Load Index (NASA-TLX) questionnaire (Hart and Staveland, 1988) to provide their subjective feedback concerning the effort made to carry out the mission.

(d) (2 min) - Users completed the Fluency questionnaire, inspired by the work in Hoffman (2019), in order to provide their subjective feedback concerning the fluency of the (adaptive) policy being used to control the interaction.

(e) (2 min) - A break was offered to the participants.

10. (few seconds) - Participants were invited to answer the KSS questionnaire again.

The experiment concluded with a data records check and removal of the ECG equipment from the participant. The entire experiment lasted approximately 1 hour and 30 minutes, including training, device setup, surveys, and the mission runs.

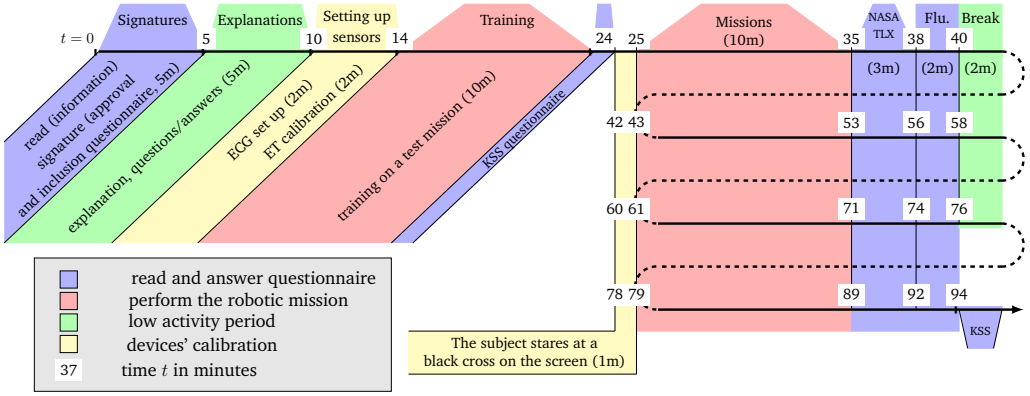


Figure 6.8: Timeline of the experimental procedure (from left to right). The participants experience different stages: signatures, explanations, training, sensors' set up, and calibration. Then, the following steps are repeated four times with a different interaction control policy during the mission: recording physiological baselines at rest, mission runs, NASA TLX questionnaire, fluency questionnaire, 2-minute break.

Data collection and processing

Several variables were collected and processed during the study, including features related to the robotic system, mission parameters, and participant interactions with the ground station (*i.e.* keyboard inputs and mouse clicks). In addition, eye-tracking and Electrocardiogram (ECG) data were also obtained. All these variables were streamed, processed, and recorded using the Lab Streaming Layer (LSL)[9], a widely recognized middleware within the neuroergonomics community. Developed by the Swartz Center for Computational Neuroscience (UCSD), LSL facilitates the unified collection of experimental data, ensuring time-synchronization and online access while mitigating networking issues. LSL also offers a software tool called LabRecorder, which enables researchers to select and record all relevant data for subsequent post-processing.

The LSL device used in this study was accompanied by data stream software developed by Mega Electronics for LSL[10]. Both raw ECG data (500 Hz) and Inter-Beat Intervals (IBIs) also known as RR peak intervals were recorded using the Faros ECG device. The IBIs were utilized to calculate Heart Rate (HR) and Heart Rate Variability (HRV) in real-time using Python scripts.

Additionally, Python scripts were developed based on the SMI SDK and LSL to create streams containing gaze and fixation event data measured by the SMI Eye Tracker (ET). These scripts also processed ET-related data online, including the total number of fixations within areas of interest (AOIs) and their total duration every 10 seconds (refer to Figure 3.4 in Chapter 3).

[9] See https://github.com/sccn/labstreaminglayer [Online: accessed 13-04-2023].
[10] Software available at https://github.com/bwrc/faros-streamer-2 [Online: accessed 13-04-2023].

Finally, Python scripts were employed to process other human and system-related data online, such as the total number of keystrokes, mouse clicks, external tank levels, and more.

The data streams and processing pipelines enabled the online availability of all features used for classification (*i.e.* to define the Y vector). Specifically, at the end of each 10-second time window, the collected features were processed to create the feature vector Y. This vector was then input into the classifier, which provided the observation used for belief state updates and policy readings. It is important to note that the classifier outputs, actions, and belief states were also recorded using LSL and LabRecorder.

Overall mission performance was assessed by the total number of extinguished fires. A human operator was required to manage several tasks simultaneously, including refilling the external tank, providing manual control when necessary, and monitoring the robot's parameters (water reserve, battery, temperature, and automation level). The global mission score was intrinsically dependent on the successful management of all these tasks.

Subjective feedback from participants was collected using the Karolinska Sleepiness Scale (KSS) (Åkerstedt and Gillberg, 1990), the NASA Task Load Index (NASA-TLX) (Hart and Staveland, 1988), and a Fluency questionnaire inspired by the work in (Hoffman, 2019).

The KSS is a survey commonly used to measure participants' subjective fatigue. A 9-point KSS was employed in the experiment, with higher scores indicating significant drowsiness. The NASA-TLX is a survey designed to assess subjective task workload, calculating an overall workload score based on a weighted average of six dimensions: mental demand, physical demand, temporal demand, performance, effort, and frustration. In the experiment, we used the HTML version[11], which automatically computes the score. It is crucial to note that a high NASA-TLX score correlates with a high subjective workload.

The Fluency questionnaire[12] serves as subjective feedback regarding the fluency of the (adaptive) policies used to control the interaction. Even in instances where efficiency does not improve, it is possible to enhance the sense of fluency in the task. This questionnaire evaluates several dimensions, including direct measures and downstream outcomes, such as the trust human teammates have in the system, the sense of improvement and system's commitment, working alliance (feelings of comfort, respect, and belief in the system's skills), and positive teammate traits. It also includes a self-report of commitment and sense of importance as human's individual measures. The use of some reversed scales helps reduce response bias and ensures the accurate measurement of the user's opinion.

In the following section, we present the metrics and results related to the various goals of this experiment.

6.3.2 Results

Evaluation of the classifiers

To assess the effectiveness of the classifiers, we calculate their balanced accuracy using all time steps from missions with global scores either below 20 or above 30. These thresholds were chosen based on the criteria utilized to partition the dataset for training the classifier. Time steps in missions with global scores less than 20 were labeled as non-performant, while those with scores greater than 30 were labeled as performant.

[11] Available at https://www.keithv.com/software/nasatlx/ [Online: accessed 13-04-2023].
[12] Available at https://gitlab.isae-supaero.fr/a.moreira-pinto/heroic-questionnaire/-/blob/master/fluency.html [Online: accessed 13-04-2023].

The classification results are summarized in Table 6.10. The balanced accuracy of the classifier in manual mode with alarms is 0.60, in manual mode without alarms it is 0.53, in automatic mode with alarms it is 0.52, and in automatic mode without alarms it is 0.50. These confusion matrices differ significantly from those used to learn the interaction model (as shown in Table 6.2) and derive the adaptive policies. Notably, in the automatic mode, the classification efficiency appears to be comparable to flipping a fair coin.

Table 6.10: Confusion matrices computed on the experimental results for the different classifiers. In parentheses the row normalized values.

(a) Manual, Alarms ON. Sensitivity: 0.38, Specificity: 0.81, Balanced Accuracy: 0.60.

	Obs. Non-perf.	Obs. Perf.
Non-perf.	1484 (0.81)	346 (0.19)
Perf.	349 (0.62)	217 (0.38)

(b) Manual, Alarms OFF. Sensitivity: 0.37, Specificity: 0.69, Balanced Accuracy: 0.53.

	Obs. Non-perf.	Obs. Perf.
Non-perf.	108 (0.69)	49 (0.31)
Perf.	43 (0.63)	25 (0.37)

(c) Auto, Alarms ON. Sensitivity: 0.54, Specificity: 0.50, Balanced Accuracy: 0.52.

	Obs. Non-perf.	Obs. Perf.
Non-perf.	383 (0.50)	376 (0.50)
Perf.	146 (0.46)	168 (0.54)

(d) Auto, Alarms OFF. Sensitivity: 1.00, Specificity: 0.01, Balanced Accuracy: 0.50.

	Obs. Non-perf.	Obs. Perf.
Non-perf.	1 (0.01)	167 (0.99)
Perf.	0 (0.00)	70 (1.00)

To determine if there is a significant difference between the classification outcomes and the previous results (Table 6.2), we conducted a two-proportion Welch's t-test for each of the eight states (refer to Table 6.11). As the p-value is less than or equal to 0.05 for all states except for $m_{np,off}$, which is never encountered during a mission using adaptive policies, we can reject the null hypothesis that the classification outcomes have equivalent proportions. As a result, at the very least, the "trivial" observation functions cannot be considered an accurate representation of the process we aimed to model with a POMDP. However, anticipating these findings is precisely why we developed the risk-sensitive pipeline discussed in the preceding section.

Table 6.11: Results of the two-proportion Welch's t-test for the eight states. Bold values indicate a p-value less than 0.05.

State	t-test (t)	Degrees of Freedom	p-value
$\boldsymbol{m_{np,on}}$	$\mathbf{-2.34}$	$\mathbf{662.90}$	$\mathbf{0.02}$
$\boldsymbol{m_{p,on}}$	$\mathbf{5.05}$	$\mathbf{804.41}$	$\mathbf{< 0.001}$
$m_{np,off}$	-0.28	268.13	0.78
$\boldsymbol{m_{p,off}}$	$\mathbf{5.06}$	$\mathbf{122.87}$	$\mathbf{< 0.001}$
$\boldsymbol{a_{np,on}}$	$\mathbf{1.93}$	$\mathbf{478.27}$	$\mathbf{0.05}$
$\boldsymbol{a_{p,on}}$	$\mathbf{5.14}$	$\mathbf{585.54}$	$\mathbf{< 0.001}$
$\boldsymbol{a_{np,off}}$	$\mathbf{17.08}$	$\mathbf{485.54}$	$\mathbf{< 0.001}$
$\boldsymbol{a_{p,off}}$	$\mathbf{-9.28}$	$\mathbf{167.00}$	$\mathbf{< 0.001}$

Evaluation of the trivial POMDP model

To assess the POMDP trivial model, we calculate the belief for each mission, including those run with a policy other than the one obtained with EvC (using SARSOP). If the model is accurate, we

anticipate a high average belief of performance,

$$\overline{\beta} = \frac{1}{T}\sum_{t=1}^{T}\beta_t \tag{6.11}$$

in missions with high global scores, and vice versa. Thus, we compute the correlation between $\overline{\beta}$ and the global score for each mission. The Spearman's rank correlation reveals a significant positive correlation between $\overline{\beta}$ and the global mission score ($\rho(\overline{\beta}) = 0.325$, $N(\overline{\beta}) = 153$, $p(\overline{\beta})$-value < 0.001). In Figure 6.9, we illustrate the relationship between $\overline{\beta}$ and the mission score, along with a fitted (monotonic) fifth-order polynomial.

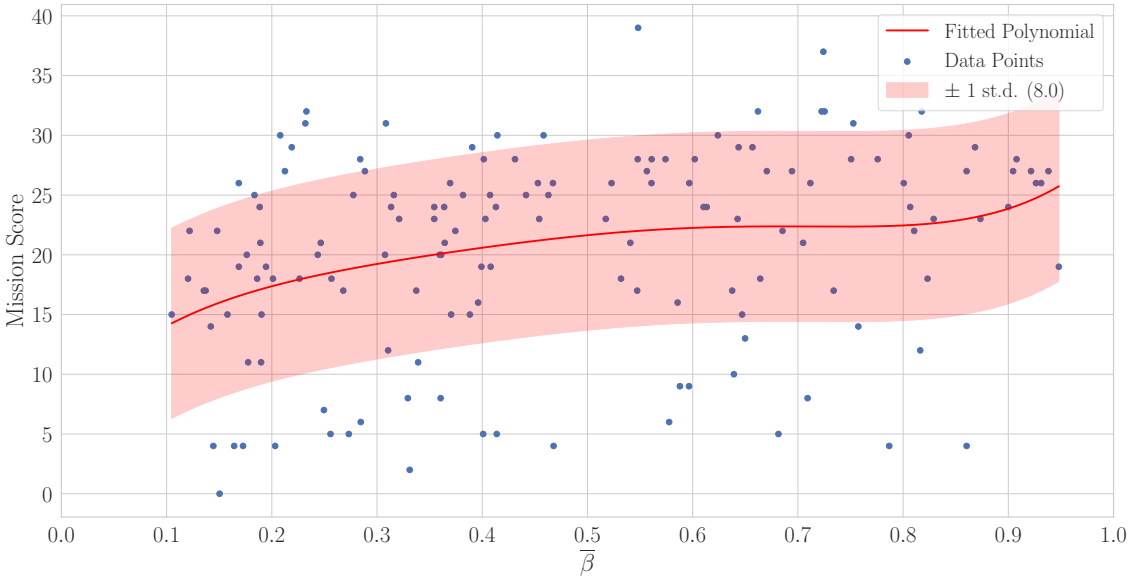


Figure 6.9: Scatter plot displaying the relationship between the average belief of performance during a mission (x-axis) and the mission global score (y-axis). A fifth-order (monotonic) polynomial is fitted to show the positive correlation (Spearman's $\rho = 0.325$) between $\overline{\beta}$ and the score. The shaded area represents plus or minus one standard deviation of the residuals of the fit (st.d. $= 8$).

We consider a positive correlation to be a promising indication since it highlights the trivial POMDP's alignment with the actual events despite the subpar classification results. As a result, we believe that transitioning to a POMDP is not unfounded, including the representation choice for modeling the POMDP, the criteria used to partition data for training the classifiers, and the method by which the classifiers were combined to construct and train an HMM using the Expectation-maximization algorithm. It is important to note that while the classification provides discrete labels with values of 0 or 1, the marginalized belief $\beta_t \in [0, 1]$ naturally accommodates the problem of determining when to switch modalities based on a specific threshold.

Since the Spearman's correlation coefficient exclusively tests for the presence of monotonic correlations without assessing the dependence between variablesmeaning it cannot detect non-linear, non-monotonic dependencewe examine the dependence between $\overline{\beta}$ and the score by computing the Randomized Dependence Coefficient (RDC) (Lopez-Paz, Hennig and Schölkopf, 2013). Utilizing the recommended hyperparameters from the reference, we obtain an RDC of 0.348. As the RDC is almost identical in magnitude to Spearman's coefficient, it is probable that there is no non-monotonic dependence between the variables.

Comparison of the policies

Performance Figure 6.10 displays the distribution of scores achieved with different policies across all participants and missions. The descriptive statistics for the results can be found in Table 6.12. We started by comparing the distribution of results obtained with the Random policy with the data in the starting batch $\mathcal{D}$ that was collected using a random policy (first line of Table 6.1). The Mann-Whitney U test was conducted to compare the two independent groups of scores. The test revealed a significant difference between the groups ($U = 686.0$, p-value < 0.05), leading us to reject the null hypothesis of no significant difference between the groups at the 5% significance level. Consequently, we cannot be "certain" that the sample we collected is a valid representation of the unknown underlying distribution. This finding underscores the challenge of our task, as we had to work with a limited dataset $\mathcal{D}$.

Among all the interaction policies, the Random policy performed the worst in terms of risk and performance metrics. The POMDP adaptive interaction policy resulted in the highest minimum mission score. The automatic control policy (with alarms) was the most performant with respect to the mean, the median, and also the most robust one taking into account the $VaR_{0.25}$. These results contradict our simulations, although in Table 6.8 it is reported that the automatic policy (without alarms) was the strategy with the highest first quartile. The second most performant and robust strategy was the POMDP (SARSOP, $\gamma = 0.98$) policy. The automatic and the POMDP policies also display the least variability. The MDP adaptive policy (full manual mode with alarms) was the one that scored the best with respect to the third quartile and the maximum value. The latter is qualitatively in agreement with the simulations for POMDP policy selection (see Table 6.8).

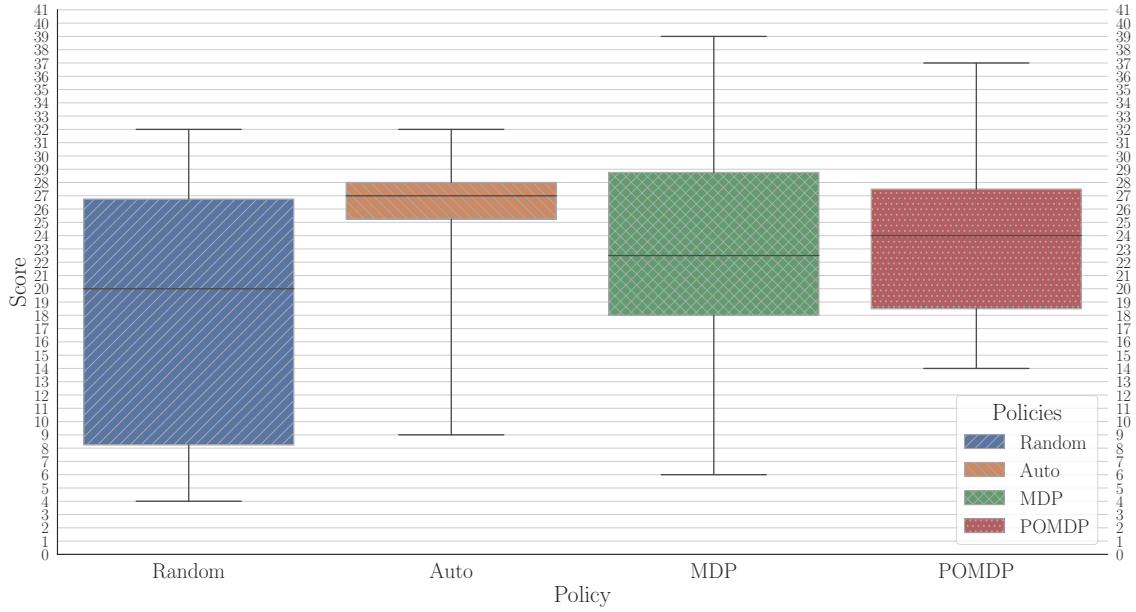


Figure 6.10: Boxplot with the distributions of scores by interaction control policy. Missions ran deploying the full automatic strategy have the highest first quartile, median, mean, and also the lowest variance.

A Friedman's test was conducted to analyze the data, revealing a significant difference among the policies ($\chi^2(3) = 13.07$, $p < 0.01$). Post-hoc analyses using the Wilcoxon signed-rank test with a Hommel correction indicated that the Random policy (Median = 20) yielded significantly lower scores compared to the Automatic policy (Median = 27, $p < 0.01$), the MDP policy (Median =

Table 6.12: Descriptive Statistics of Scores by interaction control policy. Best value per metric is displayed in **bold**.

	Score						
	mean	std	min	25%	Median	75%	max
Policy							
Random	17.9	9.6	4.0	8.3	20.0	26.8	32.0
Auto	**25.6**	**5.0**	9.0	**25.3**	**27.0**	28.0	32.0
MDP	22.7	7.6	6.0	18.0	22.5	**28.8**	**39.0**
POMDP	23.4	5.7	**14.0**	18.5	24.0	27.5	37.0

22.5, $p < 0.05$), and the POMDP policy (Median = 24, $p < 0.05$). No other statistically significant differences were observed, only trends between the Automatic and MDP policies ($p = 0.093$).

The results demonstrate that the adaptive controllers, both with and without accounting for partial observability, exhibit better performance and robustness than the data-collector policy (the random policy). Furthermore, the risk-aware policy derived from a hidden-states-based representation proves to be more robust than the one obtained through a fully observable representation. However, both adaptive policies were found to be less safe compared to the fully automatic one.

The MDP policy, which is equivalent to consistently selecting the manual robot control level (with alarms on), resulted in a score distribution with the largest high-score tail. This outcome confirms that, in this use case, skilled players can achieve higher scores manually than with the fully automatic policy. The POMDP adaptive policy ranked second in terms of the maximum score, suggesting that the idea of selecting the robot autonomy level based on an estimate of the team's performance has merit. This is supported by instances where the POMDP controller granted control of the robot to the human operator (*i.e.* putting it in manual mode) for extended periods of time. However, the effectiveness of the proposed method remains a first attempt and requires further refinement. Ideally, if the method could perfectly differentiate between high-performing and low-performing teams, we would expect a policy that is both more robust and more performant at every quantile of the global score distribution. Regrettably, achieving such a method would necessitate more accurate classifiers, more representative models, and beliefs β_t more strongly correlated with the score than those employed in these experiments and that were obtained offline using a fixed dataset containing a limited amount of demonstrations.

A crucial aspect of this study involves applying normalization to standardize the measurements of the Heart Rate and Heart Rate Variability, as outlined by Caroline P. C. Chanel et al. (2020). To achieve this, the protocol we employed recorded resting physiological baselines for participants before each mission, which were then used to normalize the HR and HRV values. However, we observed notable discrepancies in the recorded resting baselines for individual subjects across different missions. For example, a subject had a resting heart rate of approximately 70 bpm before a mission utilizing a Random policy. However, their resting heart rate recorded before a mission employing the POMDP adaptive policy could be as high as 100 bpm. The discrepancies were even more pronounced for HRV measurements at rest, which are inherently more challenging to measure. In this case, the disparity between the maximum and minimum measurements of HRV at rest per subject could be as much as eightfold. As previously discussed, physiological markers are among the most important features used by classifiers to identify high-performance time steps. Consequently, it is imperative to append the data processing and the quality of the experimental protocol to the list of critical aspects that can significantly impact the results of these experiments.

Additionally, it is important to recognize that the chosen adaptive policies, for both POMDP and MDP, were derived using methods that assumed an infinite horizon representation. While the

policy selection pipeline, based on the information from the initial dataset, would have favored these policies over the automatic policy with alarms, it is worth considering whether employing a finite horizon (60 time steps) could have resulted in different policies and outcomes.

Subjective feedbacks The average score on the KSS test *before* the experimental procedure was 3.1 (Median = 3.0, SD = 1.5), while the average score *after* the experiments was 3.3 (Median = 3.0, SD = 1.6). Recalling KSS survey is generally used to measure the subjective fatigue from participants, where a higher *after* KSS score can be associated with significant drowsiness. A Wilcoxon signed-rank test using the zsplit method was conducted to analyze the KSS questionnaire scores before and after the experimental procedure. The test indicated a sizable difference between the two sets of scores ($z = 143.5$, $p = 0.41$), but this difference was not statistically significant at the 0.05 level.

The NASA-TLX results for each scale and overall are depicted in Figure 6.11, with the fully-autonomous policy (*i.e.* Auto) being the least demanding across all scales. A Friedman's test was performed to analyze the overall rating, revealing a significant difference among the policies ($\chi^2(3) = 36.98$, $p < 0.001$). Post-hoc analysis using the Wilcoxon signed-rank test with Hommel correction showed a significant difference between the ratings obtained when the controller followed the fully-autonomous policy (Median = 40.0) and the other policies with $p < 0.001$ (Random, Median = 61.2; MDP, Median = 71.0; POMDP, Median = 65.3). These findings indicate that the fully-autonomous policy reduces the overall workload for the human operator; and, the impact on the subjective workload of adaptive policies remains inconclusive.

The results of the Fluency survey for each dimension and overall can be found in Figure 6.12. The fully-autonomous policy is perceived as providing the most fluent interaction, while the MDP adaptive control policy is considered the least fluent. A Friedman's test was conducted to analyze the overall rating, revealing a significant difference among the policies ($\chi^2(3) = 32.72$, $p < 0.001$). Post-hoc analysis using the Wilcoxon signed-rank test with Hommel correction showed a significant difference between the ratings obtained for every pair of policies ($p < 0.01$), except for the pair made of the Random (Median = 4.0) and the POMDP (Median = 4.5) policies ($p = 0.62$). The median of the overall adjusted rating for the fully-autonomous policy was 5.2, while for the MDP

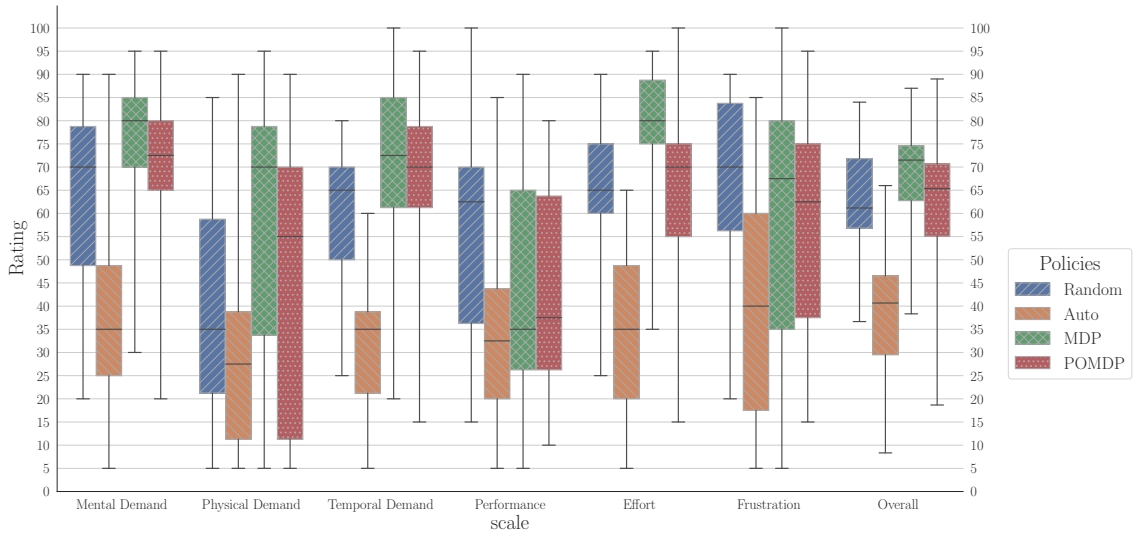


Figure 6.11: Boxplot with the distributions of NASA-TLX by interaction control policy and scale. The automatic policy is the least demanding across all scales.

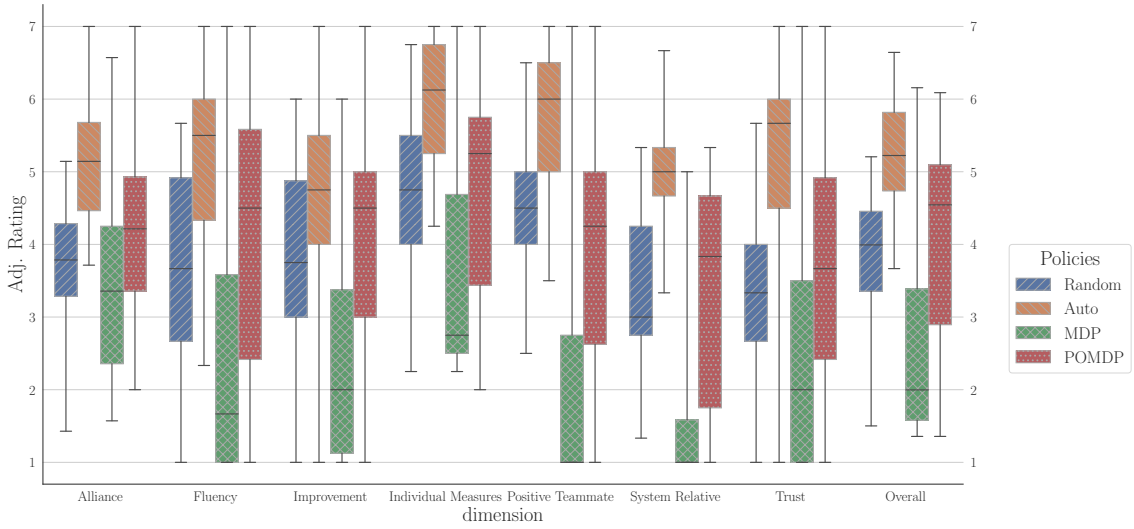
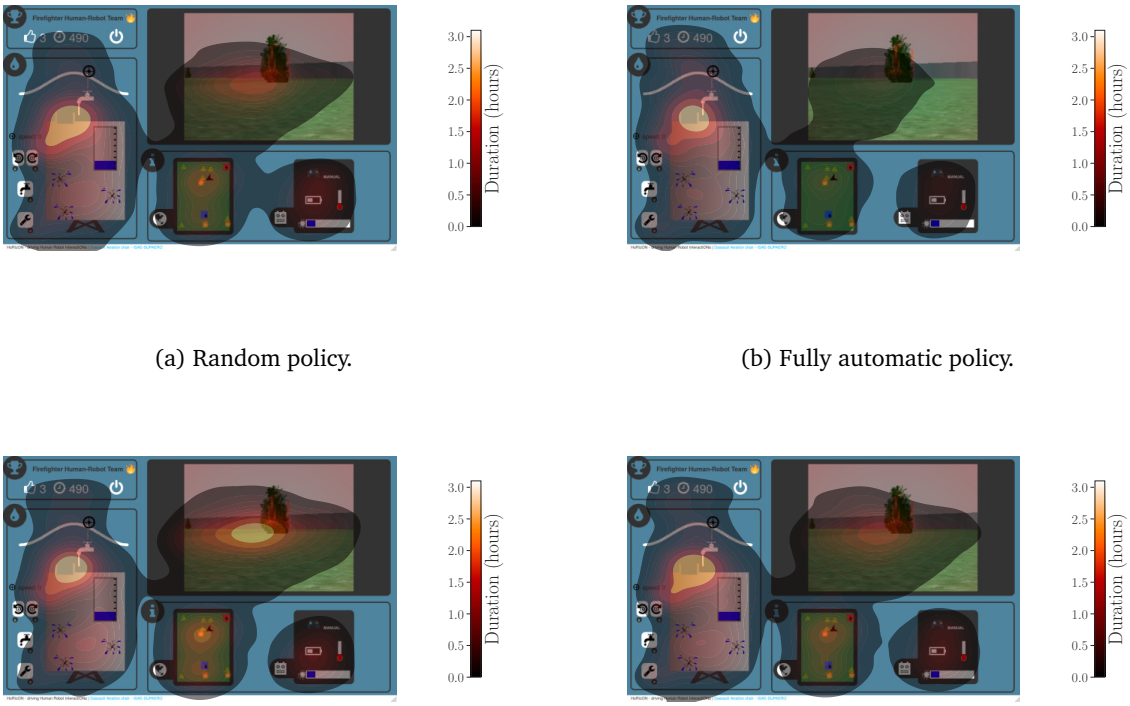


Figure 6.12: Boxplot with the distributions of the Fluency test by interaction control policy and dimension. The automatic policy is perceived as the one that leads to the most fluent interaction, while the MDP adaptive control policy is perceived as the least fluent.

policy, it was 2.0. In summary, while the fully-autonomous policy emerges as the most fluent and the MDP adaptive (manual) policy as the least, the findings on the random policy and adaptive POMDP policy warrant further investigation to better understand the potential of the latter in enhancing human-robot interaction experiences.

Physiological measurements Concerning eye-tracking measurements, the aggregate average spatial densities of the human operator's fixation durations on the GUI during missions with different interaction controllers were extracted from recorded data. These densities are displayed as heatmaps in Figure 6.13. It is worth noting that during the fully-autonomous mission, the human operator is primarily focused on the tank refill task, paying little attention to the robot navigation task, which is automatically managed (see Figure 6.13b). In contrast, when using the MDP policy, which maintains manual mode while providing alarms, the human operator's attention also extends to the navigation task (see Figure 6.13c).

Concerning ECG measurements *i.e.* HR and HRV, the distributions of the normalized average HR during each mission and the normalized HRV are shown in Figures 6.14 and 6.15, respectively. Recalling, these metrics are known to be impacted by workload: an increase in mental workload has been linked to an increase in HR and a decrease in HRV. The Friedman's test conducted to analyze the average normalized HR was significant ($\chi^2(3) = 10.95$, $p = 0.01$). A Wilcoxon signed-rank post hoc test with Hommel correction revealed a significant difference between the average normalized HR during missions following the fully-autonomous interaction policy (Median $= 0.1$) and the MDP policy (Median $= 3.6$) with $p < 0.05$. This result is in accordance with the subjective feedback from participants (*i.e.* NASA-TLX results) concerning the perceived workload. Other differences were not significant. The median of the average normalized heart rate during missions with the random policy was 2.4 and with the POMDP policy 2.1. The Friedman's test conducted to analyze the normalized HRV was not significant ($\chi^2(3) = 3.6$, $p = 0.31$) either.



(a) Random policy.

(b) Fully automatic policy.

(c) MDP policy.

(d) POMDP policy.

Figure 6.13: Heatmaps of the aggregate average density of fixation durations of the human operator on the Graphical User Interface (GUI) during missions with different interaction control policies.

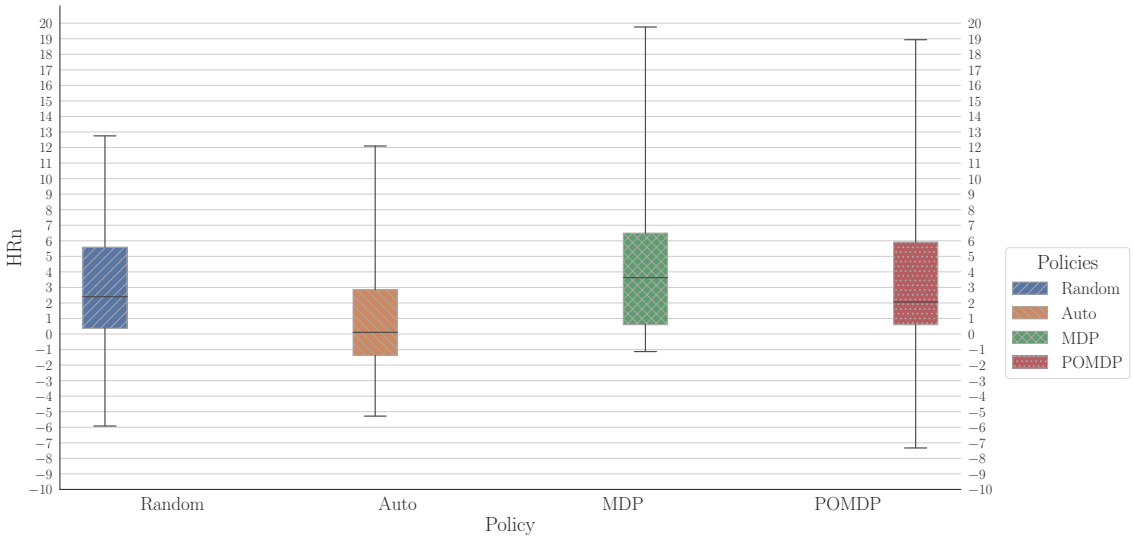


Figure 6.14: Boxplot with the distributions of the average normalized heart rate by interaction control policy.

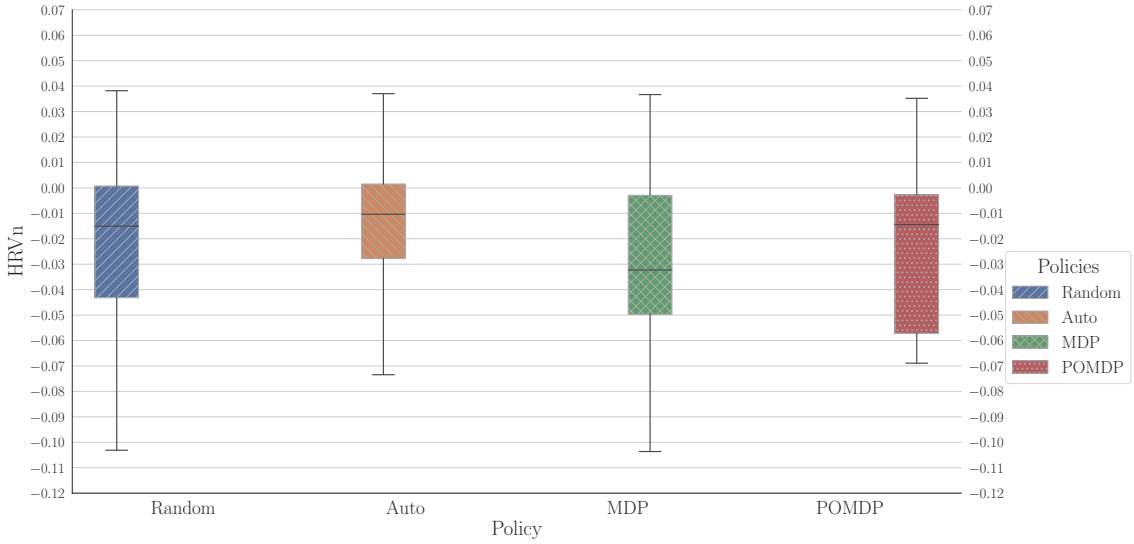


Figure 6.15: Boxplot with the distributions of the normalized heart rate variability by interaction control policy.

6.4 Conclusions

In this chapter, we have demonstrated the feasibility of tailoring a sequential decision-making problem under model uncertainty to guide the interaction of a mixed-initiative human-robot system using a (Partially Observable) Markov Decision Process. To this end, we developed adaptive policies for the interaction between a human operator and the navigation controller in the Firefighter Robot game, which also incorporated the human operator's physiological data.

We began by constructing a dataset of demonstrations that had been previously collected using a random interaction policy with multiple human participants. We then proposed a pipeline to represent the problem as both fully observable and partially observable Markov Decision Processes. Subsequently, we solved the MDP using various state-of-the-art risk-sensitive policy learning baselines and selected the most robust policy among them using our EvC approach, as discussed in Chapter 5. We also adapted EvC for POMDPs (despite a lack of theoretical guarantees on the convergence of the estimates) and selected the most robust policy among several candidates obtained with different discount factors.

We conducted experiments with human volunteers in lab facilities, demonstrating that our adaptive policies had a scientific basis and resulted in safer and more efficient policies compared to a random policy. The controllers estimated the team's performance state in real time and switched modalities accordingly. The POMDP-based adaptive policy outperformed the MDP-based policy in capturing real-time environmental dynamics. However, the adaptive policies were riskier and less effective than a fully-autonomous policy, which minimized human control and assigned greater autonomy to the machine. The interaction control policy with limited human involvement was the only one that demonstrated a reduction in perceived workload for the human operator and facilitated a more fluent interaction.

Our results are promising, despite the limitations of our pipelines and the frugality of the initial data set. The performance estimates obtained with the Trivial POMDP model were significantly correlated with the mission's final score, indicating that the model effectively captured the underlying dynamics. We addressed the well-known limitations of offline learning and the significant challenge of intersubject variability in human physiological markers and behaviors, although their impacts on the proposed method are still strong.

Potential avenues for future research involve: (i) minimizing intersubject variability in the initial dataset by selecting participants with comparable physiological baselines and training; (ii) devising more suitable methods to standardize physiological markers across different subjects; (iii) utilizing policies optimized through a finite horizon representation; (iv) forgoing direct interpretability in favor of a model-free approach to address potential problems stemming from inadequate hard-coded modeling or excessive dimensionality reduction; and (v) acknowledging the necessity for a larger dataset to facilitate data-driven inference of complex dynamics involving human behavior and physiological markers.

We envision that future mixed-initiative interaction control policies could strike a balance between a safe, adaptive, but not necessarily precise policy that performs well with a diverse range of human operators, and a precise, high-performing adaptive policy that is fine-tuned for a specific human operator or a narrow class of operators characterized by similar physiological baselines and training backgrounds. The first policy could be obtained offline and serve as a starting point to obtain the second one, whose fine-tuning could be performed online.

Key Takeaways

- Offline Reinforcement Learning shows promise for developing adaptive control policies for the interaction of mixed-initiative human-robot systems.
- Model-based approaches benefit from partially observable representations, but extracting them from data is challenging.
- Physiological computing in controlling human-robot interaction is significantly influenced by inter-subject variability.
- The experiments conducted in the Firefighter Robot Game demonstrate that adaptive policies exhibit superior robustness and performance when compared to the policy used for batch collection.
- In the Firefighter Robot Game, the experiments did not provide evidence that adaptive policies mitigate the cognitive workload of the human operator.

Does satisfying a robustness criterion suffice to earn the trust of a general audience and enable deployment of the resulting policy in real-world scenarios, potentially involving automated agents or controllers interacting with humans? Regrettably, the European Union Act for AI also stipulates that the approach must be explainable. In the subsequent chapter, we will introduce a method to elucidate the significance of the policy and the individual attributes of an automated agent by employing an environment simulator. This simulator may be oracle-based, as demonstrated in the next chapter, or learned from demonstrations. As the proposed method naturally extends to multi-agent contexts, such as human-robot interaction scenarios, it will be presented directly within the framework of Multi-Agent System (MAS).

Chapter 7

Towards the assessment of policy and attribute importances in Multi-Agent Systems

In this chapter, we introduce a method to quantitatively assess the global importance of an agent within a team while performing a cooperative task. We employ Shapley and Myerson analyses, previously discussed in Section 1.5, to investigate the contributions of both agent policies and individual attributes. Individual attributes refer to a set of characteristics that describe an agent's behavior in the environment, independent of the policy they employ. For instance, in a basketball game scenario, not only is the strategy adopted by a player crucial, but factors such as their speed, shooting accuracy, and other *individual attributes* also play a significant role.

We argue that the development of such a method is imperative for the real-world deployment and application of AI-based technologies, particularly when the Machine Learning model is trained solely on offline data. Compliance with robustness criteria used for maximizing or selecting a risk-aware policy may not be sufficient to grant legal authorization for deploying Offline Reinforcement Learning policies in contexts where humans can interact with one or more automated systems.

Simultaneously, this explainability approach could serve as an invaluable tool for strategists, decision-makers, and sports coaches. However, it is essential to acknowledge the challenge of determining an agent's importance in a cooperative task, where isolating an individual's performance from the rest of the team is difficult. Moreover, the relationship between an agent's role and their personal attributes is not always evident. It is crucial to consider that an agent possesses a policy (a set of rules, a function, or a black box dictating their actions) and individual features to tackle these challenges.

We propose a Hierarchical Knowledge Graph of agents' policies and features in a Multi-Agent System, enabling us to leverage dynamic programming and significantly reduce the computational complexity of Shapley analysis. Our approach is tested in a proof-of-concept environment using both hardcoded policies and policies obtained through Deep Reinforcement Learning, demonstrating valuable insights into an agent's importance within a team and the necessary attributes for effectively deploying their policy.

We posit that by constraining the coalitional gamecomprising agents' policies and attributeson a graph representative of the environment and enforcing an appropriate replacement rule, *i.e.* a rule to remove players from a coalition, for both attributes and policies (see Equation 1.24), the computational complexity of Shapley analysis can be reduced. This reduction can be achieved by

leveraging Myerson analysis and Dynamic Programming.

Consider a scenario where five basketball players (A, B, C, D, E) engage with one another during a match. We examine a collection of policies and individual attributes (as properties) associated with each agent. In this cooperative game coalition, both attributes and policies are regarded as players whose interaction is constrained by a graph (see Figure 7.1). When evaluating the Myerson characteristic function for a coalition that excludes D's policyreplacing it with a No-op policy for consistencythe computation involves applying the characteristic function to the original team minus agent D (and any attributes that were not removed). Therefore, the connectivity of the graph would allow to completely remove an agent during the computation for better scalability.

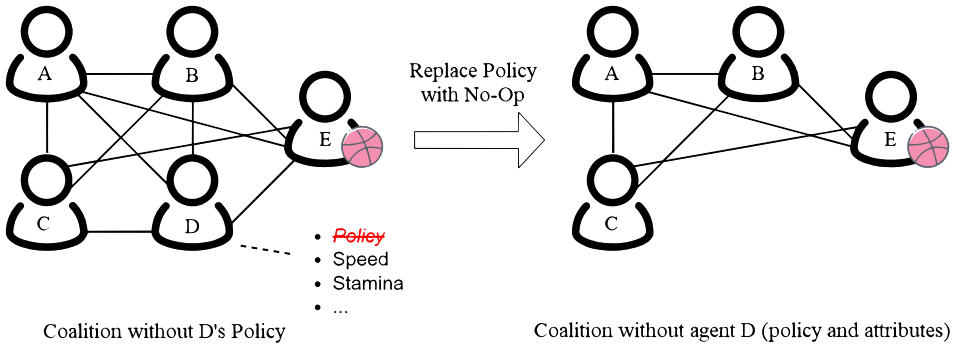
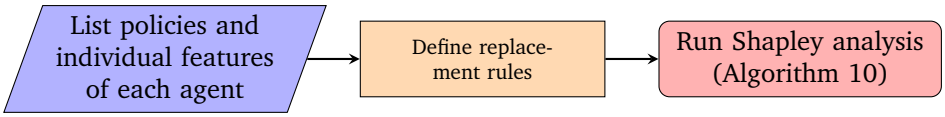


Figure 7.1: Feature and policy coalition formation example: Five basketball players (A, B, C, D, E) interact with each other during a match. We consider a set of policies and individual attributes (as properties) connected to each one of the agents. Both attributes and policies are considered players in the cooperative game coalition. When querying the characteristic function for a coalition without considering D's policy, since it would be replaced by the No-op policy for consistency, computing the characteristic function amounts to applying it to the same original team, without agent D (and without the whole set of attributes that were not removed).

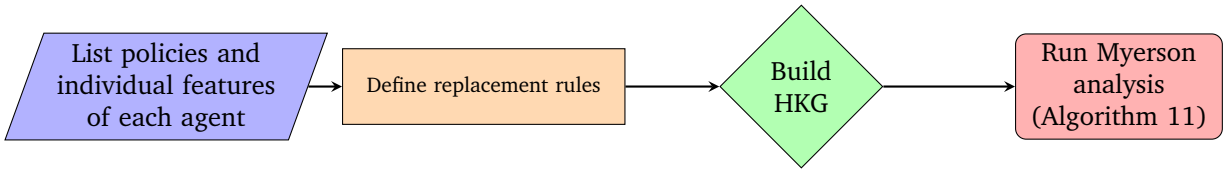
Research question We extend the research conducted on explainability in multi-agent systems and cooperative multi-agent RL discussed in Section 1.5. More specifically, since SHapley Additive exPlanations (SHAP) (S. M. Lundberg and S.-I. Lee, 2017) is not straightforwardly applicable to a RL Machine Learning, we follow the recipe proposed by Heuillet, Couthouis and Díaz-Rodríguez (2022) and compute the However, the study by Heuillet, Couthouis and Díaz-Rodríguez (ibid.) focused on agents' policies themselves, neglecting the influence of individual attributes on the implementation of a specific policy and the achievement of the objective. With this consideration in mind, our objective is to address the following research question:
Is it possible to explain both the importance of individual policies and the individual attributes of agents in a Multi-Agent System (MAS)?

Contribution We propose a methodology to explain a MAS by assessing the importance of factors that contribute to the attainment of a shared objective. The flowchart of this methodology is illustrated in Figure 7.2. The contributions of this chapter can be summarized in the following steps:

1. The proposal of aligning both agents' policies and individual attributes on equal footing with the aim to assess their importance with respect to the goal of the MAS using Shapley (see Definition 13) and Myerson analyses (see Definition 15) and a simulator (establishing suitable replacement rules). This corresponds to the first two steps in Figures 7.2a and 7.2b;



(a) Flowchart of the methodology with a simulator and Shapley analysis.



(b) Flowchart of the methodology with a simulator and Myerson analysis.

Figure 7.2: Flowcharts of the proposed methodologies.

2. The introduction of an expert-guided protocol to build a knowledge graph depicting the connectivity in the interactions between agents' policies and individual attributes in a Multi-Agent System. This graph will be defined as a Hierarchical Knowledge Graph (the third step in Figure 7.2b);

3. The empirical validation of (1) in a use-case Multi-Agent System using a simulator, showing that the computed Shapley (or Myerson) values are consistent with the system dynamics and that relevant features can be detected using this approach (the last step in Figures 7.2a and 7.2b);

4. Exploiting (2) for the empirical validation of (1), showing that the protocol to build a Hierarchical Knowledge Graph captures the connectivity of interactions in the system since the consequent Myerson analysis is statistically equivalent to the Shapley one, and on top of that less computationally expensive by an amount that depends on the case-specific graph structure;

5. Empirically showing that the operationalized technique can provide explanations regarding the accountability and roles of policies and attributes of agents trained with black box models such as Deep Reinforcement Learning architectures.

We now introduce the concept of a Hierarchical Knowledge Graph, which will be utilized for Myerson analysis, as well as a protocol for defining it within a specific multi-agent environment.

7.1 Hierarchical Knowledge Graph (HKG) for Multi-Agent Systems

A graph structure can be employed to represent the interactions between agents in a MAS. This practice is not uncommon in the context of Agent-Based Modeling, where the goal is to formally describe agent interactions (*e.g.* Kurve, Kotobi and Kesidis (2013), Rai, Minghao Wang and Hu (2015), Moya et al. (2017) and Robles et al. (2021)). However, our objective is to characterize not only the interactions between agents but also those between their individual features. Intuitively, individual features can be categorized into dynamic (changing over time) and static (remaining constant during a game run). Static attributes can be further classified into active, passive, and necessary attributes.

Necessary attributes are typically the most crucial in the hierarchy and are required for a policy to be deployed. For example, if there is a value for *Max Health Points* in a game, it becomes evident

that an agent requires this feature to have a value > 0 to act in the environment, as a dead agent cannot act. Thus, the *Maximum Health Points* are a necessary attribute since setting them to zero would render the player non-existent.

Static active attributes are expressed by the agent solely through direct actions in the system. For instance, in football, a player's shooting accuracy is demonstrated only when the player shoots. *Static passive* attributes can be expressed through interactions with the environment or other agent policies. For example, in football, resistance to pushes is exhibited only when another agent pushes the player.

In this manner, basic prior knowledge of the game rules is sufficient to create an HKG that includes agent-wise partitioned features and policies.

An HKG for a MAS is built as follows, having:

1. Fully connected interaction between static active attributes of the same agent;
2. Every static active attribute of the agent is connected to the agent policy, which is always a node in the HKG;
3. Agent's static necessary attributes are connected to the agent policy;
4. All static necessary attributes of the game (of all agents) are fully connected between them;
5. Passive attributes of an agent are fully connected between them;
6. Passive attributes of each agent are connected to the necessary attributes of the agent.

It is worth noting that the HKG is hierarchical uniquely in the sense that there is a "layer-like" grouping per agent attributes and policies (see Figure 7.3). Nevertheless, since both attributes of different agents and attributes of the same agent within the same group (*e.g.* Static Active Attributes) interact with each other, an HKG, despite its hierarchical structure, is a graph and not a tree. In Figure 7.3 an HKG for a Two Agents System is displayed. The former can be seen

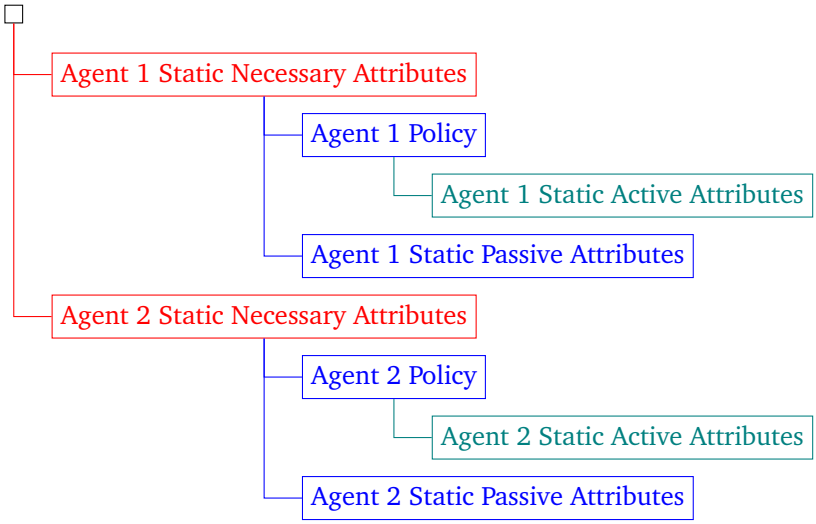


Figure 7.3: Example of Hierarchical Knowledge Graph for a Two Agent System. Same levels in the hierarchy are represented by the same color. Group of attributes with the same characteristics (same agent, active/passive) are fully connected between themselves). Edges do not connect boxes of the same level but only between parents and children. The top parent box is needed to represent interactions between static necessary attributes of different agents.

as a general template for feature-policy interaction in a MAS. However, even though it is easily

understandable, its validity is not general. One can imagine a system in which combinations of features interact with each other in the most complex and non-linear ways. If features and policies are weakly interacting, then the Myerson analysis protocol might provide a good approximation of the true interaction model. Otherwise, other means of including the prior information must be found to leverage the isolation of subgroups of features and policies that Myerson analysis with the HKG leverages to speed up the analysis.

Since the computational cost due to the number of operations needed to compute Myerson values scales with the number of connected components in the graph, the more interaction between features, the more edges in the graph. This means that the game is less constrained, hence there are more connected components, and thus the computational benefits of this approach (related to the ability to isolate connected components of features or sub-HKG) are restrained.

7.2 Requirements to compute Shapley and Myerson values for Multi-Agent Systems using a simulator

In this section, we outline the requirements to compute the Shapley and Myerson values in a MAS using a simulator for rollouts.

Replacement rule: In order to compute both Shapley and Myerson values using a simulator it

Algorithm 10: Exact Shapley Values Roll-out Computation with simulator f

Input: f simulator of the characteristic function v of the coalitional game $(\mathcal{C}, v)$, N number of simulations, ζ replacement rule
Output: $n = 1$ **to** N, $i = 1$ **to** $|\mathcal{C}|$ values $\phi_i^n(v)$ Shapley values

Initialization: $\phi_i^n(v) = 0$ with $n = 1$ **to** N
for $i \in \mathcal{C}$ **do**
  Generate the power set $\mathcal{P}(\mathcal{C} \backslash \{i\})$
  for $K \in \mathcal{P}(\mathcal{C} \backslash \{i\})$ **do**
    for $n = 1$ **to** N **do**
      $\phi_i^n(v) \leftarrow \phi_i^n(v) + \frac{|K|!(|\mathcal{C}|-|K|-1)!}{|\mathcal{C}|!} \left(f\left(\zeta(K \cup \{i\})\right) - f\left(\zeta(K)\right)\right)$
return ϕ

is mandatory to define a replacement rule for features and policies. Indeed, a simulator f of the MAS will likely require that all the agents are well-defined: they possess the full list of attributes. This means that $f : \{\sigma \text{ s.t. } |\sigma| = |\mathcal{C}|\} \to \mathbb{R}$. The simulator can only evaluate coalitions with the same number of players as the full number of features and policies to be analyzed.

How to deal with this hindrance? What is important is that the transferable utility coalitional game respects the properties of Definition 12 and hence that Equation 1.24 is valid.

We should then find a set of valid features/policies for the MAS $\Xi = \{\xi_1, \ldots, \xi_{|\mathcal{C}|}\}$ with $|\Xi| = |\mathcal{C}|$ such that $f(\Xi) = 0$. It is worth noting that ξ_i is not necessarily an element of $\mathcal{C}$, what is important is that f is well-defined when the feature/policy i has value ξ_i.

In this way, we can imagine that Ξ is equivalent to the void coalition $\varnothing$ with respect to the characteristic function v: $\Xi \overset{v}{\leftrightarrow} \varnothing$.

We put into a 1-to-1 relationship any $\sigma \in \mathcal{P}(\mathcal{C})$ such that $|\sigma| < |\mathcal{C}|$ and the coalition:

$$\zeta(\sigma) = \sigma \cup \{\xi_i \,|i \in \mathcal{C} \wedge i \notin \sigma\}. \tag{7.1}$$

Notice that $|\zeta(\sigma)| = |\mathcal{C}|$ and $f(\zeta(\sigma))$ is well-defined. We assume that $\sigma \overset{v}{\leftrightarrow} \zeta(\sigma)$.

Algorithms 3 and 4 can be adapted to take care of the simulator f and the replacement rule ζ. In particular Algorithm 10 (and 11) performs Shapley (and Myerson) analysis using a simulator f for a number of roll-outs (simulations) N given as input. Notice that the purpose of the simulator is providing the value of the characteristic function ν, otherwise not computable. Additionally, as stated before, the simulator can only work if all agents are well-defined (with a valid policy and a full list of attributes). In order to evaluate the characteristic function at coalitions without a subset of policies and/or attributes, line 6 is called in Algorithm 10, and lines 10, 14, and 16 are called in Algorithm 11. These lines perform the composition of functions $f \circ \zeta$: first, ζ replaces accordingly the removed features/policies in order to deal, successively, with a simulation f applied on a well-defined environment. Several simulation roll-outs N will be performed to explore the distribution of the results generated by the stochasticity of the environment.

Note that the results of this analysis provide intuition about the contribution of policies and attributes. However, even if an attribute is deemed more important than another, it is likely, but not granted, that the very same MAS where that single attribute is assuming a greater Shapley/Myerson value will be more performing. This is why we argue that this approach should not be used online during the learning phase of RL agents but only a posteriori, since credit assignment will only be fully given at the end of the task/game.

Algorithm 11: Exact Myerson Values Roll-out computation with simulator f

Input: $\mathcal{G}$ graph over the set of players $\mathcal{C}$ of the coalitional game $(\mathcal{C}, v)$, f simulator of the characteristic function v of the coalitional game $(\mathcal{C}, v)$, N number of simulations, ζ replacement rule

Output: $n = 1$ **to** N, $i = 1$ **to** $|\mathcal{C}|$ values $\phi_i^n(v)$ Myerson values

1 **Initialization:** $\phi_i^n(v) = 0$ with $n = 1$ **to** N, Coalitions = $\{\varnothing\}$
2 **for** $i \in \mathcal{C}$ **do**
3 Generate the power set $\mathcal{P}(\mathcal{C}\backslash\{i\})$
4 Decompose each $K \in \mathcal{P}(\mathcal{C}\backslash\{i\})$ to $\mu(K)$, the sets of connected nodes minimally covering the subgraph with vertices K
5 Decompose each $K \cup \{i\}$ to $\mu(K \cup \{i\})$, the sets of connected nodes minimally covering the subgraph with vertices $K \cup \{i\}$
6 **for** $K \in \mathcal{P}(\mathcal{C}\backslash\{i\})$ *ordered by increasing* $|K|$ **do**
7 **for** $n = 1$ **to** N **do**
8 **if** $\zeta(K) \notin$ *Coalitions* **then**
9 **for** $\sigma \in \mu(K) \wedge \zeta(\sigma) \notin$ *Coalitions* **do**
10 Coalitions$(\zeta(\sigma)) \leftarrow f(\zeta(\sigma))$
11 Coalitions$(\zeta(K)) \leftarrow \sum_{\sigma \in \mu(K)}$ Coalitions$(\zeta(\sigma))$
12 **if** $\zeta(K \cup \{i\}) \notin$ *Coalitions* **then**
13 **for** $\sigma \in \mu(K \cup \{i\}) \wedge \zeta(\sigma) \notin$ *Coalitions* **do**
14 Coalitions$(\zeta(\sigma)) \leftarrow f(\zeta(\sigma))$
15 Coalitions$(\zeta(K \cup \{i\})) \leftarrow \sum_{\sigma \in \mu(K \cup \{i\})}$ Coalitions$(\zeta(\sigma))$
16 $\phi_i^n(v) \leftarrow \phi_i^n(v) + \frac{|K|!(|\mathcal{C}|-|K|-1)!}{|\mathcal{C}|!}$ (Coalitions $(\zeta(K \cup \{i\})) -$ Coalitions $(\zeta(K))$)
17 **return** ϕ

7.3 Experimental evaluation

In Section 1.5.1, we defined a transferable utility coalitional game involving players. Here, we use the term "player" to denote actual team members in our game who cooperate to achieve a common

goal. We will define the transferable utility coalitional game over a set of players $\mathcal{C}$, which consists of features (individual attributes and policies).

We propose leveraging prior knowledge about the transferable utility coalitional game under consideration to construct a connected graph where each node represents an agent's individual attribute or policy. This approach expands upon Heuillet, Couthouis and Díaz-Rodríguez (2022) in two ways: 1) in that study, only agents' policies were analyzed and directly incorporated into the paradigm, and 2) we introduce the HKG as domain knowledge to be exploited.

We evaluate our approach in a game setting called *Arena*[13], which is inspired by World of Warcraft 3 vs 3 arena matches (Wikipedia contributors, 2023). To compute Shapley and Myerson values, we conduct rollout simulations using a game simulator. Due to the stochastic nature of the environment and occasionally the policies, multiple simulations are required. We validate the statistical significance of our results using the Mann-Whitney U test, a non-parametric statistical test specifically designed to compare the equality, in probability, of two populations.

7.3.1 *Arena* Game: RL environment description

Two teams, team *A* and team *B*, both of them made of a *Warrior*, a *Mage*, and a *Priest* fight each other. The common goal of a team is to defeat every enemy opponent in the least possible amount of moves. Given the different possible actions available to each agent, selfish strategies may easily lead to defeat. The teams perform their sequence of actions taking turns one at a time. At the beginning of each match, one team is chosen to start first with a random uniform probability. The agents in each team act abiding by the following order: 1) Warrior, 2) Mage, 3) Priest.

Victory condition: The Arena game ends when all agents in a team are dead or when $T = 1000$ rounds have passed. Let Ω be the set of all possible simulations. When team *A* wins, a simulation $\omega \in \Omega$ the game returns a result of $r(\omega) = +1$, when team *B* prevails the returned result amounts to $r(\omega) = -1$, if $T(\omega) = 1000$ rounds have passed and there is still not a winner then $r(\omega) = 0$.

The final score of a simulation is:

$$Score(\omega) = 100 \left(\frac{r(\omega)}{T(\omega)} + 1 \right) \tag{7.2}$$

where $T(\omega)$ is the total number of rounds needed to terminate the game. Notice that $Score \in [0, 200]$. If team *A* wins in one round, $Score = 200$, if it loses in one round, then $Score = 0$. Furthermore, $Score \to 100$ with $T \to 1000$. Intuitively, we will have $200 \geq Score > 100$ for simulations where team *A* won and $0 \leq Score < 100$ where it lost.

Description of agents' roles and attributes: Every agent executes a *policy*. We assume that the full list of individual attributes is known. We build an HKG for such a MAS with the individual attributes and policies following the protocol described in Section 7.1. The attributes of each agent are divided into Static Necessary Attributes, Static Active Attributes, and Dynamical Attributes:

1. Max Health Points [Static Necessary Attribute]: the maximum health points that an agent can possess;

2. Attack Power [Static Active Attribute]: the maximum damage that an agent can deal within one time step;

3. Healing Power [Static Active Attribute]: the maximum amount of health points that an agent can lend by healing himself or another one in one time step;

[13]The source code is openly accessible in the GitHub repository: https://github.com/giorgioangel/myersoncoop.

4. Control Chance [Static Active Attribute]: modulates the chance the *Mage* has to stop other agents from acting from one round (will be defined better later in Equation 7.4);

5. Current Health Points [Dynamical Attribute]: the health points of an agent at each time step.

The default value of these individual attributes is reported in Table 7.1. We do not consider Static Passive Attributes. Subsequently to the work in this chapter, we tried to add to the game a Defense attribute for each agent. Results were promising but the increase of features in the transferable utility coalitional game from 15 to 18, while still manageable for the Myerson method with HKG, made the problem already computationally too demanding for the Shapley approach in terms of our available computational resources.

Table 7.1: Default individual attributes (static and variable during the game) of each agent in the *Arena* cooperative multi-agent environment.

Static Necessary Attribute	***Value** [Range]*
MaxHealthPoints	100 [0-100]
Static Active Attributes	***Value** [Range]*
AttackPower	10 [0-20]
HealingPower	5 [0-100]
ControlChance	0.5 [0-0.5]
Dynamical Attribute	***Value** [Range]*
CurrentHealthPoints	100 [0-*MaxHealthPoints*]

Warrior

The Warrior can only *attack* an enemy agent. He damages the enemy by an amount equal to his (the Warrior's) *AttackPower*:

$$\mathit{tarHP}_{t+1} = \max\left(0, \mathit{tarHP}_t - \mathit{AttackPower}\right) \tag{7.3}$$

where *tarHP* represents the *CurrentHealthPoints* of the targeted enemy and t a time step. Any agent dies when his *CurrentHealthPoints* $= 0$.

Mage

A Mage can only *control* (put to sleep) an enemy agent. His chance P of controlling the enemy is equal to

$$P = \mathit{ControlChance}\left(1 + \frac{\mathit{AttackPower}}{20}\right). \tag{7.4}$$

When an enemy agent is put to sleep he cannot perform any action during the next turn.

Priest

A Priest can only *heal* a teammate. He heals the teammate by paying an amount equal to his *HealingPower*:

$$\mathit{tarHP}_{t+1} = min\left(\mathit{tarHP}_t + \mathit{HealingPower}, \mathit{tarMaxHP}\right) \tag{7.5}$$

where *tarMaxHP* are the *MaxHealthPoints* of the targeted agent and *HealingPower* is the one of the *Priest*.

7.3.2 Policies

Three different handcrafted policy types are enabled for all agents both in the hardcoded policy setting and, later on, in the setting with a policy learned with RL:

1. Random: with this policy, the target of the Warrior and the Mage are uniformly chosen between the alive enemies. The target of the Priest is uniformly chosen between the alive teammates.

2. Smart: the Warrior and Mage target the living enemies with the following priority list: 1) Priest, 2) Mage, 3) Warrior. The Priest always heals the living teammate with the least *CurrentHealthPoints*.

3. Do nothing (*No-Op*): While following this strategy, the agent does not perform any action.

4. Deep Reinforcement Learning (*RL*): Intending to show that the proposed approach can provide reasonable explanations of black-box Reinforcement Learning models based on Deep Neural Networks, we trained a Stable-Baselines3's A2C model (Raffin et al., 2021) where every agent can select the target of his action at every time step. Therefore, the *Warrior* will decide who to hit between the enemies, the *Mage* which enemy to control, and the *Priest* who to heal among his friends. Immediately then each agent will choose which action to perform on it. We trained the model in three phases, every time until convergence. In Phase 1 the enemy team was hardcoded to deploy a *No-op* policy, in Phase 2 the enemy team was hardcoded to act following the *Random* policy, and in Phase 3 the enemy team was hard-coded to follow a *Smart* policy. In this way, we could provide the A2C agent an adversary with increasing difficulty along the three phases. Moreover, the reward signal used to train the agent was not the sparse final score of Equation 7.2 but the difference between the total current health points of the teams. The said tricks let the training converge faster.

   With little surprise, the policy learned by the A2C model managed to overpower every hand-crafted policy with a 100% victory rate.

   We noticed that the A2C acts in the following way: the A2C trained agent learns to control all three agents in its team. The *A2C Warrior* and the *A2C Mage* both learn to attack and control the enemy's *Warrior*. Whenever this last one dies, the *A2C Mage* controls the enemy's *Priest* while the *A2C Warrior* attacks indiscriminately one between the remaining living enemies. During the whole match, the *A2C Priest* heals whoever of his team is taking damage.

7.3.3 Assumptions and goal of the evaluation

Hypothesis: we want to show that it is possible to explain both the importance of individual policies and the individual static attributes of agents in a MAS. With this in mind, we first build a transferable utility coalitional game whose players are both the policies and the individual features (separately taken), then we constrain the game onto an HKG: a graph structure for the MAS that is built following the protocol provided in Section 7.1. In order to test whether the approach is valid, we will compute both the Shapley Values (without the knowledge graph) and the Myerson values (exploiting the knowledge graph). In both cases, the characteristic function will be given by the $Score$ which is the output of a game simulator f (see Equation 7.2). Since the simulator requires

every agent to be well defined (with a valid policy and a full set of attributes), we have to apply the replacement rule ζ to coalitions before running it (see Equation 7.1). Hence, after applying ζ, every coalition will be legitimate in the sense that applying f (the simulator) to them will produce a result (a real number). However, in Shapley analysis, we will first use the replacement rule and then run the simulation (line 6 of Algorithm 10), while in Myerson analysis we will first check for the graph connectivity in order to exploit Property 5 in Definition 15 (lines 4 and 5 in Algorithm 11 decompose the coalition subgraph in connected parts and then the simulation is run only for the decomposed coalitions). We aim to show that:

1. by defining different yet correct replacement rules for policies and attributes the two can stand on the same footing with respect to these analyses;
2. the $N = 72$ results of Shapley and Myerson analysis come from the same distribution, and therefore that the HKG provides a good approximation to a latent structure of a MAS;
3. both the Shapley and the Myerson values are consistent with the rules of the game and the predicted contribution seems reasonable;
4. the number of computations needed to carry out Myerson analysis is lower than the Shapley one. Indeed computing exact Shapley values is $O(2^{|\mathcal{C}|})$ while computing Myerson values is $O(2^X)$ with $\mathcal{C}$ being defined in Definition 12 as the number of players and $X \leq |\mathcal{C}|$ being a constant proportional to the minimum number of connected nodes covering the graph $\mathcal{G}$ needed to form any coalition. In our particular case, $|\mathcal{C}| = 15$, $2^{15} = 32768$ while $X \approx 9.966$ and $2^{9.666} \approx 1000$.

All players share the same individual stats, as reported in Table 7.1.

Let us extend the formalization of the game $Score$ defined in Equation 7.2. Let $\mathcal{P}(\mathcal{C})$ be the power set of $\mathcal{C}$, the set of static attributes (Table 7.1) and policies of team *A*. We define Ω_σ with $\sigma \in \mathcal{P}(\mathcal{C})$ as the set of possible simulations for a specific coalition σ. When an individual attribute is not present in σ, then it is set to zero before starting the simulation (*e.g.* a coalition without *Warrior's AttackPower* means that in simulation the Warrior will start the simulation with *AttackPower* $= 0$). When a policy is not present, it is set to *Do nothing* (*No-Op*).

Notice that, if the policy and the features of *team B's Warrior* allow him to deal damage, then the score of every possible simulation $\omega_\varnothing \in \Omega_\varnothing$ where $\Omega_\varnothing$ is the set of simulations' outcomes attainable with an empty coalition is $Score(\omega_\varnothing) = 0$. Let us define an average score function over a set of N simulations $\Sigma : \mathbb{N}_+, \times\mathcal{P}(\mathcal{C}) \to [0, 200]$,

$$\Sigma(n, \sigma) = \frac{1}{n} \sum_{i=1}^{n} Score(\omega_{\sigma,i}). \tag{7.6}$$

where $\omega_{\sigma,i}$ is the outcome of the i-th simulation run with coalition σ.

Our goal is to compute the importance of the individual *static* attributes and policies of each member in team *A*. We will run $N = 72$ simulations with *team A* and *team B* playing all the possible combinations of policies in the set: {Random, Smart, No-Op, RL}.

We will compute these values with two different approaches: 1) naively calculating the Shapley values, 2) computing the Myerson values on a properly crafted HKG (Figure 7.4).

Replacement rule In the case of the *Arena* game,

$$\xi_{policy} = \textit{No-Op} \qquad \forall \textit{policy} \in \{\textit{Smart, Random, No-Op, RL}\},$$

$$\xi_j = 0 \qquad \forall j \in \mathcal{C} \wedge j \notin \{\textit{Smart, Random, No-Op, RL}\},$$

or in other words when j is an attribute and not a policy.

Shapley values In order to compute the Shapley values, we use as *characteristic function* v the $Score$ (Equation 7.2). We use a simulator of the game f and perform $N = 72$ different simulations. The values are computed using Algorithm 10 and the replacement rule described above.

Myerson values In order to compute the Myerson values, we first have to define a graph $\mathcal{G}$ that encompasses the relationship between the features. Using our prior knowledge about the game: if an agent has *MaxHealthPoints* $= 0$, he is already dead and then he is unable to act. If a policy is *No-Op* (do nothing) then all the other individual attributes besides *MaxHealthPoints* do not matter, and thus, by following the protocol provided in Section 7.1, we build $\mathcal{G}$ as the HKG shown in Figure 7.4. It is important to remember that the characteristic function v of a coalition σ of features or policies

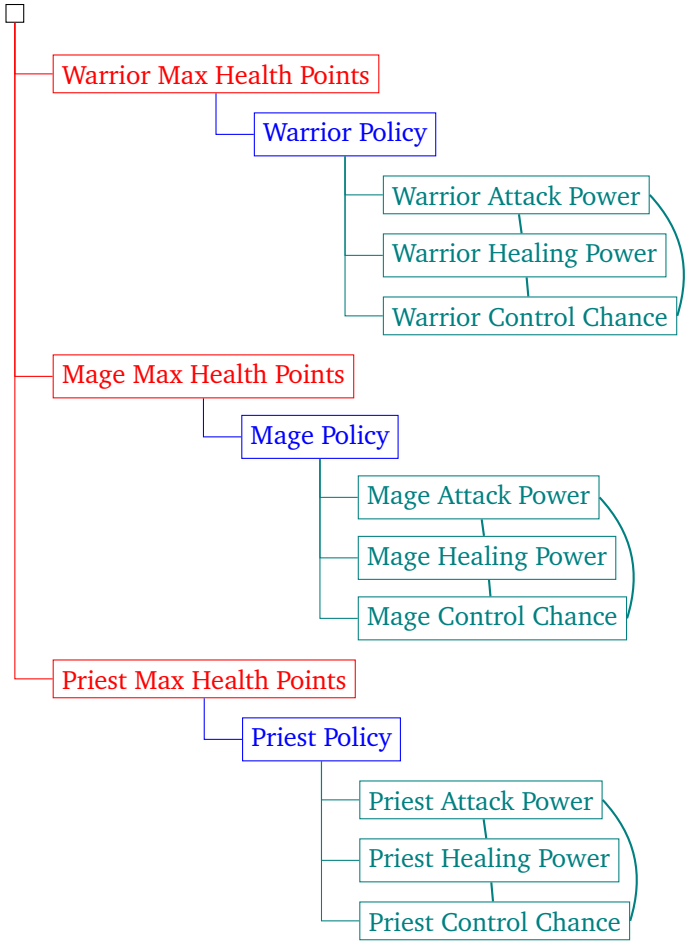


Figure 7.4: Hierarchical Knowledge Graph for the *Arena* game. Notice how the subgraph of Static Active Attributes is fully connected. In this game there are no Static Passive Attributes.

defined over the graph is the sum of the characteristic functions of the connected components of σ (Algorithm 4).

Hence, if for example a coalition σ is the whole $\mathcal{C}$ without the *Warrior's MaxHealthPoints*, we will have two connected components: a coalition σ_1 with all the attributes and policies of the *Mage* and the *Priest*, and σ_2, a coalition with just the policy and attributes of the *Warrior*. When performing a rollout for σ_2 all *MaxHealthPoints* are put to zero, hence all agents are dead, and

$Score$ is trivially zero. More generally, this is due to the structure of the HKG which allows one to completely ignore an agent, its policy, and its attributes when the high-level node in the graph hierarchy is not part of the considered coalition.

It is worth remembering that to compute the Shapley or the Myerson values we have to consider coalitions without some participants. This is not feasible when using a simulator. Therefore we had to define a rule ζ to replace attributes and policies with something else that for us is equivalent to a coalition without that given element. This rule was replacing an attribute value with zero and the policy with the *No-Op*.

If the feature *MaxHealthPoints* is available while the policy is replaced by *No-Op*, and any other subset of attributes are present, the agent will not act. In such a scenario, the coalition can be considered as one where the agent only is characterized by the *MaxHealthPoints* feature.

This greatly reduces the number of computations to be performed in order to obtain the Myerson values that were computed using Algorithm 11.

7.4 Results

The game's mean score Σ over $N = 72$ simulations is displayed in the results Table 7.2. In the first part of Table 7.2, we report the Shapley and the Myerson values obtained for all the features and policies when *team A* is playing a *Random* policy along with their computational times, in the second part of Table 7.2 *team A* is acting following the *Smart* policy, in the third part of Table 7.2 the *No-Op* policy and the last part of Table 7.2 the *RL* agent. The number of non-trivial evaluations of the characteristic function needed to assess the Shapley values in this example is 32768 while computing it for the Myerson values happens only 1000 times. It is worth noting that these values are specific to this scenario and environment, and the degree of reduction could change for a different environment that would be differently described by a different HKG.

The average total score Σ of every policy matching reflects the efficacy of the policy of *team A* against the one deployed by *team B*. When both teams are applying the same policy, the average score $\Sigma \approx 100$ means that, on average, the matches are ending in a draw. As expected, the weakest policy is *No-Op* ($\Sigma < 100$ for every policy matching), followed in increasing order of performance by *Random*, *Smart*, and *RL* ($\Sigma > 100$ in every case). Only for 1 value out of 240 ($240 = 5$ values times 3 agents times 16 policy combinations) the results obtained with the direct roll-out computation of the Myerson values are statistically different from the ones yielded by a direct estimate of the Shapley values according to the Mann-Whitney U test (*Priest's Control Chance* in *Random* vs *RL* match). This highlights that the designed *HKG* almost perfectly depicts the game's latent structure, allowing the Myerson approach to computing the very same contributions yielded by the Shapley analysis but in less time. Indeed, exploiting the knowledge of the graph structure made the approach using Myerson values from 19 up to 66 times faster.

Furtherly, we notice that some values are very close to zero (absolute value ≤ 0.02). We suspect these features actually do not contribute to the goal and therefore they are negligible. The stochasticity of the environment (and policies) and the limited number of simulations ($N = 72$) may yield results that are different from zero. Thus, we compare then the population of $N = 72$ simulations resulting Shapley and Myerson values, respectively, with a tuple of zeros using the Mann-Whitney U test to assess whether there is a statistically significant difference between those features from zero. We notice that in general the only relevant features (with $p < 0.001$) are:

1. Warrior Max Health Points;
2. Mage Max Health Points;

Table 7.2: Rollout computation of Shapley and Myerson values to explain the contribution of each static attribute and policy. Team A is playing the *Smart*, the *Random* and the No-Op policy. The time elapsed to compute the whole set of values for $N = 72$ simulations is reported below the label of each column. The displayed results are averaged over $N = 72$ different simulations. Statistical significance of a Mann-Whitney U test to check if the distribution of results is the same in probability as the one of a null contribution is reported after the mean value (when stars are present the distributions are different): * for $p < 0.05$, ** for $p < 0.01$, *** for $p < 0.001$. Statistical significance of a Mann-Whitney U test between the distribution of Shapley values and the one of Myerson values is displayed in **bold** when the distributions are different with $p < 0.05$.

	Random vs Random		Random vs Smart		Random vs No-Op		Random vs RL	
Feature	***Shapley***	***Myerson***	***Shapley***	***Myerson***	***Shapley***	***Myerson***	***Shapley***	***Myerson***
Total Score Σ	99.97	99.87	97.18	97.17	103.33	103.33	98.26	98.34
Comp. Time (s)	16796.07	392.50	34033.64	531.05	22642.04	760.45	40962.12	994.56
Agent: Warrior								
MaxHealthPoints	32.30***	32.32***	28.25***	28.25***	34.44***	34.44***	36.79***	36.81***
Policy	0.20***	0.19***	0.09***	0.09***	1.11***	1.11***	-0.21***	-0.21***
AttackPower	0.18***	0.16***	0.09***	0.09***	1.11***	1.11***	-0.22***	-0.20***
HealingPower	-0.02	-0.01	0.00	0.00	0.00	0.00	0.00	-0.01
ControlChance	0.02	-0.01	0.00	0.00	0.00	0.00	0.00	0.00
Agent: Mage								
MaxHealthPoints	32.55***	32.56***	32.81***	32.80***	33.33***	33.33***	30.49***	30.52***
Policy	0.37***	0.36***	0.09***	0.09***	0.00	0.00	0.30***	0.30***
AttackPower	0.19***	0.17***	0.03***	0.03***	0.00	0.00	0.09***	0.09***
HealingPower	0.00	-0.03	0.00	0.00	0.00	0.00	0.00	0.00
ControlChance	0.37***	0.35***	0.09***	0.09***	0.00	0.00	0.29***	0.28***
Agent: Priest								
MaxHealthPoints	32.76***	32.78***	35.70***	35.69***	33.33***	33.33***	30.20***	30.24***
Policy	0.54***	0.56***	0.02***	0.03***	0.00	0.00	0.26***	0.28***
AttackPower	-0.01	-0.04	0.00	-0.01	0.00	0.00	0.00*	0.01
HealingPower	0.55***	0.53***	0.02***	0.02***	0.00	0.00	0.25***	0.24***
ControlChance	0.00	-0.01	0.00	0.00	0.00	0.00	**0.01**	**-0.02**

	Smart vs Random		Smart vs Smart		Smart vs No-Op		Smart vs RL	
Feature	***Shapley***	***Myerson***	***Shapley***	***Myerson***	***Shapley***	***Myerson***	***Shapley***	***Myerson***
Total Score Σ	102.83	102.76	99.87	99.91	103.33	103.33	97.61	97.64
Comp. Time (s)	16284.31	385.34	34129.29	534.81	24156.22	868.38	39698.93	719.89
Agent: Warrior								
MaxHealthPoints	32.89***	32.90***	28.53***	28.53***	34.44***	34.44***	37.05***	37.05***
Policy	0.73***	0.72***	0.49***	0.43***	1.11***	1.11***	-0.18***	-0.17***
AttackPower	0.73***	0.73***	0.46***	0.41***	1.11***	1.11***	-0.18***	-0.17***
HealingPower	0.00	0.00	-0.01	0.04*	0.00	0.00	0.00	0.00
ControlChance	-0.01	0.00	0.04**	-0.02	0.00	0.00	0.00*	0.00
Agent: Mage								
MaxHealthPoints	32.90***	32.91***	32.90***	32.88***	33.33***	33.33***	29.99***	30.01***
Policy	0.13***	0.12***	0.34***	0.30***	0.00	0.00	0.00	0.00
AttackPower	0.04***	0.03***	0.19***	0.27***	0.00	0.00	0.00	0.00
HealingPower	-0.01	0.00	0.01	-0.01	0.00	0.00	0.00	0.00
ControlChance	0.13***	0.13***	0.29***	0.33***	0.00	0.00	0.00	0.00*
Agent: Priest								
MaxHealthPoints	33.36***	33.37***	36.05***	36.03***	33.33***	33.33***	30.30***	30.30***
Policy	0.95***	0.93***	0.28***	0.31***	0.00	0.00	0.31***	0.31***
AttackPower	0.00	0.00	-0.04	0.06	0.00	0.00	0.00	0.00
HealingPower	0.98***	0.91***	0.33***	0.31***	0.00	0.00	0.31***	0.31***
ControlChance	0.00	0.00	-0.01	0.02	0.00	0.00	0.00	0.00

	No-Op vs Random		No-Op vs Smart		No-Op vs No-Op		No-Op vs RL	
Feature	***Shapley***	***Myerson***	***Shapley***	***Myerson***	***Shapley***	***Myerson***	***Shapley***	***Myerson***
Total Score Σ	96.67	96.67	96.67	96.67	100.00	100.00	96.67	96.67
Comp. Time (s)	16350.52	364.33	33549.48	511.73	18503.63	435.27	44591.67	784.45
Agent: Warrior								
MaxHealthPoints	32.22***	32.22***	28.06***	28.06***	33.33***	33.33***	37.22***	37.22***
Policy	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
AttackPower	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
HealingPower	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
ControlChance	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
Agent: Mage								
MaxHealthPoints	32.22***	32.22***	33.06***	33.06***	33.33***	33.33***	29.72***	29.72***
Policy	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
AttackPower	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
HealingPower	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
ControlChance	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
Agent: Priest								
MaxHealthPoints	32.22***	32.22***	35.56***	35.56***	33.33***	33.33***	29.72***	29.72***
Policy	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
AttackPower	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
HealingPower	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
ControlChance	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00

	RL vs Random		RL vs Smart		RL vs No-Op		RL vs RL	
Feature	***Shapley***	***Myerson***	***Shapley***	***Myerson***	***Shapley***	***Myerson***	***Shapley***	***Myerson***
Total Score Σ	101.72	101.69	102.38	102.38	103.33	103.33	99.99	100.04
Comp. Time (s)	24629.82	1289.69	60915.64	1501.35	47774.24	1571.85	56939.91	1162.84
Agent: Warrior								
MaxHealthPoints	32.82***	32.81***	29.18***	29.18***	34.17***	34.17***	36.62***	36.63***
Policy	0.75***	0.75***	0.72***	0.71***	0.83***	0.83***	0.07***	0.08***
AttackPower	0.73***	0.75***	0.71***	0.72***	0.83***	0.83***	0.07***	0.07***
HealingPower	0.00	-0.02**	0.00	0.00	0.00	0.00	0.00	0.00
ControlChance	-0.01	0.00	0.00	0.00	0.00	0.00	0.01*	0.00
Agent: Mage								
MaxHealthPoints	33.02***	33.01***	34.18***	34.18***	34.17***	34.17***	30.64***	30.65***
Policy	0.53***	0.52***	0.45***	0.45***	0.00	0.00	0.90***	0.91***
AttackPower	0.16***	0.15***	0.09***	0.10***	0.00	0.00	0.30***	0.29***
HealingPower	0.00	-0.01	0.00	0.00	0.00	0.00	0.00	0.00
ControlChance	0.53***	0.53***	0.45***	0.45***	0.00	0.00	0.90***	0.90***
Agent: Priest								
MaxHealthPoints	32.60***	32.61***	36.55***	36.55***	33.33***	33.33***	30.24***	30.24***
Policy	0.29***	0.29***	0.02***	0.02***	0.00	0.00	0.11***	0.12***
AttackPower	0.00	0.01*	0.00	0.00	0.00	0.00	0.00	0.00
HealingPower	0.31***	0.28***	0.02***	0.02***	0.00	0.00	0.11***	0.12***
ControlChance	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00

3. Priest Max Health Points.

Then depending on the deployed policy, when no agent is following the *No-Op* also the following features are contributing:

1. Warrior Policy;
2. Warrior Attack Power;
3. Mage Policy;
4. Mage Attack Power;
5. Mage Control Chance;
6. Priest Policy;
7. Priest Healing Power.

This post-hoc result is consistent with the game dynamics defined in Equations 7.3-7.5.

Removing the non-relevant features according to the Mann-Whitney U test, we display in Figure 7.5 the Knowledge Graph with only the relevant attributes and policies.

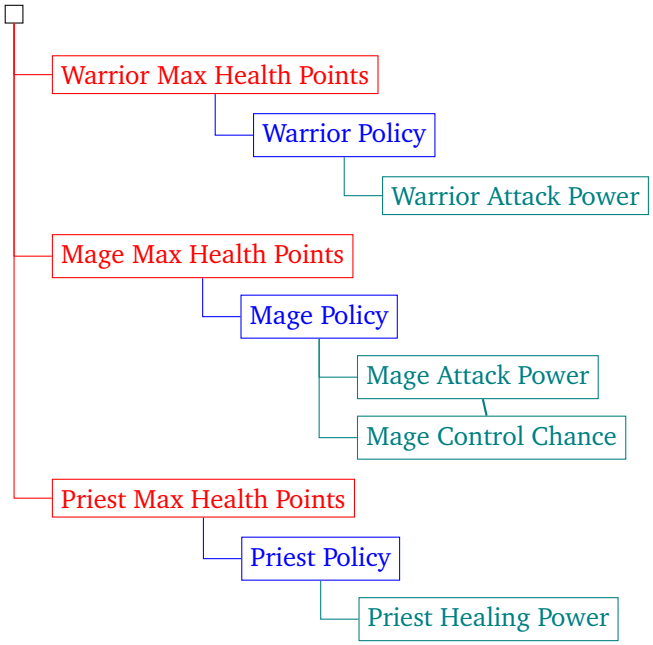


Figure 7.5: A-posteriori Hierarchical Knowledge Graph (HKG) for the relevant features of the *Arena* game. The non relevant features according to both the Shapley and the Myerson analysis were neglected since the Mann-Whitney U test between their associated Shapley and Myerson values and an atom of value equal to zero (a collection of samples all equal to zero) resulted in a p-value > 0.001 and therefore, the null hypothesis (H_0: the Shapley and Myerson value samples come from the same distribution of the atom) could not be discarded.

7.4.1 Per agent analysis

Warrior

Between the set of other individual attributes of the *Warrior* the *Attack Power* is the only feature substantially different from zero, in compliance with Equation 7.3, that dictates that the effects of *Warrior's policy* only depend on its *Attack Power*. The Warrior can have an impact on the game through his actions if and only if his *Attack Power* > 0.

Mage

As far as it concerns the other features of the *Mage*, the most relevant one is the *Control Chance* followed by the *Attack Power*. The fact that only the *Control Chance* and the *Attack Power* are relevant is in accordance with Equation 7.4. However, the actual importance order (the fact that the *Control Chance* contributes more to the goal than the *Attack Power*) is something that is hard to establish even knowing the game dynamics (Equation 7.4). Nevertheless, the results of the analysis make sense: with a *Control Chance* $\to 0$, $P \to 0$ (P is defined in 7.4) and the action of the Mage is non-effective, whether with *Attack Power* $\to 0$, $P \to$ *Control Chance*. Therefore the Mage can have an impact on the game through his actions if and only if his *Control Chance* > 0.

Priest

Finally, the only other relevant individual attribute of the *Priest* is the *Healing Power* as it is also indicated by Equation 7.5. Therefore the Priest can have an impact on the game through his actions if and only if his *Healing Power* > 0. Please notice that the paradigm is agnostic of the game dynamics and impressively obtained the correct results only by analyzing roll-out simulations.

7.4.2 Global qualitative analysis

The most important features are the *Max Health Points* of each agent. This suggests that, with respect to the adopted metric (score Σ, Equation 7.6) what matters the most in the *Arena* game is staying alive. Aside from this triviality, the second most contributing aspect of each agent is the policy since, obviously, an idle agent is not extensively contributing to the common goal.

Regarding the explanation of the different policies, we notice that the global importance of each policy and attribute depends on the team strategy and on the strategy adopted by the enemy team. The said hierarchy is established by the magnitude of the Shapley/Myerson values. In particular when *team B* is deploying the *No-Op* policy the most important agent of *team A* is the *Warrior* when it is following the *Random*, the *Smart* or the *RL* policy. These results are intuitive since the *Mage* stops enemy agents from acting, but if they are already non-acting the final outcome will be independent from its doing. The *Priest* heals teammates, but if the enemy *Warrior* is not dealing damage then it won't contribute to the victory. When also *team A* is deploying the *No-Op* strategy, and this result is independent of the enemy policy, then all the three agents are equally important (of course, in the team no-one is doing anything at all).

7.5 Explainability of a multi-agent RL model

Interpreting the part of Table 7.2 related to the *RL* policy, aside from *Max Health Points* that, as we have seen before, are always important, we noticed the following:

1. Agent trained with A2C (*RL*) *team A* vs Hardcoded *Random* Policy *team B*: the most important policy and feature are the *Warrior Policy* and the *Warrior Attack Power* followed by the *Mage Policy* and the *Mage Control Chance*. The *Priest Policy* and the *Priest Healing Power* come last, just before the *Mage Attack Power*;

2. Agent trained with A2C (*RL*) *team A* vs Hardcoded *Smart team B*: the most important policy and feature are the *Warrior Policy* and the *Warrior Attack Power* followed by the *Mage Policy*, the *Mage Control Chance* and the *Mage Attack Power*. Against this kind of team, the *Priest* is

not important since he will die more or less at the same time step than the enemy *Warrior* after having been constantly controlled by the enemy *Mage*;

3. Agent trained with A2C (*RL*) *team A* vs Hardcoded *No-op* policy *team B*: obviously only the *Warrior Policy* and the *Warrior Attack Power* are important since the enemy team is not acting and the time the Warrior spends to kill every single enemy is the only thing that matters;

4. Agent trained with A2C (*RL*) *team A* vs Agent trained with A2C (*RL*) *team B*: in this game the only important player is the *Mage* with his *Policy*, *Control Chance* and *Attack Power*. Indeed, we can imagine that since every Mage will control the enemy Warrior when he is alive, almost every match will end in a draw. Indeed, the total average score $\Sigma \approx 100$ (Table 7.2).

7.6 Conclusions

In this chapter, we proposed to exploit roll-out simulations and prior information about a transferable utility coalitional game to assess the importance of both individual attributes and policies of each agent using Myerson values. The first objective was to verify the feasibility of placing policies and attributes on equal footing. The second objective was to introduce an effective protocol for incorporating game knowledge for Multi-Agent System using Hierarchical Knowledge Graphs. As we demonstrated, the latter approach is particularly suitable for MAS since neglecting high-level features of an agent when considering a coalition can directly exclude the entire agent from the computation, thereby exploiting the compartmental nature of multi-agent systems. The third objective was to investigate whether Myerson enables computing attribution for both policies and features in a more time and compute cost-efficient manner, in the spirit of green explainable AI. The final objective was to demonstrate that this approach can reasonably provide explanations even when the policy is learned and deployed by a black-box Reinforcement Learning algorithm with Deep Neural Networks.

We tested the approach on a simple yet significant scenario that presented a plethora of non-trivial characteristics, such as nonlinear dynamics, cooperation, and diversified interaction. The experimental results showed that the presented approach not only assigns a value to the importance of each feature and policy but also correctly identifies the relevant features according to the agent role, game dynamics, and employed policy.

In particular, we observed that, despite small differences between the mean Shapley and Myerson values over $N = 72$ different simulations, the Myerson values computed using the HKG as prior knowledge could come from the same distribution as the Shapley values. This suggests that the proposed approach to building the HKG correctly isolated the game structure.

Our approach with Myerson values leverages trading off building a graph of the features and policy hierarchies to accelerate the computation of feature attribution in the post-training inference time, as a post-hoc explainable AI technique.

This approach opens up the possibility of explaining the importance of both cooperative policies and individual statistics of agents in any transferable utility coalitional game, from Cooperative AI and Multi-Agent Reinforcement Learning environments to the Offline RL evaluation of teams starting from a batch of pre-collected data, and the more generic field of eXplainable AI.

While leveraging the graph structure and Dynamic Programming reduces the computational complexity of importance assessment, the scalability of the approach is still significantly impacted by the number of agents and their individual attributes participating in the coalitional games. A potential future perspective to address the exponential scalability of the coalition number in games

with a high number of policies and attributes is to implement sampling approaches for computing Myerson values (Tarkowski et al., 2019), which can provide approximate solutions.

Key Takeaways

- Shapley and Myerson analyses can be used to study the contribution of both agent policies and attributes in a cooperative task.
- The computational complexity of Shapley analysis scales exponentially with the number of participants, so a Hierarchical Knowledge Graph is proposed to constrain the relationships between participants and allow for faster assessment of importances through Myerson analysis.
- The proposed approach is tested in a proof-of-case environment using both hard-coded policies and those obtained through Deep RL.
- The approach provides insight into the importance of an agent in a team as well as the attributes necessary for optimal policy deployment.

The content of this chapter gave rise to the following publication:

Giorgio Angelotti and Natalia Díaz-Rodríguez (2023). 'Towards a more efficient computation of individual attribute and policy contribution for post-hoc explanation of cooperative multi-agent systems using Myerson values'. In: *Knowledge-Based Systems* 260, p. 110189

☞See the publication ☞See the arXiv preprint

This chapter concludes the contributions of the present thesis. In the subsequent chapter, we will synthesize the findings related to the ensemble of addressed issues and postulate potential avenues for future research in Offline Reinforcement Learning.

Conclusions and perspectives

This doctoral dissertation has focused on learning and solving Markov Decision Processes (MDPs) and Partially Observable Markov Decision Processes (POMDPs) from demonstrations, with a particular emphasis on model uncertainty and risk awareness. We have highlighted the challenges that arise when the learning phase must rely on a fixed dataset, without the possibility of further interaction with the environment. In these scenarios, there is a high risk of overfitting the batch, resulting in policies that either do not generalize well or that the learned model does not accurately represent the dynamics of the environment. The latter can be particularly concerning when the learned policies are intended to be deployed in the real world, where safety and reliability are paramount. Moreover, as autonomous systems become more ubiquitous and begin to interact with humans on a daily basis, it is essential that their behavior is both robust and interpretable. These criteria present not only technical challenges but also legal ones, as new regulations such as the European Union AI Act (European Commission, 2020) will require compliance with robustness and explainability. With these challenges in mind, this doctoral thesis has considered both the technical aspects of learning robust policies from demonstrations and the broader implications of possible applications, such as Human-Robot Interaction, and explainability in Multi-Agent Systems.

In the **First Part** of this dissertation, we provided a comprehensive and critical description of the state-of-the-art. Notably, part of Chapter 2 gave rise to a paper presented at an international conference workshop (Angelotti, Drougard and Caroline Ponzoni Carvalho Chanel, 2020). In **Part Two**, we presented our contributions across several important subfields of offline learning for risk-sensitive planning, relying on the (PO)MDP mathematical framework. More specifically, those contributions encompass the following subtopics:

1. Data augmentation for more data-efficient learning (state-of-the-art discussed in Chapter 2, Section 2.4 - contribution in Chapter 4): We proposed a method to exploit expert knowledge to detect symmetries in an MDP. We showed that such an approach can be used to augment a batch of pre-collected trajectories, improving policy performance in offline learning. Despite its benefits, the technique has several limitations, including being affected by the density estimation technique and the state-action space preprocessing when dealing with continuous MDPs. Our work led to two presentations at an international conference as well as two publications in the conference proceedings (Angelotti, Drougard and Caroline Ponzoni Carvalho Chanel, 2022, 2023b);

2. Risk-sensitive offline hyperparameters and policy selection (state-of-the-art discussed in Chapter 2, Sections 2.2 and 2.5 - contribution in Chapter 5): We have demonstrated that the Bayesian formalism can be applied to the purpose of offline risk-sensitive policy selection, assuming a prior representing model uncertainty is obtainable from the demonstrations. The approach relies on a Monte Carlo estimation of the quantiles of the distribution of policy performance over model uncertainty. The method's scalability to larger MDPs may be limited due to the

parallel resolution of numerous models. The work led to a journal submission under review (Angelotti, Drougard and Caroline Ponzoni Carvalho Chanel, 2023a);

3. Development of methodologies to address adaptive interaction control policies in Human-Robot Interaction context, applicated to the Firefighter Robot game use-case (state-of-the-art in Chapter 3, contribution in Chapter 6): We successfully demonstrated how a sequential decision-making problem under model uncertainty can be tailored to guide human-robot interaction using MDPs and POMDPs, leveraging physiological computing. The obtained adaptive policies resulted in safer and more efficient missions compared to the one obtained by employing the data collector policy. The approach is affected by the quality and the frugality of the initial data set, and inter (human) subject variability. The work led to a journal article currently in preparation;

4. Explainability of Artificial Intelligence driven Multi-Agent Systems (state-of-the-art in Chapter 1.5 - contribution in Chapter 7): We proposed an approach to assess the importance of individual attributes and policies of each agent using Myerson values, Hierarchical Knowledge Graphs, and prior information about a transferable utility coalitional game. Our approach was tested on a proof-of-concept environment and showed that it correctly identified relevant features, as well as assigned to them reasonable importance values. Limitations include the scalability of the approach in terms of agents and attributes participating in the coalitional games. The results of this work were published as a full paper in an international journal (Angelotti and Díaz-Rodríguez, 2023).

Perspectives

Rather than focusing on the perspectives of the single-developed techniques, this section will discuss research perspectives within the field of Offline Reinforcement Learning.

Data augmentation with detected symmetries could be a useful tool to apply to small data sets and complex environments if the practitioner has an intuition about some possible symmetry and wants to validate it. While it is true that proposing a symmetric transformation for complex environments might be challenging, it is worth remembering that several possible industrial applications on top of the agenda of researchers include autonomous vehicles based on computer vision. In these environments, and with the proper precautions, it is reasonable to suppose the existence of left-right inversion symmetry. The extension of the approach proposed in Chapter 4 to states represented as images seems straightforward, but it is yet to be ascertained. Unfortunately, we could not apply symmetry detection to the case of the Firefighter Robot Game since we could not think of any valid symmetry for the coarse-grained representation we had opted to tackle the problem. However, we imagine that data augmentation with symmetric transitions could be exploited in the future in the context of HRI, especially to detect approximate symmetries, *e.g.* permutation between humans with a similar social role or features.

Following our achievements presented in Chapters 5 and 6, we firmly believe that risk-sensitive planning coupled to Reinforcement Learning (RL) is a promising way of implementing such sequential decision-making techniques in real-world contexts. Future perspectives include developments of less computationally costly algorithms not only for offline risk-aware policy selection but also for direct estimation of risk-sensitive policies, *e.g.* relying also on the Distributional RL framework. More specifically, only by estimating the distribution of returns given a policy, it is possible to

evaluate and compute policies that are aware of the risk. With this in mind, a promising research path could focus on Quantile Temporal-Difference algorithms (Rowland et al., 2023) for obtaining optimal control policies with respect to risk-sensitive metrics. Second, a non-overlapping, research branch could involve finding computationally viable discounting strategies that deal with planning under model uncertainty better than exponential discounting is also desirable. Note the proposed paths benefit more model-free rather than model-based approaches. Indeed, we suspect that in the future the most performant architectures will be model-free rather than model-based since a model-free approach is less demanding in terms of design approximations that are more often than not needed to develop a model-based approach, *e.g.* the choice of the representation and the definition of the reward function.

Throughout the manuscript, with the only exception for Chapter 6, we assumed that the reward function was known a priori and hence not learned from data. The motivation behind this idea lies in the fact that the reward depicts the desired behavior and the goal of the sequential decision-making problem. If the practitioner does not know which task she/he is about to solve, why is she/he solving it in the very first place? Notwithstanding, one could imagine that the scope of the practitioner is to find a risk-sensitive policy for a task that the agent that collected the batch wanted to solve, without possessing the information beforehand about which task was it. Learning the reward function from suboptimal demonstrations falls in the field of Inverse Reinforcement Learning (Adams, Cody and Beling, 2022). Truly agnostic and offline learning for model-based planning paradigms should infer the reward function alongside the dynamics in a complete data-driven fashion. In this context, model uncertainty would encompass both the uncertainty over the dynamics and the uncertainty about the reward. An important future perspective is therefore merging in a coherent way Risk-sensitive Model-Based Offline Reinforcement Learning and Inverse Reinforcement Learning. When the representation of the model is not known beforehand, a coarse-grained representation might transform a (PO)MDP that originally had a deterministic reward function to one with a stochastic reward function. This is exactly what happened while we were learning a model for the Firefighter Robot game use case. Eventually, we opted to use as the reward function the estimated expected value of the reward signal for each state. Nevertheless, when one does not focus anymore on finding the policy that maximizes the expected cumulative reward along a history, but rather on maximizing some risk-sensitive metrics defined on the whole distribution of returns, our choice might not be the right one. Therefore, another future perspective would involve the research on algorithms that solve for risk-sensitive policies of (PO)MDP models characterized by stochastic reward functions.

In our honest opinion, generalizable model-based approaches based on Deep Neural Networks could be a viable choice only if the right representation of the model is known a priori, and hence in learning scenarios that are not fully data-driven, or if the model is learned for a non straightforwardly interpretable sub representation of the sequential decision making problem. Concurrently, we believe that more focus is needed on performant and hyperparameter-stable Deep Neural Network-based architectures. Indeed, as we have seen in Section 4.4.3 and 4.5, such methods are enormously affected by the choice of hyperparameters. We presume that, in the future, the quest for performance and robustness will be at the expense of explainability. The AI-based methods will strive for the highest performance with respect to the optimized objectives and the technique will be explained a-posteriori.

Drawing upon our experience with the Firefighter Robot game, we recognize that generalizing across diverse human beings presents a formidable technical challenge. Despite preprocessing physiological data in a timely manner, such as normalizing Heart Rate in the Firefighter Robot

game, universal thresholds[14] in physiological or behavioral time series remain elusive. Our dataset revealed significant inter-subject variability in both physiological and behavioral time series. In our honest opinion, a promising venue could involve developing systems that abide by different policies: one that is less performant but that tries to be robust when the system is used and interacts with unknown users, and another that is obtained by fine-tuning the interaction with a single specific user or with a subclass of users that share similar features and training.

With this perspective, the pursuit of robust, performant, explainable, and interpretable offline reinforcement learning methods that comply with emerging regulations will become more competitive than ever. Explainability for planning and deep RL black-box baselines is a burgeoning field. We anticipate the extension of kernel-based approximations, like SHAP, to sequential decision-making problems. It is our belief that explainability baselines will largely rely on a posteriori analyses rather than methods intrinsic to the reinforcement learning pipeline.

Future work should further investigate explainable RL techniques with Myerson values in more complex environments, featuring a larger set of agents, a more extensive feature set, and more sophisticated policy learning models, such as those involving competition (Dhakal, Chiong, Chica et al., 2022) or graph games (D. L. Li and Shan, 2020). An intriguing future case study could involve post-hoc football statistical analysis using the Google Research Football environment for simulations (Kurach et al., 2020).

In the realm of Human-Robot Interaction, explanatory tools will be in high demand for obvious reasons. Given that methods like those discussed in this dissertation necessitate a simulator for each agent in the multi-agent interacting system, a potential research direction includes developing AI-based technologies to infer human intentions and behaviors. These methods would then be employed by a simulator. To advance in this area, during the last three and a half years of research at ISAE and ANITI, we supervised a research project by a (former) Master student on developing a behavior simulator for a metahuman operating the Firefighter Robot Game using Generative Adversarial Neural Networks (Penedo, 2022). Although the results were promising, the model has not yet achieved sufficient reliability and accuracy for inclusion in the experimental portion of this manuscript, and further work is needed. While providing promising results, the model did not yet reach a degree of reliability and accuracy to be used in the experimental part of this manuscript and supplementary work is coveted.

[14] Applicable to all possible human beings.

References

Books

Bellman, Richard (1957). *Dynamic Programming*. Rand Corporation research study. Princeton University Press.

Raiffa, Howard, Robert Schlaifer et al. (1961). *Applied Statistical Decision Theory*. Wiley New York.

Rubinstein, Reuven Y. (1981). *Simulation and the Monte Carlo Method*. John Wiley & Sons, Inc.

Ghallab, Malik, Dana Nau and Paolo Traverso (2004). *Automated Planning: Theory and Practice*. Elsevier.

Puterman, Martin L. (2005). *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. John Wiley & Sons.

Peters, Hans (2008). *Game Theory. A Multi-Leveled Approach*. Springer Berlin Heidelberg.

Lange, Sascha, Thomas Gabel and Martin Riedmiller (2012). *Batch Reinforcement Learning*. Springer, pp. 45–73.

Camacho, Eduardo F. and Carlos Bordons Alba (2013). *Model Predictive Control*. Springer Science & Business Media.

Aumann, Robert J. and Lloyd S. Shapley (2015). *Values of Non-Atomic Games*. Princeton University Press.

Föllmer, Hans and Alexander Schied (2016). *Stochastic Finance: An Introduction in Discrete Time*. Fourth. de Gruyter Graduate.

Sutton, Richard S. and Andrew G. Barto (2018). *Reinforcement Learning: An Introduction*. Adaptive Computation and Machine Learning series. MIT Press.

Bartneck, Christoph et al. (2020). *Human-Robot Interaction: An Introduction*. Cambridge University Press.

Molnar, Christoph (2020). *Interpretable Machine Learning*. Lulu. com.

Bellemare, Marc G., Will Dabney and Mark Rowland (2023). *Distributional Reinforcement Learning*. http://www.distributional-rl.org. MIT Press.

Doctoral dissertations

Drougard, Nicolas (2015). 'Exploiting imprecise information sources in sequential decision making problems under uncertainty'. PhD thesis. Toulouse, ISAE.

Journals

Samuelson, Paul A. (1937). 'A Note on Measurement of Utility'. In: *The review of Economic Studies* 4.2, pp. 155–161.

Boltyanskij, V. G., R. V. Gamkrelidze and L. S. Pontryagin (1956). 'On the theory of optimal processes'. Russian. In: *Doklady Akademii Nauk SSSR* 110, pp. 7–10.

Kalman, Rudolf Emil (1963). 'The Theory of Optimal Control and the Calculus of Variations'. In: *Mathematical Optimization Techniques* 309, p. 329.

Satia, Jay K. and Roy E. Lave (1973). 'Markovian Decision Processes with Uncertain Transition Probabilities'. In: *Operations Research* 21.3, pp. 728–740.

Smallwood, Richard D. and Edward J. Sondik (1973). 'The Optimal Control of Partially Observable Markov Processes over a Finite Horizon'. In: *Operations Research* 21.5, pp. 1071–1088.

Myerson, Roger B. (1977). 'Graphs and Cooperation in Games'. In: *Mathematics of Operations Research* 2.3, pp. 225–229.

Whitt, Ward (1978). 'Approximations of Dynamic Programs, I'. In: *Mathematics of Operations Research* 3.3, pp. 231–243.

Myerson, Roger B. (1980). 'Conference structures and fair allocation rules'. In: *International Journal of Game Theory* 9.3, pp. 169–182.

Papadimitriou, Christos H. and John N. Tsitsiklis (1987). 'The Complexity of Markov Decision Processes'. In: *Mathematics of Operations Research* 12.3, pp. 441–450.

Hart, Sandra G. and Lowell E. Staveland (1988). 'Development of NASA-TLX (Task Load Index): Results of Empirical and Theoretical Research'. In: *Advances in Psychology* 52, pp. 139–183.

Sutton, Richard S. (1988). 'Learning to predict by the methods of temporal differences'. In: *Machine Learning* 3.1, pp. 9–44.

Cybenko, George (1989). 'Approximation by superpositions of a sigmoidal function'. In: *Mathematics of Control, Signals and Systems* 2.4, pp. 303–314.

García, Carlos E., David M. Prett and Manfred Morari (1989). 'Model predictive control: Theory and practice. A survey'. In: *Automatica* 25.3, pp. 335–348.

Hornik, Kurt, Maxwell Stinchcombe and Halbert White (1989). 'Multilayer feedforward networks are universal approximators'. In: *Neural Networks* 2.5, pp. 359–366.

Åkerstedt, Torbjörn and Mats Gillberg (1990). 'Subjective and objective sleepiness in the active individual'. In: *International Journal of Neuroscience* 52.1-2, pp. 29–37.

Hornik, Kurt (1991). 'Approximation capabilities of multilayer feedforward networks'. In: *Neural Networks* 4.2, pp. 251–257.

Watkins, Chris and Peter Dayan (1992). 'Q-learning'. In: *Machine Learning* 8, pp. 279–292.

Williams, Ronald J. (1992). 'Simple statistical gradient-following algorithms for connectionist reinforcement learning'. In: *Machine Learning* 8.3, pp. 229–256.

Tsitsiklis, John N. (1994). 'Asynchronous Stochastic Approximation and Q-Learning'. In: *Machine Learning* 16.3, pp. 185–202.

White, Chelsea C. and Hany K. Eldeib (1994). 'Markov Decision Processes with Imprecise Transition Probabilities'. In: *Operations Research* 42.4, pp. 739–749.

Draeger, Andreas, Sebastian Engell and Horst Ranke (1995). 'Model predictive control using neural networks'. In: *IEEE Control Systems Magazine* 15.5, pp. 61–66.

Laibson, David (1997). 'Golden Eggs and Hyperbolic Discounting'. In: *The Quarterly Journal of Economics* 112.2, pp. 443–478.

Bilmes, J. et al. (1998). 'A gentle tutorial of the EM algorithm and its application to parameter estimation for Gaussian mixture and hidden Markov models'. In: *International Computer Science Institute* 4.510, p. 126.

Artzner, Philippe et al. (1999). 'Coherent measures of risk'. In: *Mathematical Finance* 9.3, pp. 203–228.

Hearst, Marti A. et al. (1999). 'Mixed-initiative interaction: Trends and controversies'. In: *IEEE Intelligent Systems* 14.5, pp. 14–23.

Pentland, Alex and Andrew Liu (1999). 'Modeling and Prediction of Human Behavior'. In: *Neural Computation* 11.1, pp. 229–242.

Givan, Robert, Sonia Leach and Thomas Dean (2000). 'Bounded-parameter Markov decision processes'. In: *Artificial Intelligence* 122.1-2, pp. 71–109.

Precup, Doina (2000). 'Eligibility Traces for Off-Policy Policy Evaluation'. In: *Computer Science Department Faculty Publication Series*, p. 80.

Munos, Rémi and Andrew Moore (2002). 'Variable Resolution Discretization in Optimal Control'. In: *Machine Learning* 49.2, pp. 291–323.

Rockafellar, R. Tyrrell and Stanislav Uryasev (2002). 'Conditional value-at-risk for general loss distributions'. In: *Journal of Banking & Finance* 26.7, pp. 1443–1471.

Shapiro, Alexander and Anton Kleywegt (2002). 'Minimax analysis of stochastic problems'. In: *Optimization Methods and Software* 17.3, pp. 523–542.

Givan, Robert, Thomas Dean and Matthew Greig (2003). 'Equivalence notions and model minimization in Markov decision processes'. In: *Artificial Intelligence* 147.1-2, pp. 163–223.

Green, Leonard and Joel Myerson (2004). 'A Discounting Framework for Choice With Delayed and Probabilistic Rewards'. In: *Psychological Bulletin* 130.5, p. 769.

Iyengar, Garud N. (2005). 'Robust Dynamic Programming'. In: *Mathematics of Operations Research* 30.2, pp. 257–280.

Nilim, Arnab and Laurent El Ghaoui (2005). 'Robust Control of Markov Decision Processes with Uncertain Transition Matrices'. In: *Operations Research* 53.5, pp. 780–798.

Geurts, Pierre, Damien Ernst and Louis Wehenkel (2006). 'Extremely randomized trees'. In: *Machine Learning* 63.1, pp. 3–42.

Smith, James E. and Robert L. Winkler (2006). 'The Optimizers Curse: Skepticism and Postdecision Surprise in Decision Analysis'. In: *Management Science* 52.3, pp. 311–322.

Goodrich, Michael A., Alan C. Schultz et al. (2008). 'Human–Robot Interaction: A Survey'. In: *Foundations and Trendső in Human–Computer Interaction* 1.3, pp. 203–275.

Munos, Rémi and Csaba Szepesvári (2008). 'Finite-Time Bounds for Fitted Value Iteration'. In: *Journal of Machine Learning Research* 9.5.

Nomura, Tatsuya et al. (2008). 'Prediction of Human Behavior in Human–Robot Interaction Using Psychological Scales for Anxiety and Negative Attitudes Toward Robots'. In: *IEEE Transactions on Robotics* 24.2, pp. 442–451.

Delage, Erick and Shie Mannor (2010). 'Percentile Optimization for Markov Decision Processes with Parameter Uncertainty'. In: *Operations Research* 58.1, pp. 203–213.

Esfahani, Ehsan Tarkesh and Vedantham Sundararajan (2011). 'Using Brain-Computer Interfaces to Detect Human Satisfaction in Human-Robot Interaction'. In: *International Journal of Humanoid Robotics* 8.01, pp. 87–101.

Doshi-Velez, Finale, Joelle Pineau and Nicholas Roy (2012). 'Reinforcement learning with limited reinforcement: Using Bayes risk for active learning in POMDPs'. In: *Artificial Intelligence* 187, pp. 115–132.

Mausam and Andrey Kolobov (2012). 'Planning with Markov Decision Processes: An AI Perspective'. In: *Synthesis Lectures on Artificial Intelligence and Machine Learning* 6.1, pp. 1–210.

Bellemare, M. G. et al. (2013). 'The Arcade Learning Environment: An Evaluation Platform for General Agents'. In: *Journal of Artificial Intelligence Research* 47, pp. 253–279.

Cummings, M. L. et al. (2013). 'Boredom and Distraction in Multiple Unmanned Vehicle Supervisory Control'. In: *Interacting with Computers* 25.1, pp. 34–47.

Kurve, Aditya, Khashayar Kotobi and George Kesidis (2013). 'An agent-based framework for performance modeling of an optimistic parallel discrete event simulator'. In: *Complex Adaptive Systems Modeling* 1.1, pp. 1–24.

Lattimore, Tor and Marcus Hutter (2014). 'General time consistent discounting'. In: *Theoretical Computer Science* 519, pp. 140–154.

Régis, Nicolas et al. (2014). 'Formal Detection of Attentional Tunneling in Human Operator–Automation Interactions'. In: *IEEE Transactions on Human-Machine Systems* 44.3, pp. 326–336.

Ricardo, Ricardo Chavarriaga, Aleksander Sobolewski and José del R. Millán (2014). 'Errare machinale est: the use of error-related potentials in brain-machine interfaces'. In: *Frontiers in Neuroscience* 8, p. 208.

Donath, Diana and Axel Schulte (2015). 'Behavior Based Task and High Workload Determination of Pilots Guiding Multiple UAVs'. In: *Procedia Manufacturing* 3, pp. 990–997.

Mnih, Volodymyr, Koray Kavukcuoglu et al. (2015). 'Human-level control through deep reinforcement learning'. In: *Nature* 518.7540, pp. 529–533.

Carneiro, João et al. (2016). 'Intelligent negotiation model for ubiquitous group decision scenarios'. In: *Frontiers of Information Technology & Electronic Engineering* 17.4, pp. 296–308.

Sheridan, Thomas B. (2016). 'Human–Robot Interaction. Status and Challenges'. In: *Human Factors* 58.4, pp. 525–532.

Silver, David, Aja Huang et al. (2016). 'Mastering the game of Go with deep neural networks and tree search'. In: *Nature* 529.7587, pp. 484–489.

Chen, Chih-Ming, Jung-Ying Wang and Chih-Ming Yu (2017). 'Assessing the attention levels of students by using a novel attention aware system based on brainwave signals'. In: *British Journal of Educational Technology* 48.2, pp. 348–369.

Moya, Ignacio et al. (2017). 'An agent-based model for understanding the influence of the 11-M terrorist attacks on the 2004 Spanish elections'. In: *Knowledge-Based Systems* 123, pp. 200–216.

Silver, David, Julian Schrittwieser et al. (2017). 'Mastering the game of Go without human knowledge'. In: *Nature* 550.7676, pp. 354–359.

Akkaladevi, Sharath Chandra et al. (2018). 'Toward an Interactive Reinforcement Based Learning Framework for Human Robot Collaborative Assembly Processes'. In: *Frontiers in Robotics and AI* 5, p. 126.

Briggs, K. and F. M. Ying (2018). 'How to estimate quantiles easily and reliably'. In: *Mathematics Today* 2018.February, pp. 26–29.

Peysakhovich, Vsevolod et al. (2018). 'The Neuroergonomics of Aircraft Cockpits: The Four Stages of Eye-Tracking Integration to Enhance Flight Safety'. In: *Safety* 4.1, p. 8.

Schulz, Ruth, Philipp Kratzer and Marc Toussaint (2018). 'Preferred Interaction Styles for Human-Robot Collaboration Vary Over Tasks With Different Action Types'. In: *Frontiers in Neurorobotics* 12, p. 36.

Silver, David, Thomas Hubert et al. (2018). 'A general reinforcement learning algorithm that masters chess, shogi, and Go through self-play'. In: *Science* 362.6419, pp. 1140–1144.

François-Lavet, Vincent et al. (2019). 'On Overfitting and Asymptotic Bias in Batch Reinforcement Learning with Partial Observability'. In: *Journal of Artificial Intelligence Research* 65, pp. 1–30.

Hoffman, Guy (2019). 'Evaluating Fluency in HumanRobot Collaboration'. In: *IEEE Transactions on Human-Machine Systems* 49.3, pp. 209–218.

Jain, Siddarth and Brenna Argall (2019). 'Probabilistic Human Intent Recognition for Shared Autonomy in Assistive Robotics'. In: *ACM Transactions on Human-Robot Interaction (THRI)* 9.1, pp. 1–23.

Mostafa, S., M. S. Ahmad and A. Mustapha (2019). 'Adjustable autonomy: a systematic literature review'. In: *Artificial Intelligence Review* 51.2, pp. 149–186.

Schilling, Malte et al. (2019). 'Shared AutonomyLearning of Joint Action and Human-Robot Collaboration'. In: *Frontiers in Neurorobotics* 13, p. 16.

Zheng, Wei and Hai Lin (2019). 'Vector autoregressive POMDP model learning and planning for human–robot collaboration'. In: *IEEE Control Systems Letters* 3.3, pp. 775–780.

Arrieta, Alejandro Barredo et al. (2020). 'Explainable Artificial Intelligence (XAI): Concepts, taxonomies, opportunities and challenges toward responsible AI'. In: *Information Fusion* 58, pp. 82–115.

Bellemare, Marc G., Salvatore Candido et al. (2020). 'Autonomous navigation of stratospheric balloons using reinforcement learning'. In: *Nature* 588.7836, pp. 77–82.

Chanel, Caroline P. C. et al. (2020). 'Towards Mixed-Initiative HumanRobot Interaction: Assessment of Discriminative Physiological and Behavioral Features for Performance Prediction'. In: *Sensors* 20.1, p. 296.

Dehais, Frédéric et al. (2020). 'A Neuroergonomics Approach to Mental Workload, Engagement and Human Performance'. In: *Frontiers in Neuroscience* 14, p. 268.

Dzedzickis, Andrius, Artras Kaklauskas and Vytautas Bucinskas (2020). 'Human Emotion Recognition: Review of Sensors and Methods'. In: *Sensors* 20.3, p. 592.

Kobyzev, Ivan, Simon Prince and Marcus Brubaker (2020). 'Normalizing Flows: An Introduction and Review of Current Methods'. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence*.

Li, Daniel Li and Erfang Shan (2020). 'The Myerson value for directed graph games'. In: *Operations Research Letters* 48.2, pp. 142–146.

Lundberg, Scott M., Gabriel Erion et al. (2020). 'From local explanations to global understanding with explainable AI for trees'. In: *Nature Machine Intelligence* 2.1, pp. 56–67.

Marot, Antoine et al. (2020). 'Learning to run a power network challenge for training topology controllers'. In: *Electric Power Systems Research* 189, p. 106635.

Roy, Raphaëlle N. et al. (2020). 'How Can Physiological Computing Benefit Human-Robot Interaction?' In: *Robotics* 9.4, p. 100.

Schrittwieser, Julian et al. (2020). 'Mastering Atari, Go, chess and shogi by planning with a learned model'. In: *Nature* 588.7839, pp. 604–609.

Zheng, Wei and Hai Lin (2020). 'Point-Based Value Iteration for VAR-POMDPs'. In: *IEEE Control Systems Letters* 6, pp. 7–12.

Heuillet, Alexandre, Fabien Couthouis and Natalia Díaz-Rodríguez (2021). 'Explainability in Deep Reinforcement Learning'. In: *Knowledge-Based Systems* 214, p. 106685.

Lee, Michael S., Henny Admoni and Reid Simmons (2021). 'Machine Teaching for Human Inverse Reinforcement Learning'. In: *Frontiers in Robotics and AI* 8, p. 188.

Raffin, Antonin et al. (2021). 'Stable-Baselines3: Reliable Reinforcement Learning Implementations'. In: *Journal of Machine Learning Research* 22.268, pp. 1–8.

Robles, Juan Francisco et al. (2021). 'Multimodal Evolutionary Algorithms for Easing the Complexity of Agent-Based Model Calibration'. In: *Journal of Artificial Societies and Social Simulation* 24.3.

Singh, Gaganpreet, Caroline P. C. Chanel and Raphaëlle N. Roy (2021). 'Mental Workload Estimation Based on Physiological Features for Pilot-UAV Teaming Applications'. In: *Frontiers in Human Neuroscience* 15.

Adams, Stephen C., Tyler Cody and P. Beling (2022). 'A survey of Inverse Reinforcement Learning'. In: *Artificial Intelligence Review* 55.6, pp. 4307–4346.

Dhakal, Sandeep, Raymond Chiong, Manuel Chica et al. (2022). 'Evolution of cooperation and trust in an N-player social dilemma game with tags for migration decisions'. In: *Royal Society Open Science*.

Heuillet, Alexandre, Fabien Couthouis and Natalia Díaz-Rodríguez (2022). 'Collective eXplainable AI: Explaining Cooperative Strategies and Agent Contribution in Multiagent Reinforcement Learning With Shapley Values'. In: *IEEE Computational Intelligence Magazine* 17.1, pp. 59–71.

Kurniawati, Hanna (2022). 'Partially Observable Markov Decision Processes and Robotics'. In: *Annual Review of Control, Robotics, and Autonomous Systems* 5.1, pp. 253–277.

Metulini, Rodolfo and Giorgio Gnecco (2022). 'Measuring players importance in basketball using the generalized Shapley value'. In: *Annals of Operations Research*, pp. 1–25.

Swiechowski, Maciej et al. (2022). 'Monte Carlo Tree Search: A Review of Recent Modifications and Applications'. In: *Artificial Intelligence Review*, pp. 1–66.

Angelotti, Giorgio and Natalia Díaz-Rodríguez (2023). 'Towards a more efficient computation of individual attribute and policy contribution for post-hoc explanation of cooperative multi-agent systems using Myerson values'. In: *Knowledge-Based Systems* 260, p. 110189.

Lauri, Mikko, David Hsu and Joni Pajarinen (2023). 'Partially Observable Markov Decision Processes in Robotics: A Survey'. In: *IEEE Transactions on Robotics* 39.1, pp. 21–40.

Conference proceedings

Sutton, Richard S. (1990). 'Integrated Architectures for Learning, Planning, and Reacting Based on Approximating Dynamic Programming'. In: *Proceedings of the Seventh International Conference on Machine Learning*. Elsevier, pp. 216–224.

Bertsekas, Dimitri P. and John N. Tsitsiklis (1995). 'Neuro-dynamic programming: an overview'. In: *Proceedings of 1995 34th IEEE conference on decision and control*. Vol. 1. IEEE, pp. 560–564.

Atkeson, Christopher G. and Juan Carlos Santamaria (1997). 'A comparison of direct and model-based reinforcement learning'. In: *International Conference on Robotics and Automation*. Vol. 4. IEEE, pp. 3557–3564.

Dean, Thomas and Robert Givan (1997). 'Model Minimization in Markov Decision Processes'. In: *AAAI/IAAI*, pp. 106–111.

Cassandra, Anthony R. (1998). 'A survey of POMDP applications'. In: *Working notes of AAAI 1998 Fall Symposium on Planning with Partially Observable Markov Decision Processes*. Vol. 1724.

Sozou, P. D. (1998). 'On hyperbolic discounting and uncertain hazard rates'. In: *Proceedings of the Royal Society of London. Series B: Biological Sciences*. Vol. 265. 1409. The Royal Society, pp. 2015–2020.

Konda, Vijay and John Tsitsiklis (1999). 'Actor-Critic Algorithms'. In: *Advances in Neural Information Processing Systems*. Vol. 12.

Madani, Omid, Steve Hanks and Anne Condon (1999). 'On the Undecidability of Probabilistic Planning and Infinite-Horizon Partially Observable Markov Decision Problems'. In: *AAAI/IAAI*, pp. 541–548.

Strens, Malcolm (2000). 'A Bayesian framework for Reinforcement Learning'. In: *Proceedings of the 17th International Conference on International Conference on Machine Learning*, pp. 943–950.

Kakade, Sham M. (2001). 'A Natural Policy Gradient'. In: *Advances in Neural Information Processing Systems*. Vol. 14.

Littman, Michael and Richard S. Sutton (2001). 'Predictive Representations of State'. In: *Advances in Neural Information Processing Systems*. Vol. 14.

Precup, Doina, Richard S. Sutton and Sanjoy Dasgupta (2001). 'Off-Policy Temporal Difference Learning with Function Approximation'. In: *Proceedings of the 18th International Conference on International Conference on Machine Learning*, pp. 417–424.

Girard, Agathe et al. (2002). 'Multiple-step ahead prediction for non linear dynamic systems–a gaussian process treatment with propagation of the uncertainty'. In: *Advances in Neural Information Processing Systems*. Vol. 15, pp. 529–536.

Munos, Rémi (2003). 'Error Bounds for Approximate Policy Iteration'. In: *Proceedings of the 20th International Conference on International Conference on Machine Learning*. Vol. 3, pp. 560–567.

Pineau, Joelle, Geoff Gordon, Sebastian Thrun et al. (2003). 'Point-based value iteration: An anytime algorithm for POMDPs'. In: *The 18th International Joint Conference on Artificial Intelligence*. Vol. 3, pp. 1025–1032.

Ferns, Norm, Prakash Panangaden and Doina Precup (2004). 'Metrics for Finite Markov Decision Processes'. In: *Conference on Uncertainty in Artificial Intelligence*. Vol. 4, pp. 162–169.

Kocijan, Ju et al. (2004). 'Gaussian process model based predictive control'. In: *Proceedings of the 2004 American control conference*. Vol. 3. IEEE, pp. 2214–2219.

Ravindran, Balaraman and Andrew G. Barto (2004). 'Approximate Homomorphisms: A Framework for Non-exact Minimization in Markov Decision Processes'. In: *International Conference on Knowledge Based Computer Systems*.

Abbeel, Pieter, Morgan Quigley and Andrew Y. Ng (2006). 'Using inaccurate models in Reinforcement Learning'. In: *Proceedings of the 23rd International Conference on Machine Learning*, pp. 1–8.

Kocsis, Levente and Csaba Szepesvári (2006). 'Bandit based Monte-Carlo Planning'. In: *European Conference on Machine Learning*. Springer, pp. 282–293.

Li, Lihong, Thomas J. Walsh and M. Littman (2006). 'Towards a Unified Theory of State Abstraction for MDPs'. In: *Proceedings of the Ninth International Symposium on Artificial Intelligence and Mathematics*, pp. 531–539.

Fern, Alan et al. (2007). 'A Decision-Theoretic Model of Assistance.' In: *Proceedings of the 20th International Joint Conference on Artificial IntelligenceAI*, pp. 1879–1884.

Melo, Francisco S. and M. Isabel Ribeiro (2007). 'Convergence of Q-learning with linear function approximation'. In: *International Conference on Computational Learning Theory*. Springer, pp. 308–322.

Kurniawati, Hanna, David Hsu and Wee Sun Lee (2008). 'SARSOP: Efficient Point-Based POMDP Planning by Approximating Optimally Reachable Belief Spaces'. In: *Robotics: Science and Systems*. Vol. 2008.

Narayanamurthy, Shravan Matthur and Balaraman Ravindran (2008). 'On the Hardness of Finding Symmetries in Markov Decision Processes'. In: *Proceedings of the 25th International Conference on International Conference on Machine Learning*, pp. 688–695.

Taylor, Jonathan, Doina Precup and Prakash Panagaden (2009). 'Bounding Performance Loss in Approximate MDP Homomorphisms'. In: *Advances in Neural Information Processing Systems*. Vol. 21.

Atrash, Amin and Joelle Pineau (2010). 'A Bayesian method for learning POMDP observation parameters for robot interaction management systems'. In: *The POMDP practitioners workshop*.

Morimura, Tetsuro et al. (2010). 'Parametric Return Density Estimation for Reinforcement Learning'. In: *Conference on Uncertainty in Artificial Intelligence*.

Silver, David and Joel Veness (2010). 'Monte-Carlo planning in Large POMDPs'. In: *Advances in Neural Information Processing Systems*. Vol. 23.

Broz, Frank, Illah Nourbakhsh and Reid Simmons (2011). 'Designing POMDP models of socially situated tasks'. In: *RO-MAN*. IEEE, pp. 39–46.

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

Dudik, Miroslav, John Langford and Lihong Li (2011). 'Doubly Robust Policy Evaluation and Learning'. In: *Proceedings of the 28th International Conference on International Conference on Machine Learning*, pp. 1097–1104.

Taha, Tarek, Jaime Valls Miró and Gamini Dissanayake (2011). 'A POMDP framework for modelling human interaction with assistive robots'. In: *2011 IEEE International Conference on Robotics and Automation*. IEEE, pp. 544–549.

Todorov, Emanuel, Tom Erez and Yuval Tassa (2012). 'MuJoCo: A physics engine for model-based control'. In: *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, pp. 5026–5033.

Keller, Thomas and Malte Helmert (2013). 'Trial-Based Heuristic Tree Search for Finite Horizon MDPs'. In: *Proceedings of the International Conference on Automated Planning and Scheduling*. Vol. 23, pp. 135–143.

Lopez-Paz, David, Philipp Hennig and B. Schölkopf (2013). 'The Randomized Dependence Coefficient'. In: *Advances in Neural Information Processing Systems*. Vol. 26.

Somani, Adhiraj et al. (2013). 'DESPOT: Online POMDP planning with regularization'. In: *Advances in Neural Information Processing Systems*. Vol. 26.

Silver, David, Guy Lever et al. (2014). 'Deterministic Policy Gradient Algorithms'. In: *Proceedings of the 31st International Conference on International Conference on Machine Learning*. PMLR, pp. 387–395.

Dinh, Laurent, David Krueger and Yoshua Bengio (2015). 'NICE: Non-linear Independent Components Estimation'. In: *Proceedings of the 3rd International Conference on Learning Representations*.

Gopalan, Nakul and Stefanie Tellex (2015). 'Modeling and Solving Human-Robot Collaborative Tasks Using POMDPs'. In: *RSS Workshop on Model Learning for Human-Robot Communication*. Vol. 32. 4, pp. 590–628.

Javdani, Shervin, Siddhartha Srinivasa and Andrew Bagnell (2015). 'Shared Autonomy via Hindsight Optimization'. In: *Robotics: Science and Systems*. NIH Public Access.

Jiang, Nan, Alex Kulesza et al. (2015). 'The Dependence of Effective Planning Horizon on Model Accuracy'. In: *Proceedings of the 2015 International Conference on Autonomous Agents and Multiagent Systems*, pp. 1181–1189.

Jiang, Shu and Ronald C. Arkin (2015). 'Mixed-Initiative Human-Robot Interaction: Definition, Taxonomy, and Survey'. In: *2015 IEEE International Conference on Systems, Man, and Cybernetics*. IEEE, pp. 954–961.

Nikolaidis, Stefanos, Ramya Ramakrishnan et al. (2015). 'Efficient Model Learning from Joint-Action Demonstrations for Human-Robot Collaborative Tasks'. In: *2015 10th ACM/IEEE International Conference on Human-Robot Interaction (HRI)*. IEEE, pp. 189–196.

Rai, Sanish, Minghao Wang and Xiaolin Hu (2015). 'A graph-based agent-oriented model for building occupancy simulation.' In: *SpringSim (ADS)*, pp. 76–83.

Ruan, Sherry Shanshan et al. (2015). 'Representation Discovery for MDPs Using Bisimulation Metrics'. In: *Twenty-Ninth AAAI Conference on Artificial Intelligence*.

Schulman, John, Sergey Levine et al. (2015). 'Trust Region Policy Optimization'. In: *Proceedings of the 32nd International Conference on International Conference on Machine Learning*. PMLR, pp. 1889–1897.

Souza, Paulo Eduardo Ubaldino de, Caroline Ponzoni Carvalho Chanel and Frédéric Dehais (2015). 'MOMDP-based target search mission taking into account the human operator's cognitive state'. In: *2015 IEEE 27th International Conference on Tools with Artificial Intelligence (ICTAI)*. IEEE, pp. 729–736.

Gal, Yarin, Rowan McAllister and Carl Edward Rasmussen (2016). 'Improving PILCO with Bayesian neural network dynamics models'. In: *Data-Efficient Machine Learning workshop, ICML*. Vol. 4. 34, p. 25.

Gu, Shixiang et al. (2016). 'Continuous Deep Q-Learning with Model-based Acceleration'. In: *Proceedings of the 33rd International Conference on International Conference on Machine Learning*. PMLR, pp. 2829–2838.

Hasselt, Hado Van, Arthur Guez and David Silver (2016). 'Deep Reinforcement Learning with Double Q-Learning'. In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 30. 1.

Jiang, Nan and Lihong Li (2016). 'Doubly Robust Off-policy Value Evaluation for Reinforcement Learning'. In: *Proceedings of the 33rd International Conference on International Conference on Machine Learning*. PMLR, pp. 652–661.

Lillicrap, Timothy P. et al. (2016). 'Continuous control with deep reinforcement learning'. In: *Proceedings of the 4th International Conference on Learning Representations*.

Mandel, Travis et al. (2016). 'Efficient Bayesian Clustering for Reinforcement Learning'. In: *Proceedings of the 25th International Joint Conference on Artificial Intelligence*, pp. 1830–1838.

Mnih, Volodymyr, Adria Puigdomenech Badia et al. (2016). 'Asynchronous Methods for Deep Reinforcement Learning'. In: *Proceedings of the 33th International Conference on International Conference on Machine Learning*. PMLR, pp. 1928–1937.

Petrik, Marek, Mohammad Ghavamzadeh and Yinlam Chow (2016). 'Safe Policy Improvement by Minimizing Robust Baseline Regret'. In: *Advances in Neural Information Processing Systems*. Vol. 29, pp. 2298–2306.

Ribeiro, Marco Tulio, Sameer Singh and Carlos Guestrin (2016). '"Why Should I Trust You?": Explaining the Predictions of Any Classifier'. In: *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*, pp. 1135–1144.

Schaul, Tom et al. (2016). 'Prioritized Experience Replay'. In: *ICLR (Poster)*.

Thomas, Philip and Emma Brunskill (2016). 'Data-Efficient Off-Policy Policy Evaluation for Reinforcement Learning'. In: *Proceedings of the 33rd International Conference on International Conference on Machine Learning*. PMLR, pp. 2139–2148.

Bellemare, Marc G., Will Dabney and Rémi Munos (2017). 'A Distributional Perspective on Reinforcement Learning'. In: *Proceedings of the 34th International Conference on International Conference on Machine Learning*. PMLR, pp. 449–458.

Drougard, Nicolas et al. (2017). 'Mixed-initiative mission planning considering human operator state estimation based on physiological sensors'. In: *IROS-2017 Workshop on Human-Robot Interaction in Collaborative Manufacturing Environments (HRI-CME)*.

Jaques, Natasha et al. (2017). 'Sequence Tutor: Conservative Fine-Tuning of Sequence Generation Models with KL-control'. In: *Proceedings of the 34th International Conference on International Conference on Machine Learning*. PMLR, pp. 1645–1654.

Lundberg, Scott M. and Su-In Lee (2017). 'A Unified Approach to Interpreting Model Predictions'. In: *Advances in Neural Information Processing Systems*. Vol. 30.

Nikolaidis, Stefanos, Yu Xiang Zhu et al. (2017). 'Human-Robot Mutual Adaptation in Shared Autonomy'. In: *2017 12th ACM/IEEE International Conference on Human-Robot Interaction (HRI)*. IEEE, pp. 294–302.

Shrikumar, Avanti, Peyton Greenside and Anshul Kundaje (2017). 'Learning Important Features Through Propagating Activation Differences'. In: *Proceedings of the 34th International Conference on International Conference on Machine Learning*. PMLR, pp. 3145–3153.

Charles, Jack-Antoine et al. (2018). 'Human-Agent Interaction Model Learning based on Crowdsourcing'. In: *Proceedings of the 6th International Conference on Human-Agent Interaction*, pp. 20–28.

Chua, Kurtland et al. (2018). 'Deep Reinforcement Learning in a Handful of Trials using Probabilistic Dynamics Models'. In: *Advances in Neural Information Processing Systems*. Vol. 31.

Dabney, Will, Georg Ostrovski et al. (2018). 'Implicit Quantile Networks for Distributional Reinforcement Learning'. In: *Proceedings of the 35th International Conference on International Conference on Machine Learning*. PMLR, pp. 1096–1105.

Dabney, Will, Mark Rowland et al. (2018). 'Distributional Reinforcement Learning With Quantile Regression'. In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 32. 1.

Depeweg, Stefan et al. (2018). 'Decomposition of Uncertainty in Bayesian Deep Learning for Efficient and Risk-sensitive Learning'. In: *Proceedings of the 35th International Conference on International Conference on Machine Learning*. PMLR, pp. 1184–1193.

Fujimoto, Scott, Herke Hoof and David Meger (2018). 'Addressing Function Approximation Error in Actor-Critic Methods'. In: *Proceedings of the 35th International Conference on International Conference on Machine Learning*. PMLR, pp. 1587–1596.

Geffner, Hector (2018). 'Model-free, Model-based, and General Intelligence'. In: *Proceedings of the 27th International Joint Conference on Artificial Intelligence*, pp. 10–17.

Haarnoja, Tuomas et al. (2018). 'Soft Actor-Critic: Off-Policy Maximum Entropy Deep Reinforcement Learning with a Stochastic Actor'. In: *Proceedings of the 35th International Conference on International Conference on Machine Learning*. PMLR, pp. 1861–1870.

Kamthe, Sanket and Marc Deisenroth (2018). 'Data-Efficient Reinforcement Learning with Probabilistic Model Predictive Control'. In: *International Conference on Artificial Intelligence and Statistics*. PMLR, pp. 1701–1710.

Liu, Qiang et al. (2018). 'Breaking the Curse of Horizon: Infinite-Horizon Off-Policy Estimation'. In: *Advances in Neural Information Processing Systems*. Vol. 31, pp. 5356–5366.

Nagabandi, Anusha et al. (2018). 'Neural Network Dynamics for Model-Based Deep Reinforcement Learning with Model-Free Fine-Tuning'. In: *2018 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, pp. 7559–7566.

Zheng, Wei, Bo Wu and Hai Lin (2018). 'POMDP Model Learning for Human Robot Collaboration'. In: *2018 IEEE Conference on Decision and Control (CDC)*. IEEE, pp. 1156–1161.

Devlin, Jacob et al. (2019). 'BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding'. In: *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Volume 1 (Long and Short Papers)*, pp. 4171–4186.

Fujimoto, Scott, David Meger and Doina Precup (2019). 'Off-policy Deep Reinforcement Learning without exploration'. In: *Proceedings of the 36th International Conference on International Conference on Machine Learning*. PMLR, pp. 2052–2062.

Grathwohl, Will et al. (2019). 'FFJORD: Free-Form Continuous Dynamics for Scalable Reversible Generative Models'. In: *Proceedings of the 7th International Conference on Learning Representations*.

Hafner, Danijar, Timothy Lillicrap, Jimmy Ba et al. (2019). 'Dream to Control: Learning Behaviors by Latent Imagination'. In: *International Conference on Learning Representations*.

Hafner, Danijar, Timothy Lillicrap, Ian Fischer et al. (2019). 'Learning Latent Dynamics for Planning from Pixels'. In: *Proceedings of the 36th International Conference on International Conference on Machine Learning*. PMLR, pp. 2555–2565.

Kumar, Aviral, Justin Fu et al. (2019). 'Stabilizing Off-Policy Q-Learning via Bootstrapping Error Reduction'. In: *Advances in Neural Information Processing Systems*. Vol. 33, pp. 11784–11794.

Laroche, Romain, Paul Trichelair and Remi Tachet Des Combes (2019). 'Safe Policy Improvement with Baseline Bootstrapping'. In: *Proceedings of the 36th International Conference on International Conference on Machine Learning*. PMLR, pp. 3652–3661.

Le, Hoang, Cameron Voloshin and Yisong Yue (2019). 'Batch Policy Learning under Constraints'. In: *Proceedings of the 36th International Conference on International Conference on Machine Learning*. PMLR, pp. 3703–3712.

Lecarpentier, Erwan and Emmanuel Rachelson (2019). 'Non-stationary Markov Decision Processes, a Worst-Case Approach using Model-Based Reinforcement Learning'. In: *Advances in Neural Information Processing Systems*. Vol. 32.

Petrik, Marek and Reazul Hasan Russel (2019). 'Beyond Confidence Regions: Tight Bayesian Ambiguity Sets for Robust MDPs'. In: *Advances in Neural Information Processing Systems*. Vol. 32.

Yang, Derek et al. (2019). 'Fully Parameterized Quantile Function for Distributional Reinforcement Learning'. In: *Advances in Neural Information Processing Systems*. Vol. 32.

Abel, David et al. (2020). 'Value Preserving State-Action Abstractions'. In: *International Conference on Artificial Intelligence and Statistics*. PMLR, pp. 1639–1650.

Agarwal, Rishabh, Dale Schuurmans and Mohammad Norouzi (2020). 'An Optimistic Perspective on Offline Reinforcement Learning'. In: *Proceedings of the 37th International Conference on International Conference on Machine Learning*. PMLR, pp. 104–114.

Angelotti, Giorgio, Nicolas Drougard and Caroline Ponzoni Carvalho Chanel (2020). 'Offline Learning for Planning: A Summary'. In: *Proceedings of the 1st Workshop on Bridging the Gap Between AI Planning and Reinforcement Learning at the 30th International Conference on Automated Planning and Scheduling*, pp. 153–161.

Brown, Tom et al. (2020). 'Language Models are Few-Shot Learners'. In: *Advances in Neural Information Processing Systems*. Vol. 33, pp. 1877–1901.

Castro, Pablo Samuel (2020). 'Scalable Methods for Computing State Similarity in Deterministic Markov Decision Processes'. In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 34. 06, pp. 10069–10076.

Hafner, Danijar, T. Lillicrap et al. (2020). 'Mastering Atari with Discrete World Models'. In: *International Conference on Learning Representations*.

Kidambi, Rahul et al. (2020). 'MOReL: Model-Based Offline Reinforcement Learning'. In: *Advances in Neural Information Processing Systems*. Vol. 33, pp. 21810–21823.

Kumar, Aviral, Aurick Zhou et al. (2020). 'Conservative Q-Learning for Offline Reinforcement Learning'. In: *Advances in Neural Information Processing Systems*. Vol. 34, pp. 20132–20145.

Kurach, Karol et al. (2020). 'Google Research Football: A Novel Reinforcement Learning Environment'. In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 34. 04, pp. 4501–4510.

Lee, Byungjun et al. (2020). 'Batch Reinforcement Learning with Hyperparameter Gradients'. In: *Proceedings of the 37th International Conference on International Conference on Machine Learning*. PMLR, pp. 5725–5735.

van der Pol, Elise, Thomas Kipf et al. (2020). 'Plannable Approximations to MDP Homomorphisms: Equivariance under Actions'. In: *Proceedings of the 19th International Conference on Autonomous Agents and MultiAgent Systems*, pp. 1431–1439.

van der Pol, Elise, Daniel Worrall et al. (2020). 'MDP Homomorphic Networks: Group Symmetries in Reinforcement Learning'. In: *Advances in Neural Information Processing Systems*. Vol. 33, pp. 4199–4210.

Yu, Tianhe et al. (2020). 'MOPO: Model-based Offline Policy Optimization'. In: *Advances in Neural Information Processing Systems*. Vol. 33, pp. 14129–14142.

Behzadian, Bahram et al. (Apr. 2021). 'Optimizing Percentile Criterion using Robust MDPs'. In: *Proceedings of the 24th International Conference on Artificial Intelligence and Statistics*. Vol. 130. Proceedings of Machine Learning Research. PMLR, pp. 1009–1017.

Chandak, Yash et al. (2021). 'Universal Off-Policy Evaluation'. In: *Advances in Neural Information Processing Systems*. Vol. 35.

Chen, Lili et al. (2021). 'Decision Transformer: Reinforcement Learning via Sequence Modeling'. In: *Advances in Neural Information Processing Systems*. Vol. 34, pp. 15084–15097.

Fujimoto, Scott and Shixiang Shane Gu (2021). 'A Minimalist Approach to Offline Reinforcement Learning'. In: *Advances in Neural Information Processing Systems*. Vol. 34, pp. 20132–20145.

Janner, Michael, Qiyang Li and Sergey Levine (2021). 'Offline Reinforcement Learning as One Big Sequence Modeling Problem'. In: *Advances in Neural Information Processing Systems*. Vol. 34, pp. 1273–1286.

Kaufmann, Marcel et al. (2021). 'Copilot MIKE: An Autonomous Assistant for Multi-Robot Operations in Cave Exploration'. In: *2021 IEEE Aerospace Conference (50100)*. IEEE, pp. 1–9.

Takuma Seno, Michita Imai (Dec. 2021). 'd3rlpy: An Offline Deep Reinforcement Library'. In: *NeurIPS 2021 Offline Reinforcement Learning Workshop*.

Zhang, Siyuan and Nan Jiang (2021). 'Towards Hyperparameter-free Policy Selection for Offline Reinforcement Learning'. In: *Advances in Neural Information Processing Systems*. Vol. 34.

Angelotti, Giorgio, Nicolas Drougard and Caroline Ponzoni Carvalho Chanel (2022). 'Expert-guided Symmetry Detection in Markov Decision Processes'. In: *Proceedings of the 14th International Conference on Agents and Artificial Intelligence - Volume 2: ICAART,* INSTICC. SciTePress, pp. 88–98.

Lahire, Thibault, Matthieu Geist and Emmanuel Rachelson (2022). 'Large Batch Experience Replay'. In: *Proceedings of the 39th International Conference on Machine Learning*. PMLR, pp. 11790–11813.

Singh, Gaganpreet, Raphaëlle N. Roy and Caroline P. C. Chanel (2022). 'POMDP-Based Adaptive Interaction Through Physiological Computing'. In: *HHAI2022: Augmenting Human Intellect*. IOS Press, pp. 32–45.

Angelotti, Giorgio, Nicolas Drougard and Caroline Ponzoni Carvalho Chanel (2023b). 'Data Augmentation Through Expert-Guided Symmetry Detection to Improve Performance in Offline Reinforcement Learning'. In: *Proceedings of the 15th International Conference on Agents and Artificial Intelligence - Volume 2: ICAART,* INSTICC. SciTePress, pp. 115–124.

Preprints

Brockman, Greg et al. (2016). 'OpenAI Gym'. arXiv:1606.01540.

Schulman, John, Filip Wolski et al. (2017). 'Proximal Policy Optimization Algorithms'. arXiv:1707.06347.

Fedus, W. et al. (2019). 'Hyperbolic Discounting and Learning over Multiple Horizons'. arXiv:1902.06865.

Fujimoto, Scott, Edoardo Conti et al. (2019). 'Benchmarking Batch Deep Reinforcement Learning Algorithms'. arXiv:1910.01708.

Tarkowski, M. et al. (2019). 'Monte Carlo Techniques for Approximating the Myerson Value - Theoretical and Empirical Analysis'. arXiv:2001.00065.

Fu, Justin et al. (2020). 'D4RL: Datasets for Deep Data-Driven Reinforcement Learning'. arXiv:2004.07219.

Levine, Sergey et al. (2020). 'Offline Reinforcement Learning: Tutorial, Review, and Perspectives on Open Problems'. arXiv:2005.01643.

Paine, Tom Le et al. (2020). 'Hyperparameter Selection for Offline Reinforcement Learning'. arXiv:2007.09055.

Lobo, Elita A., Mohammad Ghavamzadeh and Marek Petrik (2021). 'Soft-Robust Algorithms for Batch Reinforcement Learning'. arXiv:2011.14495.

Yang, Chao-Han Huck et al. (2021). 'Pessimistic Model Selection for Offline Deep Reinforcement Learning'. arXiv:2111.14346.

Mandhane, Amol et al. (2022). 'MuZero with Self-competition for Rate Control in VP9 Video Compression'. arXiv:2202.06626.

Reed, Scott et al. (2022). 'A Generalist Agent'. arXiv:2205.06175.

Angelotti, Giorgio, Nicolas Drougard and Caroline Ponzoni Carvalho Chanel (2023a). 'An Offline Risk-aware Policy Selection Method for Bayesian Markov Decision Processes'. arXiv:2105.13431.

Rowland, Mark et al. (2023). 'An Analysis of Quantile Temporal-Difference Learning'. arXiv:2301.04462.

Other sources

Ravindran, B. and A. G. Barto (2001). *Symmetries and Model Minimization in Markov Decision Processes*. Tech. rep. USA.

Downey, Carlton, Ahmed Hefny and Geoffrey Gordon (2017). *Practical Learning of Predictive State Representations*. Tech. rep.

Lundberg, Scott (2018). *Census income classification with scikit-learn*. https://shap.readthedocs.io/en/latest/tabular_examples.html. [Online; accessed 12-April-2023].

European Commission (2020). *White Paper on Artificial Intelligence - A European approach to excellence and trust*. https://ec.europa.eu/info/sites/default/files/commission-white-paper-artificial-intelligence-feb2020_en.pdf. [Online; accessed 12-April-2023].

Penedo, Guilherme (Mar. 2022). *Using conditional GANs to develop a realistic human-robot interaction simulator*. Tech. rep. [Online; accessed 12-April-2023]. ISAE Supaero.

Wikipedia contributors (2023). *World of Warcraft — Wikipedia, The Free Encyclopedia*. https://en.wikipedia.org/w/index.php?title=World_of_Warcraft&oldid=1142763722. [Online; accessed 12-April-2023].