Faculty of Science and Bio-Engineering Sciences
Department of Computer Science
Artificial Intelligence Laboratory

Leveraging State Access in Partially Observable Sequential Decision-Making

Dissertation submitted in fulfillment of the requirements for the degree of Doctor of Science: Computer Science

Raphaël Avalos

Brussels, February 2026

Promotors:
Prof. Dr. Ann Nowé (Vrije Universiteit Brussel)
Dr. Diederik M. Roijers (Vrije Universiteit Brussel, City of Amsterdam)

© 2026 Raphaël Avalos

Printed by
Crazy Copy Center Productions
VUB Pleinlaan 2, 1050 Brussel
Tel : +32 2 629 33 44
crazycopy@vub.ac.be
www.crazycopy.be

ISBN 9789493461413
NUR 984
THEMA: UYQM

Alle rechten voorbehouden. Niets van deze uitgave mag worden vermenigvuldigd en/of openbaar gemaakt worden door middel van druk, fotokopie, microfilm, electronisch of op welke andere wijze ook, zonder voorafgaande schriftelijke toestemming van de auteur.

All rights reserved. No part of this publication may be produced in any form by print, photoprint, microfilm, electronic or any other means without permission from the author.

Members of the Jury

Prof. Dr. Coen De Roover — Chairperson
SOFT, VUB

Prof. Dr. Lynn Houthuys — Secretary
AI Lab, DINF, VUB

Prof. Dr. Jan Lemeire
ETRO, VUB

Dr. Nicole Orzan
AI Lab, VUB

Prof. Dr. Christopher Amato
Khoury College of Computer Sciences, Northeastern University

Prof. Dr. Karl Tuyls
IMEC; University of Liverpool

Abstract

Many decision making problems in robotics, autonomous systems, and multi-agent coordination involve partial observability, where agents must act based on incomplete and noisy information about the environment. This challenge often arises from the physical limitations of sensors, which may capture only certain aspects of the underlying state, sometimes unreliably. As a result, the agent must estimate the current situation, predict future outcomes, and choose actions under uncertainty. While most approaches assume the state is never available, in practice it can sometimes be accessed in two important ways: during training, for example through a simulator that provides ground-truth states, or at a cost during execution by activating high-precision auxiliary sensors. Such access makes state information a valuable but limited resource.

This thesis investigates how strategic use of this resource can improve learning and planning under uncertainty. In reinforcement learning, we show how access to state information during training can supervise internal representations, stabilize learning, and improve performance at deployment without state access. In planning, we formalize and solve decision problems where the agent can request the state during execution, weighing the benefit of information against its cost.

We propose three methods. First, we introduce Local Advantage Networks (LAN) for cooperative multi-agent reinforcement learning, which stabilizes the training of independent agents through a centralized value baseline without requiring joint action-value factorization, achieving state-of-the-art performance among value-based methods on the SMAC benchmark. Second, we present AEMS-SR, a graph-based online planning algorithm for POMDPs with state requests, which extends Anytime Error Minimization Search to handle costly state queries while retaining ε-optimality guarantees and outperforming standard planners in test domains. Third, we develop the Wasserstein Belief Updater (WBU) for model-based reinforcement learning, which leverages state access during training to learn latent belief updates. WBU provides theoretical guarantees on belief quality via bisimulation distances and achieves strong empirical performance in partially observable environments.

Across these settings, we treat state access as a structured and limited resource. By showing when and how to exploit it in both learning and planning, this thesis demonstrates that agents can achieve more robust and effective decision making under partial observability.

Samenvatting

Veel beslissingsproblemen in robotica, autonome systemen en multi-agentcoördinatie gaan gepaard met partiële observeerbaarheid, waarbij agenten moeten handelen op basis van onvolledige en ruisachtige informatie over de omgeving. Deze uitdaging vloeit vaak voort uit de fysieke beperkingen van sensoren, die slechts bepaalde aspecten van de onderliggende toestand kunnen vastleggen, soms op onbetrouwbare wijze. Het gevolg is dat de agent de huidige situatie moet inschatten, toekomstige uitkomsten moet voorspellen en acties moet kiezen onder onzekerheid. Terwijl de meeste toepassingen veronderstellen dat de toestand nooit beschikbaar is, kan deze in de praktijk soms toch worden geraadpleegd op twee belangrijke manieren: tijdens training, bijvoorbeeld via een simulator die grondwaarheidstoestanden verschaft, of tegen een kostprijs tijdens uitvoering door het activeren van nauwkeurige hulpsensoren. Dergelijke toegang maakt toestandinformatie tot een waardevolle maar beperkte hulpbron.

Dit proefschrift onderzoekt hoe strategisch gebruik van deze hulpbron het leren en plannen onder onzekerheid kan verbeteren. In reinforcement learning tonen we aan hoe toegang tot toestandinformatie tijdens training interne representaties kan sturen, het leerproces kan stabiliseren en de prestaties bij inzet zonder toestandinformatie kan verbeteren. In planning formaliseren en lossen we beslissingsproblemen op waarbij de agent de toestand tijdens uitvoering kan opvragen, waarbij de winst van de informatie wordt afgewogen tegen de kostprijs.

We stellen drie methoden voor. Ten eerste introduceren we Local Advantage Networks (LAN) voor coöperatief multi-agent reinforcement learning, dat de training van onafhankelijke agenten stabiliseert via een gecentraliseerde waardebaseline zonder gezamenlijke actie-waardefactorisatie te vereisen, en dat state-of-the-art prestaties behaalt onder waardegebaseerde methoden op de SMAC-benchmark. Ten tweede presenteren we AEMS-SR, een grafgebaseerd online planningsalgoritme voor POMDP's met toestandsaanvragen, dat Anytime Error Minimization Search uitbreidt om kostbare toestandsqueries af te handelen, terwijl het ε-optimaliteitsgaranties behoudt en beter presteert dan standaardplanners in testdomeinen. Ten derde ontwikkelen we de Wasserstein Belief Updater (WBU) voor model-based reinforcement learning, die

toestandinformatie tijdens training benut om latente belief-updates te leren. WBU biedt theoretische garanties voor de kwaliteit van beliefs via bisimulatieafstanden en behaalt sterke empirische prestaties in partieel observeerbare omgevingen.

Over deze verschillende contexten heen behandelen we toestandinformatie als een gestructureerde en beperkte hulpbron. Door aan te tonen wanneer en hoe deze kan worden benut in zowel leren als plannen, laat dit proefschrift zien dat agenten robuustere en effectievere beslissingen kunnen nemen onder partiële observeerbaarheid.

Acknowledgements

I would like to begin by thanking my advisors, Prof. Ann Nowé and Dr. Diederik M. Roijers. Ann, I am deeply grateful for believing in me and giving me the opportunity to pursue my PhD at the AI Lab. The freedom of research you provided allowed me to explore my ideas and develop them into this work. Diederik, you reached out to me at the very beginning of COVID, just as I was starting this journey. Your guidance through weekly meetings over these five years helped me stay focused and on track, especially during the most challenging times.

I would like to thank the members of my doctoral examination committee, Prof. Dr. Coen De Roover, Prof. Dr. Lynn Houthuys, Prof. Dr. Jan Lemeire, Dr. Nicole Orzan, Prof. Dr. Christopher Amato, and Prof. Dr. Karl Tuyls, for taking the time to review this dissertation and for providing valuable feedback.

I am grateful to the FWO for the fundamental research fellowship that made this research possible.

The AI Lab has been a wonderful place to work, and I thank all its members for creating such a stimulating and friendly environment. Mathieu, you were the only one brave enough to come to the office during COVID with me. We improved our ping-pong game together, and you introduced me to Chouffe O'clock. Florent, we started at the AI Lab around the same time and became friends in our big corner office. I had a lot of fun collaborating with you on our papers, and also outside the lab, such as when you introduced me to the Doudou. Willem, I really enjoyed the two summer schools we did and organizing EWRL together. It was great working with you, even though you kept singing La Marseillaise. Hicham, you are very kind and always there to help. I had a lot of fun trying to understand weird MCMC algorithms with you. I would also like to thank Roxana for helping me with my FWO proposal when I started my PhD, Lucas for bringing Brazilian magic to the lab, Andries for our two trips to the US, Andrea for your Italian recipes, Eugenio for your C++ wizardry, Anca for your help in navigating the administrative maze, and all other current and former members of the AI Lab for the great atmosphere.

I am grateful to my co-authors Qingshuang and Denis for their contributions to this work, and to Prof. Frans Oliehoek for hosting me at TU Delft.

To all my friends, especially Hugo, Auriane, Raphaël, Théophile, Val, Nadia, Maite, François, and Nelson, thank you for your friendship and for helping me try to find balance during these years.

I am deeply grateful to my parents and my sister Diana for your unconditional love and support. You gave me the curiosity and confidence that made this work possible.

Finally, Barbara, the love of my life, thank you for your infinite patience, support, and love during these years. You have been my menhir.

To Yasmine,
who left us far too early but remains forever in my heart.

Contents

1 Introduction

1.1 Motivation

Imagine a robot learning to navigate a warehouse. It must avoid shelves, pick up packages, and reach delivery points, all while relying on noisy sensors, partially obstructed views, and imperfect localization. Acting effectively in such a setting is difficult because the robot does not have full access to the environment's true state. This is a common feature of real-world systems: from drones with low resolution sensors to voice assistants interpreting ambiguous speech, agents must routinely make decisions under partial observability.

Yet despite this uncertainty, real-world agents are not always equally impaired. During training, for instance, the robot might operate in a controlled lab. Engineers can place external motion-tracking cameras, tap into the simulator's memory, or log the exact position and velocity of every object in the scene. In such settings, accessing the full state of the environment is not only possible but often straightforward. And while this privileged information will not be available at deployment, it is natural to assume that it can still be valuable during learning. Indeed, by observing the true state during training, the agent can gain a clearer understanding of how its actions influence the environment, free from the noise and ambiguity of partial observations. This richer feedback can help shape internal models or representations that remain useful even when the agent must later act with limited information.

In some scenarios access to the true state is also possible during deployment but at a cost. The same robot might carry a high-precision sensor that consumes extra power or takes time to activate. It can switch it on to resolve uncertainty, but doing so drains battery life or slows down operations. In such cases, the agent must reason not just about which action to take, but whether it is worth paying to reveal the state. Should it spend its limited resources to gain certainty, or act based on partial observations?

These two scenarios, where the agent has access to the state either freely during training or at a cost during execution, are typically overlooked in the literature on partially observable sequential decision making, with, for the former, the exception of multi-agent reinforcement learning. Because agents must act under partial observability at deployment, most approaches impose the same constraint throughout learning and planning. However, in practice, access to the true state is sometimes possible, provided at the right moment and in the right form. This dissertation investigates how such moments of state access can be systematically exploited to improve learning and decision making.

1.2 Problem Setting and Research Question

The challenge of learning and planning under partial observability arises in many decision-theoretic models, including contextual bandits, Bayesian games, and decentralized multi-agent systems. In this thesis, we focus on the setting of Partially Observable Markov Decision Processes (POMDPs) [Åström, 1965; Kaelbling et al., 1998], which provide a standard formalism for sequential decision making when the agent cannot observe the true state of the environment. We also consider their extension to collaborative multi-agent settings through Decentralized POMDPs (Dec-POMDPs) [Oliehoek and Amato, 2016]. In this context, a widely used learning paradigm is Centralized Training with Decentralized Execution (CTDE) [Lowe et al., 2017; Foerster et al., 2018], where agents can learn from the full state and other agent's policies during training but act using only local observations at execution. This illustrates one established setting where state access is leveraged systematically. Within reinforcement learning, privileged state access has predominantly been exploited in multi-agent settings; comparatively fewer works have explored this paradigm in single-agent reinforcement learning, with some notable exceptions such as asymmetric actor–critic methods [Pinto et al., 2017]. In the POMDP framework, the agent interacts with the environment over a sequence of time steps, receiving observations that are typically partial, noisy, and insufficient to uniquely identify the underlying state. Decision making under such conditions is substantially more difficult than in fully observable settings.

To act effectively, the agent must rely on memory or perform inference over past interactions to construct a representation of what it has seen. This can take the form of a recurrent hidden state, a latent embedding, or a belief, which is a probability

distribution over possible environment states. The policy must then condition on this internal representation rather than the unobserved state. As a result, both planning and reinforcement learning become substantially more difficult. Planning must account for uncertainty over the state, requiring reasoning over the exponentially increasing space of possible histories or the computationally intensive beliefs. Learning becomes less stable and more sample inefficient, as the agent receives only partial and noisy feedback about the consequences of its actions.

At the heart of this difficulty lies the problem of representation. Because the agent does not observe the state directly, it must infer a sufficient statistic of its history that preserves the information necessary for good decision making. If the learned representation fails to capture key information, value estimation becomes inaccurate and policies degrade.

This leads to the central research question of the dissertation:

How and when can agents strategically use access to the true state of the environment, whether during training or at execution time under a cost, to improve performance in partially observable sequential decision making problems?

1.3 Core Thesis Perspective

Traditional approaches within the POMDP framework assume that the state is always hidden and that agents must rely solely on observation histories or beliefs. While this assumption is convenient for theory, it is overly restrictive for many practical systems. In reality, the true state is sometimes available: freely during training through simulators or instrumented environments, or at execution through costly high-precision sensors. Rather than treating *state access* as an exception, this dissertation proposes to view it as a structured resource that can be selectively exploited to improve learning and decision making.

This perspective underlies all contributions of the dissertation and serves as a unifying lens across different methods. The work spans two foundational and complementary paradigms in sequential decision making: reinforcement learning and planning.

Reinforcement Learning [Sutton and Barto, 1998] focuses on learning to act through experience. The agent interacts with the environment, receives rewards, and gradually improves its behavior by learning which actions lead to better outcomes. Under partial observability, this process becomes more difficult, as the agent must rely on incomplete and noisy information. When the true state is accessible during training, it can supervise the learning of internal representations, value functions, or latent models. This guidance enables the agent to form more accurate beliefs and more reliable predictions. Even

if state access is no longer available at deployment, the resulting representations can support better performance.

Planning [Kaelbling et al., 1998] involves reasoning about future outcomes before selecting actions. When the environment model is known or can be simulated, the agent can evaluate alternative action sequences by predicting their consequences. In partially observable settings, uncertainty over the current state complicates this process. In the setting considered here, the agent may query the true state during execution, but must weigh the benefit of reduced uncertainty against the cost of the query. Effective strategies therefore combine action selection with information acquisition, balancing the value of information against the resources required to obtain it.

This view of state access as a structured and limited resource motivates the algorithms developed in the following chapters. Each method investigates a different setting in which access to the true state is possible in some form, and asks how the agent can use this information most effectively. While this dissertation treats reinforcement learning and planning separately, their integration remains a promising avenue for future work. Whether through learning or planning, the central goal remains the same: to improve decision making under partial observability by using state information at the right time and in the right way.

1.4 Contributions

This dissertation develops methods that exploit access to the true state of the environment, whether during training or at execution under a cost, to improve decision making in partially observable settings. The contributions span both reinforcement learning and planning, unified by the perspective that state access is a structured resource for learning, representation, and action selection.

We note that the *fully observable learning with partially observable execution* setting, where the agent uses the environment's states to guide learning, has been explored in the multi-agent reinforcement learning literature under the CTDE paradigm. However, this thesis takes a broader view: it investigates how structured access to the state, whether during learning, execution, or both, can support decision making under uncertainty. In the case of planning, the setting differs fundamentally: the agent does not learn from experience but plans using a known model, which is by definition fully specified. As a result, there is no analogue to fully observable learning with partially observable execution in planning, and the role of state access shifts from supervision to selective querying at execution time. This distinction motivates the complementary lines of work explored in the thesis.

Local Advantage Networks (LAN). We propose LAN, a cooperative multi-agent reinforcement learning algorithm that stabilizes training under partial observability. Unlike value-decomposition methods that rely on joint action-value factorization, LAN combines locally learned advantage functions with a centralized value function trained with state access under the CTDE paradigm. This design improves credit assignment, mitigates non-stationarity, and remains fully compatible with decentralized execution. On the SMAC benchmark, LAN achieves state-of-the-art performance among value-based methods, demonstrating the effectiveness of leveraging state supervision during training.

AEMS-SR: Online Planning with Costly State Access. We introduce the POMDP-SR framework, where agents may query the true state at a known cost during execution. Standard tree search becomes intractable in this setting due to the exponential branching induced by queries. To address this, we propose AEMS-SR, a graph-based extension of heuristic search that reuses belief nodes across trajectories, thereby avoiding redundant expansions while preserving solution-quality guarantees. This shows how integrating information acquisition into planning enables robust and cost-sensitive decision making.

Wasserstein Belief Updater (WBU). We develop WBU, a model-based RL algorithm that learns belief updates in latent space. While recurrent encoders are the standard approach to partial observability, they provide no sufficiency guarantees and depend on unstable backpropagation through time. WBU leverages state access during training to learn a latent dynamics model and a corresponding belief updater, trained to approximate belief updates while providing theoretical guarantees. At deployment, the agent plans and acts in the latent MDP defined by these beliefs, enabling scalable and principled representation learning for decision making under uncertainty.

Together, these contributions demonstrate how treating state access as a structured and limited resource can improve both learning and planning under partial observability, advancing the broader goal of enabling agents to make robust decisions in complex environments.

1.5 Thesis Structure

The remainder of this dissertation is structured as follows.

Chapter 2 presents background material on reinforcement learning, planning, and decision making under partial observability, with a focus on representation learning and belief-based methods.

Chapter 3 to 5 form the core contributions of the dissertation. Each chapter addresses the central question of how access to the true state can improve learning or planning under uncertainty. Chapter 3 introduces a reinforcement learning algorithm for multi-agent coordination with centralized training. Chapter 4 presents an online planning algorithm for environments where state access is available at a cost. Chapter 5 develops a model-based reinforcement learning approach that enables latent belief learning with theoretical guarantees.

Finally, Chapter 6 concludes the dissertation with a summary of the findings and a discussion of limitations and future directions.

The appendix includes an overview of the papers that form the core of this dissertation, as well as additional research contributions made during the PhD that are not included in the main chapters.

2 Background

In this chapter, we present the foundational concepts necessary to understand the methods and contributions of this dissertation. We begin with mathematical preliminaries (Section 2.1), which establish notation and tools such as divergences and optimal transport that will later underpin the analysis of our Wasserstein Belief Updater in Chapter 5. We then review Markov decision processes (Section 2.2), which serve as a fully observable baseline and a stepping stone toward partially observable settings. Building on this, we introduce partially observable Markov decision processes (Section 2.3), the central model of this thesis and the foundation for our online planning approach in Chapter 4 and our model-based RL method in Chapter 5. Next, we extend to the multi-agent setting of decentralized POMDPs with centralized training and decentralized execution (Section 2.4), which provides the substrate for our multi-agent value-based method in Chapter 3. Finally, we survey representation learning techniques (Section 2.5), a key ingredient for scaling to complex environments and the basis for our contribution in Chapter 5.

2.1 Mathematical Background

This section introduces core mathematical tools used throughout the dissertation, including measure-theoretic probability, divergences, and optimal transport. These concepts provide the formal language for reasoning about uncertainty, state estimation,

and representation learning in partially observable environments. Measure-theoretic probability [Klenke, 2008] offers a rigorous foundation for defining random variables, distributions, and expectations in both discrete and continuous spaces, which is essential when formalizing decision processes and belief updates. Divergences such as Kullback–Leibler [Kullback and Leibler, 1951] and total variation [Dudley, 2018] distances quantify differences between probability distributions, enabling us to compare beliefs, measure information gain, and regularize learned models. Optimal transport theory [Villani, 2009] provides a way to compare probability distributions that accounts for the geometry of the underlying space, a property that we exploit to define Wasserstein-based distances for representation learning and bisimulation. By introducing these concepts here, we prepare the groundwork for the formal analysis and algorithmic contributions in later chapters, where they serve as building blocks.

2.1.1 Measure and Integration

In this subsection, we introduce some basic concepts from measure theory and integration. We do not aim to provide a comprehensive overview, but rather to establish the notation and concepts that will be used throughout the dissertation. For a more detailed introduction to measure theory, we refer the reader to Klenke [2008] or Delgrange [2024].

Probability Spaces and Measurability

A probability space is a triple $(\Omega, \mathcal{F}, \mathbb{P})$, where Ω is the sample space, $\mathcal{F}$ is a σ-algebra, and $\mathbb{P}$ is a probability measure with $\mathbb{P}(\Omega) = 1$. *In words: Ω contains all possible outcomes (e.g., the faces of a die), $\mathcal{F}$ is the collection of events we can assign probabilities to (e.g., "the die shows an even number"), and $\mathbb{P}$ specifies the probability of each event.*

Let $(\mathcal{X}, \Sigma(\mathcal{X}))$ be a measurable space. A function $X : \Omega \to \mathcal{X}$ is a random variable if it is measurable, that is,

$$X^{-1}(B) \in \mathcal{F} \quad \text{for all } B \in \Sigma(\mathcal{X}).$$

Probability Densities and Expectations

Assuming a probability measure $\mathbb{P}$ on $\mathcal{X}$ admits a density p with respect to a reference measure μ (e.g., Lebesgue or counting), denoted

$$\mathrm{d}\mathbb{P}(x) = p(x)\,\mathrm{d}\mu(x).$$

The expectation of a measurable function $f : \mathcal{X} \to \mathbb{R}$ is then

$$\mathop{\mathbb{E}}_{\mathbb{P}}[f(X)] = \int_{\mathcal{X}} f(x)\,\mathrm{d}\mathbb{P}(x) = \int_{\mathcal{X}} f(x)\,p(x)\,\mathrm{d}\mu(x).$$

In the discrete case, this reduces to a sum over $\mathcal{X}$.

$$\mathop{\mathbb{E}}_{\mathbb{P}}[f(X)] = \sum_{x \in \mathcal{X}} p(x) f(x).$$

Dirac Delta and Indicator Functions

The Dirac delta at $x \in \mathcal{X}$, denoted δ_x, is the distribution defined by

$$\int_{\mathcal{X}} f(y)\, \delta_x(y)\, \mathrm{d}y = f(x),$$

for all functions f. It represents a deterministic distribution concentrated at x.

The Dirac delta can also be viewed as the limit of a sequence of probability densities that converge to a point mass at x, such as a Gaussian centered at x with variance tending to zero.

The indicator function $\mathbb{1}_A(x)$ returns 1 if $x \in A$ and 0 otherwise. When $A = \{x_0\}$, we write $\mathbb{1}_{x_0}(x)$.

Simplex

The probability simplex over a finite set $\mathcal{X}$ is defined as

$$\Delta_{\mathcal{X}} = \left\{ p \in \mathbb{R}^{|\mathcal{X}|} \;\middle|\; p(x) \geq 0 \text{ for all } x \in \mathcal{X}, \sum_{x \in \mathcal{X}} p(x) = 1 \right\}.$$

Support

The support of a probability measure $\mathbb{P}$, denoted $\mathrm{supp}(\mathbb{P})$, is the smallest closed set $A \subseteq \mathcal{X}$ such that $\mathbb{P}(A) = 1$. It contains all points where the measure is non-zero. For a density p, the support is the closure of the set $\{x \in \mathcal{X} \mid p(x) > 0\}$.

This formalism is crucial for defining and understanding the notion of beliefs and belief updates (Sondik [1971]; Kaelbling et al. [1998];Property 2.3) in partially observable settings, which is central to this thesis, as well as for analyzing the properties of learned models and representations (Section 2.5 and Chapter 5).

2.1.2 Comparing Distributions

We now introduce several ways to compare probability distributions. The KL divergence $D_{\mathrm{KL}}(p \,\|\, q)$ measures how much information is lost when q is used to approximate p, while the total variation distance $d_{TV}(p, q)$ measures the maximum difference in assigned probability mass between the two distributions.

Kullback–Leibler Divergence [Kullback and Leibler, 1951]

Let p and q be distributions on $\mathcal{X}$. The Kullback–Leibler (KL) divergence is

$$D_{\mathrm{KL}}(p \parallel q) = \mathop{\mathbb{E}}_{p}\left[\log \frac{p(x)}{q(x)}\right].$$

This divergence is always nonnegative, and $D_{\mathrm{KL}}(p \parallel q) = 0$ if and only if $p = q$ almost everywhere. KL is not symmetric and does not satisfy the triangle inequality.

Total Variation Distance [Dudley, 2018]

The total variation (TV) distance is defined for any two probability measures p and q on $\mathcal{X}$ by

$$d_{TV}(p,\, q) = \sup_{A \in \Sigma(\mathcal{X})} |p(A) - q(A)|.$$

Equivalently, it can be expressed as

$$d_{TV}(p,\, q) = \frac{1}{2} \int_{\mathcal{X}} |p(x) - q(x)| \, \mathrm{d}\mu(x),$$

which reduces in the discrete case to

$$d_{TV}(p,\, q) = \frac{1}{2} \sum_{x \in \mathcal{X}} |p(x) - q(x)|.$$

TV is closely related to the Kullback–Leibler (KL) divergence: both quantify differences between probability distributions, but TV is symmetric and bounded between 0 and 1, while KL is asymmetric and can be unbounded. The two are linked by the Pinsker inequality (Eq. (2.1)).

Inequalities: Pinsker and Jensen

Pinsker's inequality [Csiszár and Körner, 2011] relates TV and KL:

$$d_{TV}(p,\, q) \leq \sqrt{\frac{1}{2} D_{\mathrm{KL}}(p \parallel q)} \tag{2.1}$$

Jensen's inequality [Jensen, 1906] states that for any convex function $f : \mathbb{R} \to \mathbb{R}$ and random variable X,

$$f\big(\mathop{\mathbb{E}}_{X \sim p}[X]\big) \leq \mathop{\mathbb{E}}_{X \sim p}[f(X)].$$

KL divergence and TV distance, along with these inequalities, are fundamental tools to establish the theoretical guarantees presented in Chapter 5.

2.1.3 Optimal Transport and Wasserstein Distance

Optimal transport (OT) [Villani, 2009] provides a framework for comparing probability distributions by considering the cost of transforming one distribution into another. This is particularly useful for representation learning as as developed further in Section 2.5.2 and applied in Chapter 5.

Couplings and Optimal Transport

Let p and q be probability measures on $\mathcal{X}$. A *coupling* of p and q is a joint distribution γ on $\mathcal{X} \times \mathcal{X}$ such that the marginals satisfy

$$\gamma(A \times \mathcal{X}) = p(A), \quad \gamma(\mathcal{X} \times B) = q(B) \quad \text{for all } A, B \in \Sigma(\mathcal{X}).$$

The set of such couplings is denoted $\Gamma(p, q)$. Given a measurable cost function $c : \mathcal{X} \times \mathcal{X} \to \mathbb{R}_{\geq 0}$, the optimal transport cost is

$$D(p, q)_c = \inf_{\gamma \in \Gamma(p,q)} \int_{\mathcal{X} \times \mathcal{X}} c(x, y) \, \mathrm{d}\gamma(x, y).$$

Wasserstein Distance [Kantorovich, 1942]

When $c(x, y) = d(x, y)$ is a metric on $\mathcal{X}$, the optimal transport cost defines the Wasserstein distance of order 1:

$$W_1(p, q) = \inf_{\gamma \in \Gamma(p,q)} \int_{\mathcal{X} \times \mathcal{X}} d(x, y) \, \mathrm{d}\gamma(x, y).$$

Alternatively, by Kantorovich duality, this admits the dual formulation:

$$W_1(p, q) = \sup_{f \in \mathcal{L}ipsch_d^1} \left(\mathop{\mathbb{E}}_{p} [f(x)] - \mathop{\mathbb{E}}_{q} [f(x)] \right),$$

where $\mathcal{L}ipsch_d^1$ is the set of 1-Lipschitz functions. K-Lipschitz functions are defined as follows:

$$\mathcal{L}ipsch_d^K = \{ f : \mathcal{X} \to \mathbb{R} \mid |f(x) - f(y)| \leq K d(x, y) \text{ for all } x, y \in \mathcal{X} \}. \tag{2.2}$$

This dual view plays a central role in learning representations for distributions and defining bisimilarity metrics (Section 2.5.2).

Wasserstein distances have recently become popular in machine learning, for example in Wasserstein GANs [Arjovsky et al., 2017a] and Wasserstein auto-encoders [Tolstikhin et al., 2018], and we will later leverage them to define latent belief representations in Chapter 5.

Together, these concepts form the mathematical tools that will resurface throughout this dissertation.

2.2 Markov Decision Processes

In this section we introduce Markov Decision Processes (MDPs) [Bellman, 1957], which model sequential decision-making in stochastic environments under the assumption of full observability. MDPs provide the formal foundation for more complex settings such as Partially Observable Markov Decision Processes (POMDPs) [Kaelbling et al., 1998], which we cover in Section 2.3.

2.2.1 Fundamentals

Definition 2.1: Markov Decision Process (MDP)

An MDP $\mathcal{M}$ is defined as a tuple $\langle \mathcal{S}, \mathcal{A}, \mathbf{P}, \mathcal{R}, s_I, \gamma \rangle$, where:

- $\mathcal{S}$ is a (finite) set of states.
- $\mathcal{A}$ is a (finite) set of actions.
- $\mathbf{P} : \mathcal{S} \times \mathcal{A} \to \Delta_{\mathcal{S}}$ is the transition function, where $\mathbf{P}(s' \mid s, a)$ gives the probability of transitioning to state s' after taking action a in state s.
- $\mathcal{R} : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ is the reward function, assigning the reward for taking action a in state s.
- $s_I \in \Delta_{\mathcal{S}}$ is the initial state distribution.
- $\gamma \in [0, 1)$ is the discount factor, which determines the relative importance of future rewards.

A fundamental assumption in MDPs is the *Markov property*, which states that the environment is *memoryless*: the future state depends only on the current state and action, not on the past.

Property 2.1: Markov Property [Puterman, 1990]

Let $(s_i, a_i)_{i \leq t} = (s_0, a_0, \dots, s_t, a_t)$ be a trajectory prefix. The Markov property asserts:

$$\mathbb{P}(s_{t+1} \mid s_0, a_0, \dots, s_t, a_t) = \mathbf{P}(s_{t+1} \mid s_t, a_t)$$

In an MDP, the agent uses a policy π that maps states to (distributions over) actions to interact with the environment. The Property 2.1 justifies using a policy that conditions only on the current state.

Definition 2.2: Policy

A policy π is a mapping from states to distributions over actions, $\pi : \mathcal{S} \to \Delta_{\mathcal{A}}$. If it depends on the time step t, we denote it as π_t.
A *stationary policy* is one that does not depend on time, i.e., $\pi_t = \pi$ for all t.
A *deterministic policy* is a special case where the action is chosen deterministically, i.e., $\pi : \mathcal{S} \to \mathcal{A}$.

A policy, together with the MDP, induces a distribution over trajectories. We denote a trajectory as $\tau = (s_t, a_t, r_t)_{t=0}^{T-1}$, where T is the (possibly infinite) time horizon. The *return* of a trajectory τ is the cumulative discounted reward:

$$R(\tau) = \sum_{t=0}^{T-1} \gamma^t r_t$$

In the infinite-horizon setting, $\gamma < 1$ ensures that the return is finite. The discount factor γ determines how much future rewards are valued compared to immediate ones: as $\gamma \to 0$, the agent becomes more myopic; as $\gamma \to 1$, it becomes more farsighted. Alternatively, when γ is interpreted via geometric horizon sampling, $1-\gamma$ can be viewed as the probability of an episode terminating at each timestep.

We now define key value functions used to evaluate policies:

Definition 2.3: Value, Q-function, and Advantage Function

The value function $V^\pi(s)$ of a policy π in state s is the expected return when starting from state s and following policy π.

$$V^\pi(s) = \mathop{\mathbb{E}}_{\pi}\left[\sum_{t=0}^{\infty} \gamma^t r_t \mid s_0 = s\right]$$

The Q-function $Q^\pi(s, a)$ is the expected return when starting from state s, taking action a, and then following policy π.

$$Q^\pi(s, a) = \mathop{\mathbb{E}}_{\pi}\left[\sum_{t=0}^{\infty} \gamma^t r_t \mid s_0 = s, a_0 = a\right]$$

The advantage function $A^\pi(s, a)$ measures how much better taking action a in state s is compared to the average action under policy π.

$$A^\pi(s, a) = Q^\pi(s, a) - V^\pi(s)$$

The value function, Q-function, and advantage function are core components of sequential decision-making. They allow to evaluate the quality of states and actions under a given policy, and to compare different policies.

Definition 2.4: Bellman Equations [Bellman, 1957]

The value and Q-functions satisfy recursive equations known as the Bellman equations:

$$V^{\pi}(s) = \mathop{\mathbb{E}}_{a\sim\pi(s)}\left[r(s,a) + \gamma \mathop{\mathbb{E}}_{s'\sim\mathbf{P}(\cdot|s,a)} V^{\pi}\left(s'\right)\right]$$
$$Q^{\pi}(s,a) = r(s,a) + \gamma \mathop{\mathbb{E}}_{s'\sim\mathbf{P}(\cdot|s,a)} V^{\pi}\left(s'\right)$$

The goal in MDPs is to find a policy that maximizes expected return from every state. We call such a policy *optimal*, denoted π^*:

$$\pi^* \in \arg\max_{\pi} V^{\pi}(s) \quad \forall s \in \mathcal{S}$$

Property 2.2: Existence of Optimal Policy [Bellman, 1957]

In infinite-horizon discounted MDPs with finite state and action spaces, there exists at least one deterministic stationary policy π^* that is optimal for all states.

The value and Q-functions of the optimal policy π^* are called the *optimal value function* $V^{\star}(s)$ and the *optimal Q-function* $Q^{\star}(s,a)$, respectively. They satisfy the Bellman optimality equations, which are similar to the Bellman equations but maximize over actions instead of averaging:

Definition 2.5: Bellman Optimiality Equation [Bellman, 1957]

The optimal value function $V^{\star}(s)$ and the optimal Q-function $Q^{\star}(s,a)$ satisfy the Bellman optimality equations:

$$V^{\star}(s) = \max_{a\in\mathcal{A}}\left(r(s,a) + \gamma \mathop{\mathbb{E}}_{s'\sim\mathbf{P}(\cdot|s,a)} V^{\star}\left(s'\right)\right)$$
$$Q^{\star}(s,a) = r(s,a) + \gamma \mathop{\mathbb{E}}_{s'\sim\mathbf{P}(\cdot|s,a)} \max_{a'\in A} Q^{\star}\left(s',a'\right)$$

These equations define a fixed-point problem: the optimal value function $V^\star$ is the unique fixed point of the Bellman optimality operator $\mathcal{B}$ [Bellman, 1957]:

$$\mathcal{B}V(s) = \max_{a \in \mathcal{A}} \left(r(s,a) + \gamma \mathop{\mathbb{E}}_{s' \sim \mathbf{P}(\cdot|s,a)} V\left(s'\right) \right) \tag{2.3}$$

This fixed-point characterization forms the basis for dynamic programming methods and many RL algorithms.

> *Remark 2.1: Existence of Stationary Distribution*
>
> In the infinite-horizon setting, many reinforcement learning algorithms rely on the existence of a stationary distribution over state-action pairs induced by a fixed policy. This is often ensured by assuming that the MDP is *ergodic*, i.e., that under any policy, for all visited states the agent will eventually revisits them with non-zero probability in an aperiodic manner. This means that the Markov chain induced by the policy is irreducible and aperiodic, allowing for convergence to a unique stationary distribution over state-action pairs. For example, this can be guaranteed by modeling the environment as *episodic*: having a designated reset state that is always eventually reached and transitions to the initial distribution. In this case, the induced Markov chain becomes ergodic or can be made ergodic without loss of generality [Huang, 2020], and a stationary distribution ξ_π over state-action pairs exists.

2.2.2 Planning in MDPs

In this section, we describe classic algorithms for planning in MDPs, where the agent has full access to the model of the environment (i.e., transition and reward functions). We distinguish between two settings: *offline planning*, where the agent computes a policy before interacting with the environment, and *online planning*, where planning is interleaved with decision-making moddeling the case where the agent is given some compute budget before each action. We focus on value iteration and policy iteration [Puterman, 1990] for the offline case, and introduce Monte Carlo Tree Search (MCTS) [Coulom, 2007] as a prominent online planning algorithm.

Value Iteration and Policy Iteration

Value iteration [Bellman, 1957] (Algorithm 1) and policy iteration [Howard, 1960] are foundational algorithms for solving MDPs, and many modern reinforcement learning and planning algorithms (including the contributions of this dissertation) are built upon its principles.

Value iteration applies the Bellman optimality operator (Eq. (2.3)) iteratively to the value function until convergence. Once converged, the optimal policy can be recovered by acting greedily with respect to the value function.

Algorithm 1: Value Iteration

Input: MDP $\mathcal{M} = \langle \mathcal{S}, \mathcal{A}, \mathbf{P}, \mathcal{R}, s_I, \gamma \rangle$, convergence threshold $\epsilon > 0$
Output: V
Initialize $V(s) \leftarrow 0$ for all $s \in \mathcal{S}$, and $\Delta \leftarrow \infty$
while $\Delta \geq \epsilon$
  $V_{\text{old}} \leftarrow V$
  foreach $s \in \mathcal{S}$
    $V(s) \leftarrow \max_{a \in \mathcal{A}} \left(\mathcal{R}(s, a) + \gamma \sum_{s' \in \mathcal{S}} \mathbf{P}(s' \mid s, a) V_{\text{old}}(s') \right)$ *// Bellman operator*
  $\Delta \leftarrow \| V - V_{\text{old}} \|_\infty$
return V

Policy iteration alternates between two steps: policy evaluation, where the value function for the current policy is computed, and policy improvement, where the policy is greedily updated with respect to the value function.

Both algorithms are guaranteed to converge to the optimal value function and an associated optimal policy. Value iteration is typically preferred when approximate solutions are sufficient or when one seeks to reduce per-iteration computation time, as each update is relatively inexpensive. Policy iteration, although more computationally demanding per iteration due to the policy evaluation step, can reach the optimal policy in fewer iterations in terms of Bellman updates.

We include the pseudocode for value iteration (Algorithm 1), as it will be relevant for the online planning algorithm introduced in Chapter 4.

As the size of the state and action spaces grows, both algorithms become computationally intensive. Practical implementations often rely on approximations such as function approximation, sampling-based updates, or partial sweeps to scale to large problems [Bertsekas, 2018; Scherrer et al., 2012; Szepesvári and Munos, 2005].

Monte Carlo Tree Search (MCTS)

Monte Carlo Tree Search (MCTS) [Coulom, 2007] is a sample-based planning algorithm that incrementally builds a partial search tree by simulating trajectories from the current state. It is particularly well-suited for large or combinatorial decision spaces, where exact planning is computationally infeasible.

At a high level, MCTS explores action trajectories by balancing exploitation of known good actions with exploration of uncertain ones, typically guided by an upper confidence

bound (UCB, Auer et al. 2002). The widely used UCT [Kocsis and Szepesvári, 2006] algorithm formalizes this trade-off using principled exploration bonuses.

By repeatedly simulating trajectories and updating value estimates based on the outcomes, MCTS can concentrate computational effort on promising parts of the search space. This approach has proven highly effective in challenging domains such as Go, where it was used in combination with learned value and policy networks in AlphaGo [Silver et al., 2016].

2.2.3 Learning in MDPs

So far, we have introduced the formal structure of MDPs and described how an agent can plan when the model is known. In practice, however, the dynamics and rewards are often not accessible to the agent in advance. Before turning to the more general and challenging setting of partial observability in the next section, we first examine how agents can learn to act effectively in MDPs through interaction with the environment. We focus on two main classes of methods: value-based methods, which learn to estimate the value of states and actions, and policy-based methods, which directly optimize policies.

Value-based Methods

Q-learning [Watkins and Dayan, 1992] is a classical value-based method that learns an approximation of the optimal action-value function. Its update rule is derived from the Bellman optimality equation (Definition 2.5). After observing a transition $\langle s, a, r, s' \rangle$, the agent updates its estimate of $Q(s, a)$ as follows:

$$Q(s, a) \leftarrow Q(s, a) + \alpha \left[r + \gamma \max_{a'} Q\left(s', a'\right) - Q(s, a) \right] \tag{2.4}$$

where $\alpha \in (0, 1]$ is the learning rate. The term in brackets is known as the *temporal difference (TD) error*. Under suitable assumptions on the learning rate and exploration, Q-learning converges to the optimal Q-function $Q^\star(s, a)$ in the tabular setting.

However, tabular Q-learning becomes infeasible in high-dimensional or continuous state spaces due to the curse of dimensionality. To address this, *function approximation* is used to generalize across states and actions [Tsitsiklis and Van Roy, 1996]. Deep Q-Networks (DQN) [Mnih et al., 2015] popularized this approach by using deep neural networks to approximate Q-values directly from high-dimensional observations, such as raw pixels.

To improve the stability of training, DQN combined several practical techniques:

- **Target networks**, which maintain a delayed copy of the Q-network to compute more stable targets in the TD update.

- **Experience replay** [Lin, 1992], a buffer D that stores past transitions and enables training on decorrelated minibatches.

To further improve the stability and performance of value-based methods, several refinements have been proposed. One of the key issues in classical Q-learning is the *overestimation bias* introduced by the max operator when computing targets:

$$\delta = r + \gamma \max_{a'} Q\left(s', a'\right)$$

Since the same values are used both to select and evaluate actions, noise or approximation errors can lead to overoptimistic value estimates.

Double Q-learning [van Hasselt, 2010] addresses this issue by decoupling action selection and evaluation in the target computation. In the context of Deep Q-Networks, this leads to the *Double DQN* update rule [van Hasselt et al., 2015], where the action is selected using the current network and evaluated using the target network with parameters θ_-:

$$L(\theta) = \mathop{\mathbb{E}}_{\langle s,a,r,s' \rangle \sim D} \left[\left(r + \gamma Q_{\theta_-} \left(s, \arg\max_{a'} Q_\theta\left(s', a'\right) \right) - Q_\theta(s, a) \right)^2 \right]$$

This modification helps mitigate the upward bias in Q-value estimates, leading to improved stability and performance in practice [Hasselt et al., 2018].

Additionally, architectural changes to the Q-network have been explored. The *Dueling Network Architecture* [Wang et al., 2016] splits the Q-function approximation into two streams: one estimating the state value $V(s)$ and the other estimating the advantage $A(s, a)$. These are combined into a single Q-value by summing the two components. This decomposition encourages better value learning in states where the choice of action does not significantly affect the outcome, such as in terminal or bottleneck states.

In Chapter 3, we build upon the Double and Dueling DQN architectures to develop the LAN algorithm, which extends these techniques to the setting of value-based multi-agent learning under partial observability.

Policy-based Methods

Policy-based methods aim to directly learn a parameterized policy π_θ that maps states to action distributions. These methods are particularly well-suited to environments with high-dimensional or continuous action spaces and when stochastic policies are beneficial.

The learning objective is to maximize the expected return:

$$J(\pi_\theta) = \mathop{\mathbb{E}}_{\tau \sim \pi_\theta} [R(\tau)]$$

The REINFORCE algorithm [Williams, 1992] estimates the gradient of J using Monte Carlo rollouts and applies stochastic gradient ascent:

Theorem 2.1: Policy Gradient Theorem [Sutton et al., 2000]

The gradient of the expected return is:

$$\nabla J(\pi_\theta) = \mathop{\mathbb{E}}_{\tau \sim \pi_\theta} \left[\sum_{t=0}^{\infty} \nabla_\theta \log \pi_\theta(a_t \mid s_t) R(\tau) \right]$$

Since this estimate has high variance, a common variance reduction technique is to subtract a *baseline* $b(s)$ from the return without introducing bias. A typical choice is the state-value function $V^\phi(s)$, which yields the following update:

$$\nabla_\theta J(\pi_\theta) = \mathop{\mathbb{E}}_{\tau \sim \pi_\theta} \left[\sum_{t=0}^{\infty} \nabla_\theta \log \pi_\theta(a_t \mid s_t) \left(R(\tau) - V^\pi(s_t) \right) \right]$$

Bootstrapping the return using a learned value function enables learning from partial trajectories. This gives rise to *actor-critic methods* [Konda and Tsitsiklis, 2000; Mnih et al., 2016], where the policy (actor) and value function (critic) are learned jointly. A common actor-critic loss is:

$$L_\theta = \mathop{\mathbb{E}}_{\langle s,a,r,s' \rangle \sim \pi_\theta} \left[\nabla_\theta \log \pi_\theta(a \mid s) \left(r + \gamma V^\phi(s') - V^\phi(s) \right) \right]$$

The quantity $r + \gamma V^\phi(s') - V^\phi(s)$ is the one-step TD error and serves as an estimate of the advantage function $A^\pi(s, a)$. Other estimates of the advantage function can be used, such as Generalized Advantage Estimation (GAE) [Schulman et al., 2016], which combines multiple steps of bootstrapping to reduce variance further.

Modern actor-critic algorithms improve performance and stability by regularizing the policy update. *Proximal Policy Optimization* (PPO) [Schulman et al., 2017] introduces a clipped surrogate objective that discourages large policy shifts. Actor-critic methods, especially PPO, are widely used in modern deep reinforcement learning due to their robustness, sample efficiency.

Building on the actor-critic framework discussed here, the WBU algorithm in Chapter 5 introduces a principled approach for policy learning in partially observable settings.

In summary, MDPs provide the fully observable foundation on which more complex models are built. While they are often too restrictive for realistic scenarios, they serve as a baseline and a stepping stone toward partially observable settings, which we study in Section 2.3.

2.3 Partially Observable Markov Decision Processes

Many real-world scenarios involve uncertainty about the true state of the environment due to limited sensing or noisy observations. To capture this, *Partially Observable Markov Decision Processes* (POMDPs) [Åström, 1965; Kaelbling et al., 1998] extend MDPs by explicitly modeling partial observability. POMDPs provide the canonical framework for sequential decision-making under uncertainty and form the foundation for the planning and representation learning methods developed in this thesis.

2.3.1 Fundamentals

Definition 2.6: Partially Observable Markov Decision Process (POMDP)

A POMDP is a tuple $\mathcal{P} = \langle \mathcal{M}, \Omega, \mathcal{O} \rangle$, where:

- $\mathcal{M} = \langle \mathcal{S}, \mathcal{A}, \mathbf{P}, \mathcal{R}, s_I, \gamma \rangle$ is the underlying (fully observable) MDP.
- Ω is the set of observations.
- $\mathcal{O} : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to \Delta_\Omega$ is the observation function, where $\mathcal{O}(o \mid s', a)$ is the probability of observing o after taking action a and transitioning to state s'.

In contrast to MDPs, the agent in a POMDP does not observe the true state s_t at time t, but instead receives an observation o_t drawn from $\mathcal{O}(\cdot \mid s_t, a_{t-1})$. Because observations are generally not sufficient to infer the full state, they do not satisfy the Markov property. As a result, optimal decision-making cannot rely solely on the current observation but must instead condition on the full interaction history.

We define the history at time T as $h_T = (o_0, a_0, o_1, a_1, \dots, o_{T-1}, a_{T-1}, o_T)$, and denote the space of all such histories by $\mathcal{H}$.

Definition 2.7: History MDP

Every POMDP $\mathcal{P} = \langle \mathcal{M}, \Omega, \mathcal{O} \rangle$ can be transformed into a *history* MDP $\mathcal{M}_\mathcal{H} = \langle \mathcal{H}, \mathcal{A}, \mathbf{P}_\mathcal{H}, \mathcal{R}_\mathcal{H}, s_{I,\mathcal{H}}, \gamma \rangle$ where:

- $\mathcal{H}$ is the set of all possible histories which is used as the state space.
- $\mathcal{A}$ is the set of actions.
- $\mathbf{P}_\mathcal{H} : \mathcal{H} \times \mathcal{A} \to \Delta_\mathcal{H}$ defines the transition function over histories which depends on the transition and observation functions of the POMDP
- $\mathcal{R}_\mathcal{H} : \mathcal{H} \times \mathcal{A} \to \mathbb{R}$ defines the reward function
- $s_{I,\mathcal{H}}$ is the initial history which corresponds to the initial observation.
- $\gamma \in [0, 1)$ is the same discount factor as in the POMDP.

Similar to the MDP case we can define the value function and Q-function over histories and express their recursive structure via the Bellman equations [Kaelbling et al., 1998].

$$V^\pi(h) = \mathop{\mathbb{E}}_{\tau\sim\pi}\left[R(\tau) \mid h_0 = h\right] = \mathop{\mathbb{E}}_{a\sim\pi(h)}\left[r(h,a) + \gamma \mathop{\mathbb{E}}_{h'\sim\mathbf{P}_\mathcal{H}(\cdot|h,a)} V^\pi\left(h'\right)\right] \tag{2.5}$$

$$Q^\pi(h,a) = \mathop{\mathbb{E}}_{\tau\sim\pi}\left[R(\tau) \mid h_0 = h, a_0 = a\right] = r(h,a) + \gamma \mathop{\mathbb{E}}_{h'\sim\mathbf{P}_\mathcal{H}(\cdot|h,a)} V^\pi\left(h'\right) \tag{2.6}$$

In the history-MDP formulation, each time step extends the history by one observation–action pair, and the number of possible distinct histories grows exponentially with the horizon length. This combinatorial growth makes direct policy representations over histories computationally intractable.

To mitigate this, we seek a sufficient statistic of the history, one that captures all the information necessary for decision-making, a canonical choiche is the *belief state* defined below.

Definition 2.8: Belief State [Åström, 1965; Sondik, 1971]

The belief state $b(s)$ at time t is defined as the probability distribution over states given the history $h = (o_0, a_0, o_1, a_1, \dots, o_t)$:

$$b(s) = \mathbb{P}(s \mid h) \tag{2.7}$$

Property 2.3: Belief Update Rule

Beliefs provide a compact representation of the agent's knowledge and are sufficient for optimal decision-making in any POMDP. Crucially, they can be updated recursively as new actions are taken and observations received. Given the belief b_t at time t, action a_t, and observation o_{t+1}, the updated belief b_{t+1} is given by:

$$b_{t+1}(s_{t+1}) = \frac{\mathbb{E}_{s_t\sim b_t}\mathbf{P}(s_{t+1} \mid s_t, a_t)\mathcal{O}(o_{t+1} \mid s_{t+1}, a_t)}{\mathbb{E}_{\tilde{s}_t\sim b_t}\mathbb{E}_{\tilde{s}_{t+1}\sim\mathbf{P}(\cdot|\tilde{s}_t,a_t)}\mathcal{O}(o_{t+1} \mid \tilde{s}_{t+1}, a_t)} \tag{2.8}$$

This recursive update allows us to represent the POMDP as an equivalent fully observable MDP over the belief space, known as the *Belief MDP*.

Definition 2.9: Belief MDP

The Belief MDP induced by a POMDP is a tuple $\mathcal{M}_\mathcal{B} = (\mathcal{B}, \mathcal{A}, \mathbf{P}_\mathcal{B}, \mathcal{R}_\mathcal{B}, b_I, \gamma)$, where:

- $\mathcal{B}$ is the set of belief states (probability distributions over $\mathcal{S}$).
- $\mathcal{A}$ is the set of actions.

- $\mathbf{P}_{\mathcal{B}} : \mathcal{B} \times \mathcal{A} \to \Delta_{\mathcal{B}}$ defines the belief update via the observation and transition models.
- $\mathcal{R}_{\mathcal{B}}(b, a) = \sum_{s \in \mathcal{S}} b(s)\mathcal{R}(s, a)$ is the expected reward under belief b.
- b_I is the initial belief, typically the prior distribution over initial states.
- $\gamma \in [0, 1)$ is the discount factor.

While belief states are not always the most efficient representation for specific tasks, they are universally applicable and sufficient for optimality in any POMDP. Alternative representations—such as finite-state controllers, recurrent neural networks, or learned latent states—may offer computational advantages in practice, but they typically lack the generality and interpretability of belief-based methods.

Solving POMDPs is computationally challenging. For finite-horizon problems, finding an optimal policy is PSPACE-complete [Papadimitriou and Tsitsiklis, 1987], and for infinite-horizon settings, it is undecidable in general [Madani et al., 1999]. This intractability arises from the uncountable belief space and the combinatorially growing set of possible histories.

2.3.2 Planning in POMDPs

Planning in POMDPs entails computing a policy that maximizes expected return under partial observability. Unlike in MDPs, where planning occurs over a discrete state space, POMDP planning operates over the belief space, which is continuous—even when the underlying state space is finite. This significantly increases the computational complexity of planning.

Property 2.4: Piecewise Linearity of the Value Function [Sondik, 1971]

In a finite-horizon POMDP, the optimal value function at timestep t is piecewise linear and convex over the belief space. That is, there exists a finite set of α-vectors Γ_t such that:

$$V_t^\star(b) = \max_{\alpha \in \Gamma_t} \alpha^\top b \tag{2.9}$$

Offline Planning

Offline planning in POMDPs computes a policy before deployment by reasoning over the belief space. As in the fully observable setting (Section 2.2.2), value iteration can be applied here as well, but instead of updating a scalar value for each state, it updates a set of α-vectors that represent the piecewise-linear and convex value function over beliefs (see Property 2.4). In principle, this representation is exact in the finite-horizon case [Sondik,

1978]. However, the number of α-vectors grows exponentially with the horizon, making exact belief-space value iteration impractical for all but very small problems.

To address this, a family of approximate algorithms known as *Point-Based Value Iteration (PBVI)* methods have been developed [Pineau et al., 2003; Spaan and Vlassis, 2005]. These methods approximate the value function using a finite set of representative belief points sampled from reachable regions of the belief space. They iteratively apply a point-based backup operator to update value estimates at those beliefs, using a restricted subset of α-vectors. The use of sampled beliefs enables scalability to moderately sized problems.

One notable offline point-based planners is *SARSOP* [Kurniawati et al., 2009], which focuses sampling on the *optimally reachable* region of the belief space, significantly reducing unnecessary backups and accelerating convergence. SARSOP maintains lower and upper bounds on the value function and refines them during planning.

Online Planning Basics

Online planning methods compute a policy on-the-fly by building a lookahead search tree from the current belief. This approach is particularly well suited to large-scale environments where offline planning is computationally prohibitive.

Two main families of online POMDP planners are:

- **Heuristic search tree algorithms** [Ross et al., 2008], which use value bounds and admissible heuristics to guide tree expansion.
- **Monte Carlo sampling-based algorithms**, which approximate the tree using random rollouts and sampling-based belief updates.

Heuristic Search Tree Algorithms These methods, such as *HSVI* [Smith and Simmons, 2004] or *AEMS* [Ross et al., 2007], perform exact belief updates along the tree and select nodes to expand using heuristics that aim to reduce uncertainty at the root belief. By maintaining both upper and lower bounds on the value function, they provide anytime performance guarantees and can focus computation where it is most impactful. While exact belief updates can be costly in high-dimensional spaces, these algorithms are particularly appealing when accurate value bounds are critical for decision making. In Chapter 4, we build on AEMS and adapt it to a setting where the agent can pay a cost to access the full state of the environment.

Monte Carlo sampling-based algorithms *Partially Observable Monte Carlo Planning (POMCP)* [Silver and Veness, 2010] is a sample-based online planner that combines MCTS (Section 2.2.2) with particle filtering. Rather than maintaining an explicit belief representation, POMCP represents beliefs implicitly via a set of particles (samples of possible states) and builds a search tree rooted at the current belief. POMCP is effective

in large domains with a generative model, as it avoids costly explicit belief updates and handles high-dimensional state spaces and long horizons. Other notable algorithm include DESPOT [Somani et al., 2013], which extends POMCP with a heuristic search tree structure.

2.3.3 Learning in POMDPs

In the learning setting, the agent does not have access to the full model of the environment, including the transition, reward, and observation functions. This rules out tracking belief and requires learning policies directly from experience under partial observability.

The agent must therefore condition its decisions on the full interaction history, which consists of sequences of observations and actions. Since the history space grows unbounded with time, representing or learning from raw histories directly is impractical. A common approach is to use a *Recurrent Neural Network (RNN)* [Elman, 1990] to summarize the history into a latent state.

RNNs process sequences by maintaining a hidden state that is updated at each time step based on the current input and the previous hidden state. This makes them a natural choice for modeling agents under partial observability, as they can serve as function approximators over the space of histories. More advanced variants such as *Gated Recurrent Units (GRUs)* [Cho et al., 2014] and *Long Short-Term Memory (LSTM)* networks [Hochreiter and Schmidhuber, 1997] are widely used in practice due to their ability to mitigate the vanishing and exploding gradient problems during backpropagation through time (BPTT).

The goal is that the RNN learns an internal representation that serves as a sufficient statistic of the history for decision-making—analogous to the belief state in classical POMDP solutions. However, unlike in supervised learning where targets are known, in reinforcement learning it is unclear what features the agent should extract to support good control. This makes the problem substantially more difficult.

Value-based and policy-based RL methods designed for MDPs can be adapted to POMDPs by inserting a recurrent layer into the network architecture. For example:

- **DRQN (Deep Recurrent Q-Network)** [Hausknecht and Stone, 2015] extends DQN by adding an LSTM between the convolutional feature extractor and the output layer.
- **Recurrent Actor-Critic** algorithms similarly maintain hidden states across time and optimize both a recurrent policy (actor) and a value estimator (critic).

As RNNs must operate on temporally contiguous sequences to capture meaningful memory, this breaks the i.i.d. assumptions typically relied upon in minibatch stochastic

gradient descent (SGD) and leads to highly correlated updates, which can destabilize training.

Despite these challenges, recurrent architectures have shown strong empirical performance in some partially observable domains, particularly in visual control, navigation, and games [Hausknecht and Stone, 2015; Heess et al., 2015].

Beyond the addition of RNNs to MDP methods, more structured approaches have been proposed to approximate Bayesian inference in POMDPs. One such method is *Deep Variational Reinforcement Learning (DVRL)* [Igl et al., 2018], which we compare against in Chapter 5. It extends the A2C algorithm [Mnih et al., 2016] by integrating a learned belief model based on variational inference and particle filtering. DVRL uses an RNN to encode sequences and maintain a posterior over latent states, trained jointly with the policy using auxiliary objectives based on the Evidence Lower Bound (ELBO; Kingma et al., 2013), a variational inference criterion that jointly encourages accurate reconstruction of observations and adherence to a learned latent dynamics model.

While DVRL aims to learn belief-like representations, it assumes factorized Gaussian beliefs and lacks theoretical guarantees regarding the quality of its posterior estimates. Nevertheless, it highlights a promising direction toward bridging model-free learning with approximate Bayesian filtering, and has demonstrated improved robustness in partially observable control tasks.

POMDPs thus formalize decision-making under uncertainty by introducing the belief as a sufficient statistic of the history. This formalism is central to this thesis: it underpins our online planning approach in Chapter 4 and provides the basis for our model-based reinforcement learning method in Chapter 5.

2.4 Multi-Agent with Partial Observability

Many real-world decision-making problems are inherently multi-agent in nature. Applications such as distributed sensor networks [Mihaylov et al., 2010], wildlife conservation [Xu et al., 2020], and orbital debris removal [Klima et al., 2018] involve multiple autonomous agents operating in large-scale environments where centralized control is impractical or infeasible. In such settings, each agent typically has access only to local, partial observations of the environment and must coordinate through interaction alone. This makes partial observability more severe than in the single-agent case, since agents must also reason about the behavior of other learning agents, which may change over time. This challenge is known as the *moving target problem* [Tuyls and Weiss, 2012] and is discussed further in Chapter 3.

In this thesis, we focus exclusively on *fully cooperative* settings, where all agents share a common reward function and work towards the same goal. The standard decision-theoretic model for such problems is the *Decentralized Partially Observable Markov*

Decision Process (Dec-POMDP) [Oliehoek et al., 2008; Oliehoek and Amato, 2016], which extends the POMDP framework (Section 2.3) to multiple agents.

Definition 2.10: Dec-POMDP

A Dec-POMDP is defined as a tuple $\mathcal{D} = \langle \mathcal{N}, \mathcal{S}, \boldsymbol{\mathcal{A}}, \Omega, \mathbf{P}, \mathcal{O}, \mathcal{R}, s_I, \gamma \rangle$ where:

- $\mathcal{N}$ is a finite set of agents,
- $\mathcal{S}$ is the set of environment states,
- $\boldsymbol{\mathcal{A}} = \bigtimes_{i \in \mathcal{N}} \mathcal{A}_i$ is the joint action space, with $\mathcal{A}_i$ the action space of agent i,
- $\Omega = \bigtimes_{i \in \mathcal{N}} \Omega_i$ is the joint observation space, with Ω_i the observation space of agent i,
- $\mathbf{P} : \mathcal{S} \times \boldsymbol{\mathcal{A}} \to \Delta_{\mathcal{S}}$ is the state transition function,
- $\mathcal{O} : \mathcal{S} \times \boldsymbol{\mathcal{A}} \to \Delta_{\Omega}$ is the joint observation function,
- $\mathcal{R} : \mathcal{S} \times \boldsymbol{\mathcal{A}} \to \mathbb{R}$ is the shared reward function.

At each time step, every agent i receives an individual observation $o_i \in \Omega_i$ and selects an action $a_i \in \mathcal{A}_i$. The environment transitions to a new state according to $\mathbf{P}$, produces a shared reward via $\mathcal{R}$, and generates new observations according to $\mathcal{O}$. Each agent i maintains an individual history $h_i \in \mathcal{H}_i \equiv (\Omega_i \times \mathcal{A}_i)^*$, and policies π_i map these histories to action distributions. Because agents cannot observe the global state or each other's private histories, optimal coordination requires learning effective decentralized policies from local information.

A common strategy to alleviate these challenges is to exploit privileged information during training, such as access to the state, while still enforcing decentralized execution. This approach is known as *centralized training with decentralized execution* (CTDE).

Centralized Training with Decentralized Execution (CTDE)

As mentioned above, a widely adopted paradigm in cooperative multi-agent reinforcement learning is *centralized training with decentralized execution* (CTDE) [Bernstein et al., 2002; Lowe et al., 2017; Foerster et al., 2018]. The key idea is to relax the partial observability constraint during training: while each agent is restricted to using only its local information at execution time, additional information such as the full environment state, other agents' observations and actions, or aggregated global statistics can be made available during training to improve learning efficiency, stability, and credit assignment. By leveraging centralized information in this way, CTDE can accelerate convergence, stabilize policy optimization, and facilitate coordination in high-dimensional, partially observable environments, although such benefits are not guaranteed and may depend strongly on the task and training procedure.

This relaxation, however, can introduce a potential pitfall known as the *representation bottleneck*. The critic or value target may be trained using information that the actor (policy network) cannot access at execution time, creating a mismatch that can increase the variance of policy gradient estimates [Lyu et al., 2021]. The severity of this effect depends on the problem structure and the informativeness of the agents' local observations, so whether CTDE is ultimately beneficial is highly setting-dependent.

CTDE is closely aligned with the broader theme of this thesis, which studies how access to the underlying state during learning can improve performance in partially observable environments. In Chapter 3, we build on this paradigm to develop *Local Advantage Networks*, a method that exploits state access during training while ensuring decentralized execution.

2.5 Representation Learning and World Models

Planning and learning in both MDPs and POMDPs face severe scalability and generalization challenges, particularly in high-dimensional or noisy observation spaces common in real-world environments. A central goal in modern reinforcement learning is therefore to learn compact, structured representations that capture the relevant dynamics of the environment while abstracting away irrelevant variability.

This section reviews methods for representation learning and world modeling in both fully and partially observable settings, with an emphasis on how on how principled criteria from theory, such as bisimulation, can guide the construction of behaviorally meaningful representations. These ideas form the basis for our model-based contribution in Chapter 5.

2.5.1 World Models

World models are predictive models of environment dynamics learned in a latent space. These models aim to encapsulate environment transitions and reward functions in a compact representation, enabling agents to plan and act using imagined rollouts. A typical world model consists of:

- an *encoder* $\Phi : \mathcal{S} \to \overline{\mathcal{S}}$, mapping observations to latent states;
- a *latent transition function* $\overline{\mathbf{P}} : \overline{\mathcal{S}} \times \mathcal{A} \to \Delta_{\overline{\mathcal{S}}}$;
- a *latent reward function* $\overline{\mathcal{R}} : \overline{\mathcal{S}} \times \mathcal{A} \to \mathbb{R}$.

One of the earliest influential approaches is the *World Models* framework by Ha and Schmidhuber [2018], which combines a variational autoencoder (VAE) [Kingma et al., 2013] with a recurrent network and a compact controller trained within the learned model. Policies trained in the imagined latent space were shown to transfer effectively

to the true environment. This framework sparked renewed interest in separating environment modeling from control learning.

Subsequent work formalized the objectives of world modeling more rigorously. Gelada et al. [2019] introduced the *DeepMDP* framework, which proposes learning latent state abstractions that are jointly predictive of future dynamics and rewards. Their objective formalizes the desiderata that

1. *Latent states preserve reward equivalence*: if two states yield different expected rewards under any action, they should be mapped to different latent representations. This ensures that reward-relevant distinctions are preserved in the abstraction.

2. *Latent transitions are predictive of future latent states*: the learned representation should allow modeling future dynamics as a Markovian process in the latent space, effectively forming a latent MDP.

Theoretical results show that minimizing the DeepMDP objective bounds the value function error, thereby justifying its use for policy learning.

Building on this foundation, Zhang et al. [2020] propose a method for learning latent representations that are *bisimulation-invariant*, i.e., where distances in the latent space reflect bisimulation distances between states. Instead of relying on pixel-level reconstruction, their objective directly encourages control-equivalent states to have similar embeddings. This approach improves robustness to observation-level distractors and leads to more generalizable agents.

Dreamer [Hafner et al., 2019a, 2021] integrates representation learning with model-based reinforcement learning by training policies entirely from imagined rollouts in a learned latent space. It constructs a compact, stochastic recurrent world model that captures environment dynamics and reward structure.

2.5.2 Bisimulation

Bisimulation [Larsen and Skou, 1991; Givan et al., 2003] formalizes the idea of behavioral equivalence in MDPs: two states are bisimilar if they produce the same reward and transition distributions under all actions.

Definition 2.11: Bisimulation Relation

A relation $\sim$ over states $\mathcal{S}$ is a *bisimulation* if it is an equivalence relation and satisfies the following condition:
For all $s, s' \in \mathcal{S}$, if $s \sim s'$, then for all actions $a \in \mathcal{A}$,

- $\mathcal{R}(s, a) = \mathcal{R}(s', a)$
- $\mathbf{P}([s''] \mid s, a) = \mathbf{P}([s''] \mid s', a) \quad$ for all equivalence classes $[s''] \in \mathcal{S}/\sim$

where $\mathbf{P}([s''] \mid s, a)$ denotes the probability of transitioning into the equivalence class $[s'']$:

$$\mathbf{P}([s''] \mid s, a) = \sum_{s'' \in [s'']} \mathbf{P}(s'' \mid s, a)$$

This concept provides a theoretical foundation for state abstraction, allowing agents to compress their representation space without losing information relevant for control.

One of the key challenges with bisimulation is that it is often too strict for practical applications, as it requires exact equivalence across all actions and transitions and even ε small differences in rewards or transition distributions lead to non-bisimilar states. Bisimilation metrics [Desharnais et al., 2004; Ferns et al., 2004, 2011] provide a more flexible framework by defining a distance between states based on how distinguishable their future behaviors are, rather than requiring exact equivalence.

Definition 2.12: Bisimulation Metric

A *bisimulation metric* $d : \mathcal{S} \times \mathcal{S} \to [0, \infty)$ is a distance function defined recursively by:

$$d(s, s') = \max_{a \in \mathcal{A}} \left[\left| \mathcal{R}(s, a) - \mathcal{R}(s', a) \right| + \gamma \cdot W\big(\mathbf{P}(\cdot \mid s, a), \mathbf{P}(\cdot \mid s', a)\big) \right], \tag{2.10}$$

where W is a distributional distance (typically Wasserstein) measured with respect to d itself.

Bisimulation metrics have become a powerful tool for learning representations that capture control-relevant information. Recent work shows that embeddings trained to reflect these metrics are robust to observation-level noise and generalize across tasks [Zhang et al., 2020; Castro et al., 2021].

2.5.3 Wasserstein MDPs

The notion of bisimulation offers a principled foundation for representation learning, but exact equivalence is often too rigid for practical learning scenarios. Recent work explores softer approximations by aligning latent models with observed dynamics using optimal transport.

Wasserstein Auto-encoded MDPs (WAE-MDPs) [Delgrange et al., 2023] extend the DeepMDP framework [Gelada et al., 2019] by aligning entire trajectories in latent and

observed environments. The key idea is to train a latent MDP $\overline{\mathcal{M}}_\theta$ such that trajectories sampled from it match those observed in the true environment $\mathcal{M}$, using an optimal transport (OT, Section 2.1.3) objective [Villani, 2009]. This OT cost is defined over real trajectories and their reconstructions, encouraging the latent dynamics to preserve the control-relevant structure of the original environment.

To support this, in addition to the standard world model components (encoder, latent transition, and reward function), WAE-MDPs also learn a decoder $\Psi : \overline{\mathcal{S}} \to \mathcal{S}$ that maps latent states back to observations. This decoder is used to compare generated rollouts with real ones in observation space.

A temperature parameter $\lambda \in (0, 1)$ controls the continuity of the latent space, interpolating between smooth and discrete abstractions. In the low-temperature limit ($\lambda \to 0$), the learned model becomes probably approximately *bisimilarly close* to the original MDP .That is, the latent model preserves both the expected returns and the transition dynamics of the original process up to bisimulation metrics.

Summary

This dissertation investigates how access to the hidden state of the environment can be strategically used to improve decision-making in partially observable settings. Such access may be available during training or at execution time under a cost. Our objective is to design algorithms that exploit this access to enhance performance, while ensuring that at execution time they adhere to the partial observability constraints of the environment. The theoretical and algorithmic foundations introduced in this chapter provide the basis for the methods developed in the remainder of the thesis, beginning with the cooperative multi-agent setting.

3

Local Advantage Networks

This chapter is based on the paper *Local Advantage Networks for Multi-Agent Reinforcement Learning in Dec-POMDPs* [Avalos et al., 2023], co-authored with Raphaël Avalos, Mathieu Reymond, Ann Nowé, and Diederik M. Roijers, published in the Transactions on Machine Learning Research (TMLR) in 2023. An earlier version of this work appeared as *Local Advantage Networks for Cooperative Multi-Agent Reinforcement Learning* [Avalos et al., 2022] in the proceedings of the 21st International Conference on Autonomous Agents and MultiAgent Systems (AAMAS 2022) as an extended abstract.

Core Intuition

Independent Q-Learning (IQL) performs well in cooperative multi-agent environments when agents' policies are effectively decoupled and coordination is not required. However, under partial observability and strong interaction, IQL becomes unstable due to nonstationarity and poor credit assignment, as each agent learns from a shared reward while treating other agents as part of the environment. Most centralized training with decentralized execution approaches address this issue by explicitly learning a joint action-value function and imposing a factorization structure to recover decentralized policies.

Local Advantage Networks take a different perspective and instead aim to stabilize IQL directly. The key idea is to retain the independent learning architecture of IQL, but to reinterpret its outputs as local advantage functions rather than standalone Q-values. These local advantages are then combined with a centralized value function, conditioned on the global state and the joint history, to form a structured Q-value proxy for each agent.

The resulting Q-value proxy provides a stable learning target that captures the evolving behavior of other agents without transforming the multi-agent problem into a single-agent one. By replacing the individual Q-values with this proxy inside a standard DQN-style loss, LAN enables all components of the model to be trained jointly in a single backward pass. Decentralized execution is preserved as the centralized value is only used during training.

3.1 Introduction

This chapter presents the first contribution of this thesis, developed in the context of fully cooperative multi-agent reinforcement learning with partial observability (Section 2.4). As mentionned in the Background, fully cooperative multi-agent systems are of practical interest because many real-world domains naturally involve multiple autonomous decision-makers pursuing a shared goal, such as distributed sensor networks [Mihaylov et al., 2010], wildlife conservation [Xu et al., 2020], and orbital debris removal [Klima et al., 2018]. In these settings, centralized control is often infeasible, and agents must act on local observations while coordinating implicitly through interaction.

We adopt the *centralized training with decentralized execution* (CTDE) paradigm (Section 2.4), in which training can exploit privileged information such as access to the environment state or other agents' policies, but execution is restricted to decentralized policies based solely on local information. This paradigm is highly related to the overall theme of the thesis: investigating how targeted access to state can improve performance in partially observable environments.

In the cooperative Partially Observable Multi-Agent RL (PO-MARL) setting considered here, agents aim to optimize a shared team reward while acting independently based on their local information. Two principal difficulties arise:

- **Nonstationarity (Moving Target Problem)**: Each agent learns in the presence of other concurrently learning agents, making the environment non-stationary from any individual agent's perspective [Tuyls and Weiss, 2012]. This removes the standard Markovian assumptions (Property 2.1) required by most single-agent RL methods. Definition 3.1 formalizes this non-stationarity by defining the induced POMDP for each agent, which captures the dynamics of the environment as perceived by that agent.
- **Credit Assignment**: To learn a policy, each agent must determine which of its actions contribute to maximizing the shared return. While in single-agent RL this problem is primarily temporal, due to sparse and delayed rewards, shared rewards in multi-agent settings compound the difficulty, as each agent must also disentangle its own contribution from those of others.

Many CTDE approaches rely on decomposing the joint action-value function into agent-wise utilities, such as in Value Decomposition Networks (VDN) [Sunehag et al., 2018], QMIX [Rashid et al., 2018], and QPLEX [Wang et al., 2021]. These methods typically cast the multi-agent learning problem as a single-agent one, where a factorized architecture over the joint action-value function enables decentralized policies to be recovered at execution time. This design facilitates scalable learning while maintaining tractability, and has proven effective on a range of cooperative benchmarks. However,

the imposed structure can also limit the expressiveness of the joint value function, which motivates continued exploration of alternative formulations.

In contrast to this line of work, we propose a conceptually different approach that avoids joint Q-value factorization. Our method, *Local Advantage Networks (LAN)*, learns for each agent a *local advantage function* that captures the value of executing the best-response policy to the other agents' policies, conditioned only on local information. These local advantages are computed based solely on each agent's local trajectory and do not require the explicit construction of a joint Q-function. From these, LAN derives decentralized policies that correspond to local best-responses, making it better suited to handle the challenges of decentralized coordination.

A key technical contribution is the construction of a *Q-value proxy* for each agent, composed of its local advantage and a centralized value estimate. This hybrid structure leverages centralized information during training to stabilize learning, while preserving decentralized executability. Importantly, the centralized component reuses a shared representation extracted from the agents' hidden states, and its number of parameters remains independent of the number of agents. This design choice contributes to LAN's scalability, as it avoids the growth in parameter count that typically accompanies fully centralized critics.

Moreover, the centralized value leverages privileged access to the full state during training, enabling LAN to more effectively integrate the evolving behavior of other agents and guide the learning of each agent's local advantage function. As a result, LAN's Q-value proxy both mitigates the moving target problem—by adapting more rapidly to policy changes in other agents—and improves multi-agent credit assignment by isolating the contribution of each agent through its local advantage.

LAN's structure imposes no a priori restrictions on the class of decentralized policies it can represent, unlike QMIX, which is limited by monotonicity constraints. Since optimal policies in cooperative settings correspond to joint best-responses, LAN's architecture is naturally aligned with the structure of the solution space.

We evaluate LAN on the StarCraft Multi-Agent Challenge (SMAC) [Samvelyan et al., 2019], a widely used benchmark for cooperative multi-agent reinforcement learning. SMAC offers a diverse suite of challenging tasks that require partial observability, decentralized execution, and varying levels of coordination complexity, making it a natural testbed for evaluating CTDE methods. We compare LAN to established value-based baselines: independent Q-learners [Tan, 1993; Tampuu et al., 2015], VDN, QMIX, and QPLEX, which represent successive advances in centralized training with decentralized execution. Across the 14 benchmark maps of SMAC, LAN matches or exceeds the performance of state-of-the-art methods, especially excelling in scenarios requiring complex coordination. In the most demanding tasks, it learns temporally extended strategies, such as sacrificial decoy maneuvers, to achieve high success rates where existing methods fail. Moreover, LAN achieves these results with a significantly

more compact centralized component, using up to 7× fewer parameters than QPLEX in large-agent settings.

To assess the generality of these results, we also evaluate LAN on a modified version of the simple spread task from the Multi-Agent Particle Environment (MPE) [Mordatch and Abbeel, 2017], which features different dynamics, observability, and coordination structure. LAN consistently outperforms or matches other methods as the number of agents increases, demonstrating its ability to scale and generalize beyond SMAC.

Crucially, the purpose of this contribution is not merely to improve empirical performance, but to present an alternative research direction for cooperative MARL, centered on best-response advantage estimation rather than joint value factorization.

This aligns with the overarching theme of this thesis: leveraging structured access to thehidden state to design scalable and robust learning algorithms in sequential decision-making under uncertainty. In the case of LAN, access to the global state during training enables the construction of a centralized value function that serves as a stable learning signal for each agent's local advantage. This design allows LAN to decouple the complexity of coordinating with evolving teammates from the decentralized policy learning process, illustrating how state access can be used in a principled way to improve learning stability and credit assignment without sacrificing decentralized execution.

3.2 Background

The setting considered in this chapter is that of Decentralized Partially Observable Markov Decision Processes (Dec-POMDPs) [Oliehoek et al., 2008; Oliehoek and Amato, 2016] which is defined in Definition 2.10.

Throughout this chapter, we refer to an agent's individual observation-action history simply as its *history*, and to the full joint history as the *joint history*. To simplify notation, we assume the observation function is deterministic; the extension to stochastic observations is straightforward and does not affect the results presented here.

Given a joint policy $\boldsymbol{\pi}$ whose individual components are conditioned on local histories, we define the centralized value function, Q-function, and advantage function recursively as follows:

$$
\begin{aligned}
V^{\boldsymbol{\pi}}(s, \boldsymbol{h}) &= \mathbb{E}_{\boldsymbol{a}\sim\boldsymbol{\pi}(\cdot|\boldsymbol{h})}\left[\mathcal{R}(s, \boldsymbol{a}) + \gamma \mathbb{E}_{s'\sim \mathrm{P}(\cdot|s,\boldsymbol{a})} V^{\boldsymbol{\pi}}\left(s', \boldsymbol{h}'\right)\right] \\
Q^{\boldsymbol{\pi}}(s, \boldsymbol{h}, \boldsymbol{a}) &= \mathcal{R}(s, \boldsymbol{a}) + \gamma \mathbb{E}_{s'\sim \mathrm{P}(\cdot|s,\boldsymbol{a})} V^{\boldsymbol{\pi}}\left(s', \boldsymbol{h}'\right) \\
A^{\boldsymbol{\pi}}(s, \boldsymbol{h}, \boldsymbol{a}) &= Q^{\boldsymbol{\pi}}(s, \boldsymbol{h}, \boldsymbol{a}) - V^{\boldsymbol{\pi}}(s, \boldsymbol{h})
\end{aligned}
$$

Remark 3.1: Extension to Stochastic Observations

Although we assume deterministic observations for notational simplicity, the definitions above naturally extend to the stochastic case. In this setting, the next joint observation $\boldsymbol{o}' \sim \mathcal{O}(\cdot \mid s', \boldsymbol{a})$ must be sampled to construct the next joint history $\boldsymbol{h}' = (\boldsymbol{h}, \boldsymbol{a}, \boldsymbol{o}')$. The expectation over next observations can then be incorporated into the Bellman updates accordingly.

These functions define the expected future return and action value of a joint policy conditioned on both the underlying state and the agents' joint histories.

In particular, the LAN algorithm introduced in this chapter builds on these centralized quantities to define a proxy Q-value for each agent that enables the learning of decentralized best-response policies from local information. This proxy, composed of a centralized value and a local advantage function, plays a central role in addressing the challenges of nonstationarity and credit assignment introduced in the previous section.

3.3 Related Work

Applying single-agent RL algorithms to Dec-POMDPs, such as Independent Q-Learning (IQL) and Independent Actor-Critic, often results in poor performance due to the moving target and multi-agent credit assignment problems [Tan, 1993; Tampuu et al., 2017; Foerster et al., 2018]. While these methods may perform adequately in stateless normal-form games [Nowé et al., 2012], they fail to scale to temporally extended settings. In particular, the use of a replay buffer (a central component of DQN, Section 2.2.3) exacerbates the moving target problem: as policies evolve, transitions stored in the buffer quickly become outdated and may correspond to unreachable states. Attempts to mitigate this issue through importance sampling or policy fingerprinting yield limited improvements [Foerster et al., 2017]. In contrast, LAN's centralized value function mitigates the moving target problem effectively, enabling stable off-policy learning and competitive performance using experience replay.

COMA [Foerster et al., 2018] and MADDPG [Lowe et al., 2017] introduced the CTDE paradigm to deep MARL by extending single-agent actor-critic methods with centralized critics. These centralized critics improve the value estimates used for policy updates. LAN differs from these approaches in that it is a value-based method, making it significantly more sample-efficient. Furthermore, while LAN also uses a centralized component during training, its role is distinct: rather than directly guiding updates through advantage estimates, the centralized value in LAN stabilizes the learning of local advantage functions that define best-response behavior.

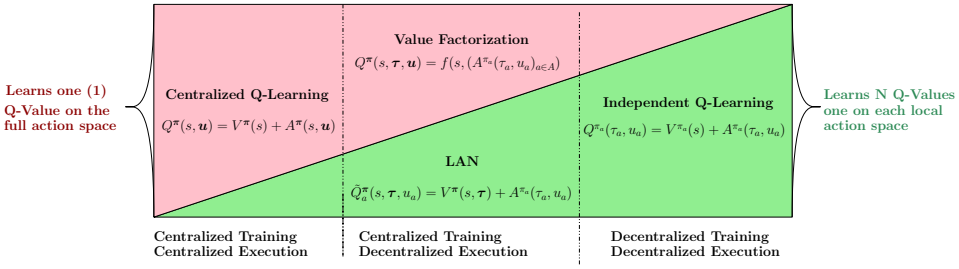


Figure 3.1: Comparison between Centralized Q-Learning, Independent Q-Learning, value factorization methods, and LAN.

Value-based approaches to MARL span two extremes: Centralized Q-Learning (CQL) [Claus and Boutilier, 1998] and Independent Q-Learning (IQL) [Tan, 1993; Tampuu et al., 2017]. CQL maintains a single Q-function over the full joint action space and joint history. While this allows exploiting full observability during training and leads to optimal policies in theory, the approach is neither scalable—due to the exponential growth of the joint action space—nor executable in a decentralized setting. On the other hand, IQL learns individual Q-functions conditioned on each agent's local history and action space. This method is scalable and decentralized but suffers from instability due to nonstationarity. Nevertheless, in environments with weak inter-agent dependencies, IQL can perform competitively.

To balance scalability and coordination, value factorization methods such as VDN [Sunehag et al., 2018], QMIX [Rashid et al., 2018], and QPLEX [Wang et al., 2021] have emerged as prominent CTDE algorithms. These methods learn a centralized Q-function over the joint action space and then factorize it into per-agent utility functions used for decentralized action selection. To ensure consistency between centralized training and decentralized execution, these methods adopt the *individual-global-max* (IGM) principle: the maximizing joint action of the centralized Q-function must match the set of individually maximizing actions for each agent's utility. Enforcing this requires architectural constraints—typically a monotonicity condition on the mixing network, ensuring that each agent's utility contributes positively to the joint value.

VDN implements the simplest form of factorization by summing individual utilities. QMIX improves upon VDN by learning state-dependent, positive mixing weights to relax the linearity constraint. QATTEN [Yang et al., 2020] introduces multi-head attention mechanisms [Vaswani et al., 2017] to compute richer mixing coefficients, and QPLEX further generalizes this approach by transferring the IGM principle to the advantage space instead of the Q-value directly. While QPLEX improves performance over QMIX

on SMAC, it comes at the cost of increased model complexity and roughly double the parameter count.

In contrast, LAN does not factorize a joint Q-function. Instead, it learns an individual Q-value proxy for each agent, based on local best-response advantages and a centralized value term. This avoids the architectural limitations of factorization and enables LAN to represent arbitrary decentralized policies. As shown in Fig. 3.1, while both value factorization methods and LAN operate within the CTDE framework, LAN is structurally closer to IQL in that each agent learns and acts based on its own local Q-function.

Other methods such as MAVEN [Mahajan et al., 2019] and RODE [Wang et al., 2020] propose improvements in multi-agent exploration and structured action spaces, respectively. While both build on QMIX, their core ideas are orthogonal to ours and could be integrated with LAN as well. For this reason, they are not included as direct baselines in our evaluation.

Actor-critic approaches such as MAPPO [Yu et al., 2021] and IPPO [de Witt et al., 2020] have also shown promising results on cooperative MARL benchmarks. However, these methods typically require an order of magnitude more environment interactions (e.g., 10 million vs. 2 million timesteps) and rely on significantly more compute. Additionally, comparing LAN to MAPPO or IPPO is complicated by differences in benchmark configurations: MAPPO modifies the state space, IPPO alters enemy team difficulty, and both tune hyperparameters per map, while LAN and other value-based baselines use a single set across all tasks. Despite these differences, LAN, MAPPO, and IPPO all represent alternative approaches to value factorization, and a dedicated comparative study under standardized settings would be valuable for understanding their respective strengths and limitations.

3.4 Method

3.4.1 Motivation and Overview

In this section, we introduce **Local Advantage Networks (LAN)**, a novel value-based algorithm for cooperative partially observable multi-agent reinforcement learning. In contrast to the dominant trend in MARL, which focuses on factorizing the Q-value of the joint policy Q^{π} into individual utilities, LAN takes a different approach. It learns, for each agent, the advantage of its best-response policy to the policies of the other agents. These local advantage functions are conditioned solely on each agent's own observation-action history, enabling decentralized execution.

The core contribution of LAN is to stabilize the learning of these local advantages using the CTDE paradigm. In particular, LAN leverages centralized access to the environment state and joint observation-action history during training to compute the value of the

joint policy $V^{\boldsymbol{\pi}}$. This centralized value serves as a coordination signal for learning decentralized best-response policies. It also helps reduce partial observability, mitigate the moving target problem, and address the multi-agent credit assignment problem. By combining the local advantages with the centralized value, LAN defines a proxy Q-value for each agent that can be trained using a standard DQN like algorithm (Section 2.2.3, Mnih et al. 2015).

Compared to Q-value factorization methods, LAN exhibits two critical differences:

1. LAN does not learn the Q-value of the joint policy, which is more complex to approximate than the centralized value. Instead, it learns individual Q-value proxies that are easier to optimize.
2. Unlike VDN and QMIX, LAN places no structural restrictions on the class of decentralized policies it can represent. While QPLEX can, in principle, represent the same policies, it does so at the cost of a more complex architecture.

3.4.2 Best-Response Learning and Induced POMDP

> *Remark 3.2: Optimal Dec-POMDP Policies Are Best Responses*
>
> In a Dec-POMDP, when the agents collectively follow an optimal joint policy, each agent's policy must be a best response to the policies of the others. If any agent could unilaterally improve its performance by changing its policy while the others remain fixed, then the joint policy cannot be optimal—since all agents share the same reward.

Based on this observation, LAN focuses on learning decentralized best-response policies, π_i for all agents i. To understand how this can be done, we begin by considering a single agent $i \in \mathcal{N}$, assuming the policies of the other agents, $\boldsymbol{\pi}_{-i}$, are fixed.

> **Definition 3.1: Induced POMDP $\mathcal{P}_i$ [Foerster et al., 2017]**
>
> For each agent i in a Dec-POMDP $\mathcal{D} = \langle \mathcal{N}, \mathcal{S}, \boldsymbol{\mathcal{A}}, \Omega, \mathbf{P}, \mathcal{O}, \mathcal{R}, s_I, \gamma \rangle$ and fixed joint policy $\boldsymbol{\pi}_{-i}$ of the other agents, we define an *induced POMDP* $\mathcal{P}_i = \langle \tilde{\mathcal{S}}, \mathcal{A}_i, \mathbf{P}_i, \Omega_i, \mathcal{R}_i, s_I, \gamma \rangle$, where:
>
> - $\tilde{\mathcal{S}} = \langle \mathcal{S}, \boldsymbol{\mathcal{H}}_{-i} \rangle$ is the environment state extended with the other agents' observation-action histories.
> - $\mathbf{P}_i$ is defined as:
>
> $$\mathbf{P}_i\big(\big\langle s', \boldsymbol{h}'_{-i} \big\rangle \mid \langle s, \boldsymbol{h}_{-i} \rangle, a_i\big) = \mathop{\mathbb{E}}_{\boldsymbol{a}_{-i} \sim \boldsymbol{\pi}_{-i}(\cdot \mid \boldsymbol{h}_{-i})} \mathbf{P}\big(s' \mid s, \langle a_i, \boldsymbol{a}_{-i} \rangle\big) p\big(\boldsymbol{h}'_{-i} \mid \boldsymbol{h}_{-i}, s, s', \boldsymbol{a}_{-i}\big)$$

- $\mathcal{R}_i$ the reward function defined as follows:

$$\mathcal{R}_i(\langle s, \boldsymbol{h}_{-i}\rangle, a_i) = \mathop{\mathbb{E}}_{\boldsymbol{a}_{-i}\sim\boldsymbol{\pi}_{-i}(\cdot|\boldsymbol{h}_{-i})} \mathcal{R}(s, \langle a_i, \boldsymbol{a}_{-i}\rangle)$$

The value, Q-value, and advantage of the induced POMDP $\mathcal{P}_i$ are then defined as follows, where $p(\tilde{s} \mid h_i)$ is the belief over extended states $\tilde{s} \in \tilde{\mathcal{S}}$ given agent i's local history h_i:

$$V^{\pi_i}(h_i) = \mathop{\mathbb{E}}_{a_i\sim\pi_i(\cdot|h_i)} \mathop{\mathbb{E}}_{\tilde{s}\sim p(\cdot|h_i)} \mathop{\mathbb{E}}_{\boldsymbol{a}_{-i}\sim\boldsymbol{\pi}_{-i}(\cdot|\boldsymbol{h}_{-i})} \left[\mathcal{R}(s, \langle a_i, \boldsymbol{a}_{-i}\rangle) + \gamma \mathop{\mathbb{E}}_{s'\sim\ \mathbf{P}(\cdot|s,\langle a_i,\boldsymbol{a}_{-i}\rangle)} V^{\pi_i}\left(h_i'\right)\right] \tag{3.1}$$

$$Q^{\pi_i}(h_i, a_i) = \mathop{\mathbb{E}}_{\tilde{s}\sim p(\cdot|h_i)} \mathop{\mathbb{E}}_{\boldsymbol{a}_{-i}\sim\boldsymbol{\pi}_{-i}(\cdot|\boldsymbol{h}_{-i})} \left[\mathcal{R}(s, \langle a_i, \boldsymbol{a}_{-i}\rangle) + \gamma \mathop{\mathbb{E}}_{s'\sim\ \mathbf{P}(\cdot|s,\langle a_i,\boldsymbol{a}_{-i}\rangle)} V^{\pi_i}\left(h_i'\right)\right] \tag{3.2}$$

$$A^{\pi_i}(h_i, a_i) = Q^{\pi_i}(h_i, a_i) - V^{\pi_i}(h_i) \tag{3.3}$$

As in standard POMDPs, the agent's policy must be conditioned on its entire history, or on a sufficient statistic thereof. In practice, this is often achieved by using a recurrent neural network (RNN) to encode the history into a fixed-size representation without any guarantee to obtain a sufficient statistic as explained in Section 2.3.3. However, the state space of the induced POMDP $\mathcal{P}_i$ is significantly more complex than in the single-agent case, as it includes $\boldsymbol{h}_{-i}$, the joint history of the other agents [Oliehoek and Amato, 2016]. This complexity stems from the fact that the other agents' policies are conditioned on their own histories.

Applying DQN to the induced POMDP $\mathcal{P}_i$ yields a best-response policy for agent i to the fixed policies of the others, as the environment dynamics $\mathbf{P}_i$ are fixed. A natural approach would be to improve agents one at a time. However, this sequential learning fails in environments where optimal behavior requires simultaneous exploration. On the other hand, learning best responses simultaneously—as in Independent Q-Learning (IQL) [Tan, 1993; Tampuu et al., 2015]—leads to instability due to the moving target problem. As the other agents' policies evolve, the environment $\mathcal{P}_i$ perceived by agent i keeps shifting.

Remark 3.3: Challenge of Best-Response Learning in MARL

To learn optimal decentralized policies, agents must improve their behavior in parallel. Yet doing so in isolation causes instability, while sequential updates can prevent the discovery of coordinated strategies. This fundamental tension lies at the heart of cooperative MARL: agents must learn in parallel, but in a coordinated fashion.

3.4.3 LAN's Q-Value Proxy

Q-Value Proxy. LAN enables simultaneous learning of best-response policies while mitigating nonstationarity. Each best-response policy is expressed via a *local advantage function*, $A^{\pi_i}(h_i, a_i)$, conditioned only on agent i's local history, allowing for decentralized execution.

To stabilize the learning of these local advantages, LAN leverages centralized training. It computes a global value function $V^{\boldsymbol{\pi}}$ based on access to the full state and joint history. This leads to a per-agent Q-value proxy:

Definition 3.2: LAN's Q-Value Proxy

For each agent $i \in \mathcal{N}$, LAN defines the proxy Q-function as the sum of the centralized value and the agent's local advantage:

$$\tilde{Q}_i^{\boldsymbol{\pi}}(s, \boldsymbol{h}, a_i) = V^{\boldsymbol{\pi}}(s, \boldsymbol{h}) + A^{\pi_i}(h_i, a_i) \tag{3.4}$$

Although $\tilde{Q}_i^{\boldsymbol{\pi}}$ is not a true Q-function, as the value and advantage do not correspond to the same environment, it can be used to extract decentralized policies. This is because the maximizing action depends only on the local advantage:

Property 3.1

The maximizing action of the Q-value proxy $\tilde{Q}_i^{\boldsymbol{\pi}}$ is the same as that of the local advantage function and the true local Q-value:

$$\arg\max_{a_i} \tilde{Q}_i^{\boldsymbol{\pi}}(s, \boldsymbol{h}, a_i) = \arg\max_{a_i} A^{\pi_i}(h_i, a_i) = \arg\max_{a_i} Q^{\pi_i}(h_i, a_i) \tag{3.5}$$

LAN uses a similar learning scheme as DRQN (Section 2.3.3, Hausknecht and Stone 2015) to jointly learn all Q-value proxies $\tilde{Q}_i^{\boldsymbol{\pi}}$, enabling the simultaneous optimization of local advantages A^{π_i} and the centralized value $V^{\boldsymbol{\pi}}$ under a unified loss. The resulting algorithm is sample-efficient and scalable.

To stabilize learning, LAN uses a target network. The DQN target for agent i is defined as:

$$y_i = r + \gamma \tilde{Q}_i^{\boldsymbol{\pi}}\left(s', \boldsymbol{\tau}', \arg\max_{a_i'} \tilde{Q}_i^{\boldsymbol{\pi}}\left(s', \boldsymbol{\tau}', a_i'\right)\right) \tag{3.6}$$

$$= r + \gamma \left[V^{\boldsymbol{\pi}}\left(s', \boldsymbol{\tau}'\right) + A^{\pi_i}\left(h_i', \arg\max_{a_i'} A^{\pi_i}\left(h_i', a_i'\right)\right)\right] \tag{3.7}$$

Algorithm 2 presents the full pseudo-code for LAN. The algorithm alternates between two phases: (1) collecting transitions by executing synchronized rollouts in the

Algorithm 2: Local Advantage Networks Pseudo Code

Input : Agent set $\mathcal{N}$, replay memory capacity M, target update frequency C, initial exploration rate ϵ

Initialize Replay buffer D with capacity M, centralized value function V with random weights, local advantage functions A^i for each agent $i \in \mathcal{N}$ with RNNs and random weights and target networks $V_t \leftarrow V$, $A_t^i \leftarrow A^i$ for all $i \in \mathcal{N}$

foreach *episode*
- *// Interaction with environment*
- $s, \boldsymbol{o} \leftarrow$ env.reset()
- $h_i \leftarrow \emptyset, a_i \leftarrow \emptyset$ for all $i \in \mathcal{N}$
- $E \leftarrow \emptyset$ *// Initialize empty episode memory*
- **while** *episode not terminated*
  - **foreach** $i \in \mathcal{N}$
    - $h_i \leftarrow \text{RNN}_i(h_i, a_i, oi)$
    - $a_i \leftarrow \pi_\epsilon\left(A^i(h_i, \cdot)\right)$ *// Select action using ϵ-greedy policy*
  - $s', \boldsymbol{o}', r \leftarrow$ env.step($\boldsymbol{a}$) *// Execute joint action*
  - $E \leftarrow E \cup \{(s, \boldsymbol{o}, \boldsymbol{a}, r, s', \boldsymbol{o}')\}$ *// Store transition in episode memory*
  - $s \leftarrow s'$; $\boldsymbol{o} \leftarrow \boldsymbol{o}'$ *// Update current state and joint observation*
- $\mathcal{D}$.store(E) *// Store episode memory E in replay buffer D*
- *// Learning update*
- Sample batch $B \sim \mathcal{D}$
- **foreach** *episode* $e \in B$
  - **for** $t = 1$ **to** $|e|$
    - Unroll RNNs for current and target networks
    - **foreach** $i \in \mathcal{N}$
      - Estimate $\tilde{Q}^{i,t} = V(s^t, \boldsymbol{h}^t) + A^i\left(h_i^t, a_i^t\right)$ *// Current Q-value proxy*
      - Compute TD target $y^{i,t}$ using target networks *// Equation* (3.7)
      - Compute TD error $\delta^{i,t} = (y^{i,t} - \tilde{Q}^{i,t})^2$
  - Update at the same time V and A^i for all $i \in \mathcal{N}$ by minimizing the mean square TD error using gradient descent
  - **if** *episode* $\%\ C = 0$ **then**
    - Update target networks $V_t \leftarrow V$, $A_t^i \leftarrow A^i$ for all $i \in \mathcal{N}$
  - $\epsilon \leftarrow \max(\epsilon \cdot \epsilon_{\text{decay}}, \epsilon_{\min})$ *// Decay exploration rate*

environment using epsilon-greedy policies derived from each agent's local advantage estimates, and (2) updating the Q-value proxies $\tilde{Q}_i^{\boldsymbol{\pi}}$ by sampling episodes from the replay buffer. These updates jointly optimize the local advantage functions and the centralized value function using the DQN update rule described in Section 2.2.3. This design enables all agents to learn their policies in parallel while maintaining coordinated exploration.

Before proceeding, we emphasize that LAN learns greedy policies and we thus consider only deterministic policies throughout.

3.4.4 Correctness of the Q-Value Proxy

We now show that the Q-value proxy (Definition 3.2) is an unbiased estimator of the true local Q-value it approximates (Equation (3.2)). The proof builds on the following lemmas.

Lemma 3.1

Let h_i be a realizable local history for agent i. For any next history h_i' reachable under policy π_i, we can express the transition probability $p(h_i' \mid h_i)$ as:

$$p\big(h_i' \mid h_i\big) = \mathop{\mathbb{E}}_{s,\boldsymbol{h}_{-i}\sim p(\cdot|h_i)} \mathop{\mathbb{E}}_{s'\sim\mathbf{P}(\cdot|s,\boldsymbol{\pi}(\langle \boldsymbol{h}_{-i},h_i\rangle))} p\big(h_i' \mid s', \langle \boldsymbol{h}_{-i}, h_i\rangle, \boldsymbol{\pi}(\langle \boldsymbol{h}_{-i}, h_i\rangle)\big)$$

Proof. We start by decomposing the next joint history $\boldsymbol{h}'$, and local history h_i as tuples containing the new observation, the action and the previous history to obtain the following equalities:

$$p\Big(\boldsymbol{h}' = \big\langle \boldsymbol{o}', \boldsymbol{a}, \tilde{\boldsymbol{h}}\big\rangle \mid s', \boldsymbol{\pi}(\boldsymbol{h}), \boldsymbol{h}\Big) = \delta_{\tilde{\boldsymbol{h}}}(\boldsymbol{h})\delta_{\boldsymbol{a}}(\boldsymbol{\pi}(\boldsymbol{h}))\mathcal{O}\big(\boldsymbol{o}' \mid s', \boldsymbol{\pi}(\boldsymbol{h})\big) \tag{3.8}$$

$$p\Big(h_i' = \big\langle o_i', a_i, \tilde{h}_i\big\rangle \mid s', \boldsymbol{\pi}(\langle \boldsymbol{h}_{-i}, h_i\rangle), \langle \boldsymbol{h}_{-i}, h_i\rangle\Big) = \delta_{\tilde{h}_i}(h_i)\delta_{a_i}(\pi_i(h_i))\mathcal{O}_i\big(o_i' \mid s', \pi_i(h_i)\big) \tag{3.9}$$

$$\begin{aligned}
&p\big(h_i' \mid h_i\big) \\
&= \int_s \int_{s'} \int_{\boldsymbol{h}_{-i}} p\big(h_i' \mid s', \langle \boldsymbol{h}_{-i}, h_i\rangle, \boldsymbol{\pi}(\langle \boldsymbol{h}_{-i}, h_i\rangle), s\big) p\big(s', \boldsymbol{h}_{-i}, \boldsymbol{\pi}(\langle \boldsymbol{h}_{-i}, h_i\rangle), s \mid h_i\big) \mathrm{d}\boldsymbol{h}_{-i}\mathrm{d}s'\mathrm{d}s \\
&\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \text{(law of total probability)} \\
&= \int_s \int_{s'} \int_{\boldsymbol{h}_{-i}} p\big(h_i' \mid s', \langle \boldsymbol{h}_{-i}, h_i\rangle, \boldsymbol{\pi}(\langle \boldsymbol{h}_{-i}, h_i\rangle), s\big) p(s, \boldsymbol{h}_{-i} \mid h_i) \\
&\qquad\qquad \mathbf{P}\big(s' \mid s, \langle \boldsymbol{h}_{-i}, h_i\rangle, \boldsymbol{\pi}(\langle \boldsymbol{h}_{-i}, h_i\rangle)\big) \mathrm{d}\boldsymbol{h}_{-i}\mathrm{d}s'\mathrm{d}s \qquad\qquad \text{(chain rule)} \\
&= \int_s \int_{\boldsymbol{h}_{-i}} p(s, \boldsymbol{h}_{-i} \mid h_i) \int_{s'} \mathbf{P}\big(s' \mid s, \langle \boldsymbol{h}_{-i}, h_i\rangle, \boldsymbol{\pi}(\langle \boldsymbol{h}_{-i}, h_i\rangle)\big)
\end{aligned}$$

$$p\left(h_i' \mid s', \langle \boldsymbol{h}_{-i}, h_i \rangle, \boldsymbol{\pi}(\langle \boldsymbol{h}_{-i}, h_i \rangle), s\right) \mathrm{d}s' \mathrm{d}s \mathrm{d}\boldsymbol{h}_{-i} \qquad \text{(linearity)}$$

$$= \int_s \int_{\boldsymbol{h}_{-i}} p(s, \boldsymbol{h}_{-i} \mid h_i) \mathop{\mathbb{E}}_{s' \sim \mathbf{P}(\cdot \mid s, \boldsymbol{\pi}(\langle \boldsymbol{h}_{-i}, h_i \rangle))} p\left(h_i' \mid s', \langle \boldsymbol{h}_{-i}, h_i \rangle, \boldsymbol{\pi}(\langle \boldsymbol{h}_{-i}, h_i \rangle), s\right) \mathrm{d}s \mathrm{d}\boldsymbol{h}_{-i}$$

(definition of expectation)

$$= \mathop{\mathbb{E}}_{s, \boldsymbol{h}_{-i} \sim p(\cdot \mid h_i)} \mathop{\mathbb{E}}_{s' \sim \mathbf{P}(\cdot \mid s, \boldsymbol{\pi}(\langle \boldsymbol{h}_{-i}, h_i \rangle))} p\left(h_i' \mid s', \langle \boldsymbol{h}_{-i}, h_i \rangle, \boldsymbol{\pi}(\langle \boldsymbol{h}_{-i}, h_i \rangle), s\right)$$

(definition of expectation)

$$= \mathop{\mathbb{E}}_{s, \boldsymbol{h}_{-i} \sim p(\cdot \mid h_i)} \mathop{\mathbb{E}}_{s' \sim \mathbf{P}(\cdot \mid s, \boldsymbol{\pi}(\langle \boldsymbol{h}_{-i}, h_i \rangle))} p\left(h_i' \mid s', \langle \boldsymbol{h}_{-i}, h_i \rangle, \boldsymbol{\pi}(\langle \boldsymbol{h}_{-i}, h_i \rangle)\right)$$

(conditional independence of h_i' and s given s')

□

Lemma 3.2

For any realisable history h_i, that is realisable under the policy π_i, and any next state s' and next joint history $\boldsymbol{h}'$ we have

$$p\left(s', \boldsymbol{h}' \mid h_i\right) = \mathop{\mathbb{E}}_{s, \boldsymbol{h}_{-i} \sim p(\cdot \mid h_i)} \mathbf{P}\left(s' \mid s, \boldsymbol{\pi}(\langle \boldsymbol{h}_{-i}, h_i \rangle)\right) p\left(\boldsymbol{h}' \mid s', \langle \boldsymbol{h}_{-i}, h_i \rangle, \boldsymbol{\pi}(\langle \boldsymbol{h}_{-i}, h_i \rangle)\right)$$

Proof.

$$\begin{aligned}
& p\left(s', \boldsymbol{h}' \mid h_i\right) \\
&= \mathop{\mathbb{E}}_{s, \boldsymbol{h}_{-i} \sim p(\cdot \mid h_i)} p\left(s', \boldsymbol{h}' \mid s, \langle \boldsymbol{h}_{-i}, h_i \rangle\right) && \text{(law of total probability)} \\
&= \mathop{\mathbb{E}}_{s, \boldsymbol{h}_{-i} \sim p(\cdot \mid h_i)} p\left(s' \mid s, \langle \boldsymbol{h}_{-i}, h_i \rangle\right) p\left(\boldsymbol{h}' \mid s', s, \langle \boldsymbol{h}_{-i}, h_i \rangle\right) && \text{(chain rule)} \\
&= \mathop{\mathbb{E}}_{s, \boldsymbol{h}_{-i} \sim p(\cdot \mid h_i)} \mathbf{P}\left(s' \mid s, \boldsymbol{\pi}(\langle \boldsymbol{h}_{-i}, h_i \rangle)\right) p\left(\boldsymbol{h}' \mid s', s, \langle \boldsymbol{h}_{-i}, h_i \rangle, \boldsymbol{\pi}(\langle \boldsymbol{h}_{-i}, h_i \rangle)\right) \\
&= \mathop{\mathbb{E}}_{s, \boldsymbol{h}_{-i} \sim p(\cdot \mid h_i)} \mathbf{P}\left(s' \mid s, \boldsymbol{\pi}(\langle \boldsymbol{h}_{-i}, h_i \rangle)\right) p\left(\boldsymbol{h}' \mid s', \langle \boldsymbol{h}_{-i}, h_i \rangle, \boldsymbol{\pi}(\langle \boldsymbol{h}_{-i}, h_i \rangle)\right) \\
&\qquad \text{(conditional independence of } \boldsymbol{h}' \text{ and } s \text{ given } s', \langle \boldsymbol{h}_{-i}, h_i \rangle, \boldsymbol{\pi}(\langle \boldsymbol{h}_{-i}, h_i \rangle))
\end{aligned}$$

□

We now state and prove the main theoretical result of this section.

Theorem 3.1

For any agent $i \in \mathcal{N}$, realizable local history $h_i \in \mathcal{H}_i$, and any action $a_i \in \mathcal{A}_i$, the Q-value proxy $\tilde{Q}_i^{\boldsymbol{\pi}}$ is an unbiased estimator of the local Q-value Q^{π_i}:

$$\mathop{\mathbb{E}}_{s,\boldsymbol{h}_{-i}\sim p(\cdot|h_i)} \tilde{Q}_i^{\boldsymbol{\pi}}(s, \langle \boldsymbol{h}_{-i}, h_i\rangle, a_i) = Q^{\pi_i}(h_i, a_i)$$

Proof. We fix $i \in \mathcal{N}$, $a_i \in \mathcal{A}_i$, $h_i \in \mathcal{H}_i$

$$\begin{aligned}
&\mathop{\mathbb{E}}_{s,\boldsymbol{h}_{-i}\sim p(\cdot|h_i)} \left[\tilde{Q}_i(s, \langle \boldsymbol{h}_{-i}, h_i\rangle, a_i) - Q^{\pi_i}(h_i, a_i)\right] \\
&\quad = \mathop{\mathbb{E}}_{s,\boldsymbol{h}_{-i}\sim p(\cdot|h_i)} \left[V^{\boldsymbol{\pi}}(s, \langle \boldsymbol{h}_{-i}, h_i\rangle) + A^{\pi_i}(h_i, a_i) - \left(V^{\pi_i}(h_i) + A^{\pi_i}(h_i, a_i)\right)\right] \\
&\quad = \mathop{\mathbb{E}}_{s,\boldsymbol{h}_{-i}\sim p(\cdot|h_i)} \left[V^{\boldsymbol{\pi}}(s, \langle \boldsymbol{h}_{-i}, h_i\rangle) - V^{\pi_i}(h_i)\right] \qquad (3.10)
\end{aligned}$$

By definition we have:

$$\begin{aligned}
V^{\boldsymbol{\pi}}(s, \boldsymbol{h}) &= r(s, \boldsymbol{\pi}(\boldsymbol{h})) + \gamma \mathop{\mathbb{E}}_{s'\sim \mathbf{P}(\cdot|s,\boldsymbol{\pi}(\boldsymbol{h}))} \mathop{\mathbb{E}}_{\boldsymbol{h}'\sim p(\cdot|s',\boldsymbol{\pi}(\boldsymbol{h}),\boldsymbol{h})} V^{\boldsymbol{\pi}}\left(s', \boldsymbol{h}'\right) \\
V^{\pi_i}(h_i) &= \mathop{\mathbb{E}}_{s,\boldsymbol{h}_{-i}\sim p(\cdot|h_i)} \Big[r(s, \boldsymbol{\pi}(\langle \boldsymbol{h}_{-i}, h_i\rangle)) \\
&\qquad + \gamma \mathop{\mathbb{E}}_{s,\boldsymbol{h}_{-i}\sim p(\cdot|h_i)} \mathop{\mathbb{E}}_{s'\sim \mathbf{P}(\cdot|s,\boldsymbol{\pi}(\langle \boldsymbol{h}_{-i}, h_i\rangle))} \mathop{\mathbb{E}}_{h_i'\sim p(\cdot|s',\boldsymbol{\pi}(\langle \boldsymbol{h}_{-i}, h_i\rangle),\langle \boldsymbol{h}_{-i}, h_i\rangle)} V^{\pi_i}\left(h_i'\right)\Big]
\end{aligned}$$

We define Δ_r and Δ_p as follow:

$$\begin{aligned}
\Delta_r(s, \langle \boldsymbol{h}_{-i}, h_i\rangle) &= r(s, \boldsymbol{\pi}(\langle \boldsymbol{h}_{-i}, h_i\rangle)) - \mathop{\mathbb{E}}_{\tilde{s},\tilde{\boldsymbol{h}}_{-i}\sim p(\cdot|h_i)} r\left(\tilde{s}, \boldsymbol{\pi}\left(\left\langle \tilde{\boldsymbol{h}}_{-i}, h_i\right\rangle\right)\right) \\
\Delta_p(s, \langle \boldsymbol{h}_{-i}, h_i\rangle) &= \mathop{\mathbb{E}}_{s'\sim \mathbf{P}(\cdot|s,\boldsymbol{\pi}(\langle \boldsymbol{h}_{-i}, h_i\rangle))} \mathop{\mathbb{E}}_{\boldsymbol{h}'\sim p(\cdot|s',\boldsymbol{\pi}(\langle \boldsymbol{h}_{-i}, h_i\rangle),\langle \boldsymbol{h}_{-i}, h_i\rangle)} V^{\boldsymbol{\pi}}\left(s', \boldsymbol{h}'\right) \\
&\quad - \mathop{\mathbb{E}}_{\tilde{s},\tilde{\boldsymbol{h}}_{-i}\sim p(\cdot|h_i)} \mathop{\mathbb{E}}_{s'\sim \mathbf{P}\left(\cdot|\tilde{s},\boldsymbol{\pi}\left(\left\langle \tilde{\boldsymbol{h}}_{-i}, h_i\right\rangle\right)\right)} \mathop{\mathbb{E}}_{h_i'\sim p\left(\cdot|\tilde{s}',\boldsymbol{\pi}\left(\left\langle \tilde{\boldsymbol{h}}_{-i}, h_i\right\rangle\right),\left\langle \tilde{\boldsymbol{h}}_{-i}, h_i\right\rangle\right)} V^{\pi_i}\left(h_i'\right)
\end{aligned}$$

This allows us to rewrite Eq 3.10 as

$$\begin{aligned}
&\mathop{\mathbb{E}}_{s,\boldsymbol{h}_{-i}\sim p(\cdot|h_i)} \left[\tilde{Q}_i(s, \langle \boldsymbol{h}_{-i}, h_i\rangle, a_i) - Q^{\pi_i}(h_i, a_i)\right] \\
&\quad = \mathop{\mathbb{E}}_{s,\boldsymbol{h}_{-i}\sim p(\cdot|h_i)} \left[\Delta_r(s, \langle \boldsymbol{h}_{-i}, h_i\rangle)\right] + \gamma \mathop{\mathbb{E}}_{s,\boldsymbol{h}_{-i}\sim p(\cdot|h_i)} \left[\Delta_p(s, \langle \boldsymbol{h}_{-i}, h_i\rangle)\right] \qquad (3.11)
\end{aligned}$$

Let's first focus on the first part of the RHS of Equation 3.10.

$$\mathop{\mathbb{E}}_{s,\boldsymbol{h}_{-i}\sim p(\cdot|h_i)} \left[\Delta_r(s, \langle \boldsymbol{h}_{-i}, h_i\rangle)\right]$$

$$= \underset{s,\boldsymbol{h}_{-i}\sim p(\cdot|h_i)}{\mathbb{E}}\left[r(s,\boldsymbol{\pi}(\langle \boldsymbol{h}_{-i},h_i\rangle)) - \underset{\tilde{s},\tilde{\boldsymbol{h}}_{-i}\sim p(\cdot|h_i)}{\mathbb{E}} r\left(\tilde{s},\boldsymbol{\pi}\left(\left\langle \tilde{\boldsymbol{h}}_{-i},h_i\right\rangle\right)\right)\right]$$

$$= \underset{s,\boldsymbol{h}_{-i}\sim p(\cdot|h_i)}{\mathbb{E}} r(s,\boldsymbol{\pi}(\langle \boldsymbol{h}_{-i},h_i\rangle)) - \underset{s,\boldsymbol{h}_{-i}\sim p(\cdot|h_i)}{\mathbb{E}}\ \underset{\tilde{s},\tilde{\boldsymbol{h}}_{-i}\sim p(\cdot|h_i)}{\mathbb{E}} r\left(\tilde{s},\boldsymbol{\pi}\left(\left\langle \tilde{\boldsymbol{h}}_{-i},h_i\right\rangle\right)\right)$$

(linearity of expectation)

$$= \underset{s,\boldsymbol{h}_{-i}\sim p(\cdot|h_i)}{\mathbb{E}} r(s,\boldsymbol{\pi}(\langle \boldsymbol{h}_{-i},h_i\rangle)) - \underset{\tilde{s},\tilde{\boldsymbol{h}}_{-i}\sim p(\cdot|h_i)}{\mathbb{E}} r\left(\tilde{s},\boldsymbol{\pi}\left(\left\langle \tilde{\boldsymbol{h}}_{-i},h_i\right\rangle\right)\right)$$

(second part does not depend on s, $\boldsymbol{h}_{-i}$)

$$= 0$$

Let's now focus on the second part of the RHS of Equation 3.10.

$$\underset{s,\boldsymbol{h}_{-i}\sim p(\cdot|h_i)}{\mathbb{E}}\left[\Delta_p(s,\langle \boldsymbol{h}_{-i},h_i\rangle)\right]$$

$$= \underset{s,\boldsymbol{h}_{-i}\sim p(\cdot|h_i)}{\mathbb{E}}\Bigg[\underset{s'\sim \mathbf{P}(\cdot|s,\boldsymbol{\pi}(\langle \boldsymbol{h}_{-i},h_i\rangle))}{\mathbb{E}}\ \underset{\boldsymbol{h}'\sim p(\cdot|s',\boldsymbol{\pi}(\langle \boldsymbol{h}_{-i},h_i\rangle),\langle \boldsymbol{h}_{-i},h_i\rangle)}{\mathbb{E}} V^{\boldsymbol{\pi}}\left(s',\boldsymbol{h}'\right)$$
$$- \underset{\tilde{s},\tilde{\boldsymbol{h}}_{-i}\sim p(\cdot|h_i)}{\mathbb{E}}\ \underset{s'\sim \mathbf{P}\left(\cdot|\tilde{s},\boldsymbol{\pi}\left(\left\langle \tilde{\boldsymbol{h}}_{-i},h_i\right\rangle\right)\right)}{\mathbb{E}}\ \underset{h_i'\sim p\left(\cdot|\tilde{s}',\boldsymbol{\pi}\left(\left\langle \tilde{\boldsymbol{h}}_{-i},h_i\right\rangle\right),\left\langle \tilde{\boldsymbol{h}}_{-i},h_i\right\rangle\right)}{\mathbb{E}} V^{\pi_i}\left(h_i'\right)\Bigg]$$

$$= \overbrace{\underset{s,\boldsymbol{h}_{-i}\sim p(\cdot|h_i)}{\mathbb{E}}\ \underset{s'\sim \mathbf{P}(\cdot|s,\boldsymbol{\pi}(\langle \boldsymbol{h}_{-i},h_i\rangle))}{\mathbb{E}}\ \underset{\boldsymbol{h}'\sim p(\cdot|s',\boldsymbol{\pi}(\langle \boldsymbol{h}_{-i},h_i\rangle),\langle \boldsymbol{h}_{-i},h_i\rangle)}{\mathbb{E}} V^{\boldsymbol{\pi}}\left(s',\boldsymbol{h}'\right)}^{\text{A}}$$
$$- \underbrace{\underset{s,\boldsymbol{h}_{-i}\sim p(\cdot|h_i)}{\mathbb{E}}\ \underset{\tilde{s},\tilde{\boldsymbol{h}}_{-i}\sim p(\cdot|h_i)}{\mathbb{E}}\ \underset{s'\sim \mathbf{P}\left(\cdot|\tilde{s},\boldsymbol{\pi}\left(\left\langle \tilde{\boldsymbol{h}}_{-i},h_i\right\rangle\right)\right)}{\mathbb{E}}\ \underset{h_i'\sim p\left(\cdot|\tilde{s}',\boldsymbol{\pi}\left(\left\langle \tilde{\boldsymbol{h}}_{-i},h_i\right\rangle\right),\left\langle \tilde{\boldsymbol{h}}_{-i},h_i\right\rangle\right)}{\mathbb{E}} V^{\pi_i}\left(h_i'\right)}_{\text{B}}$$

(linearity of expectation)

$$\text{A} = \underset{s,\boldsymbol{h}_{-i}\sim p(\cdot|h_i)}{\mathbb{E}}\ \underset{s'\sim \mathbf{P}(\cdot|s,\boldsymbol{\pi}(\langle \boldsymbol{h}_{-i},h_i\rangle))}{\mathbb{E}}\ \underset{\boldsymbol{h}'\sim p(\cdot|s',\boldsymbol{\pi}(\langle \boldsymbol{h}_{-i},h_i\rangle),\langle \boldsymbol{h}_{-i},h_i\rangle)}{\mathbb{E}} V^{\boldsymbol{\pi}}\left(s',\boldsymbol{h}'\right)$$

$$= \underset{s,\boldsymbol{h}_{-i}\sim p(\cdot|h_i)}{\mathbb{E}} \int_{s'} \mathbf{P}\left(s' \mid s,\boldsymbol{\pi}(\langle \boldsymbol{h}_{-i},h_i\rangle)\right) \quad \text{(definition of expectation)}$$
$$\int_{\boldsymbol{h}'} p\left(\boldsymbol{h}' \mid s',\boldsymbol{\pi}(\langle \boldsymbol{h}_{-i},h_i\rangle),\langle \boldsymbol{h}_{-i},h_i\rangle\right) V^{\boldsymbol{\pi}}\left(s',\boldsymbol{h}'\right) \mathrm{d}s'\,\mathrm{d}\boldsymbol{h}'$$

$$= \int_{s'}\int_{\boldsymbol{h}'} \underset{s,\boldsymbol{h}_{-i}\sim p(\cdot|h_i)}{\mathbb{E}} \mathbf{P}\left(s' \mid s,\boldsymbol{\pi}(\langle \boldsymbol{h}_{-i},h_i\rangle)\right) p\left(\boldsymbol{h}' \mid s',\boldsymbol{\pi}(\langle \boldsymbol{h}_{-i},h_i\rangle),\langle \boldsymbol{h}_{-i},h_i\rangle\right)$$
$$V^{\boldsymbol{\pi}}\left(s',\boldsymbol{h}'\right) \mathrm{d}s'\,\mathrm{d}\boldsymbol{h}' \quad \text{(linearity)}$$

$$= \int_{s'}\int_{\boldsymbol{h}'} p\left(s',\boldsymbol{h}' \mid h_i\right) V^{\boldsymbol{\pi}}\left(s',\boldsymbol{h}'\right) \mathrm{d}s'\,\mathrm{d}\boldsymbol{h}' \quad \text{(see Lemma 3.2)}$$

$$= \mathop{\mathbb{E}}_{s',\boldsymbol{h}'\sim p(\cdot|h_i)} V^{\boldsymbol{\pi}}\left(s',\boldsymbol{h}'\right) \qquad \text{(definition of expectation)}$$

$$\begin{aligned}
\mathrm{B} &= \mathop{\mathbb{E}}_{s,\boldsymbol{h}_{-i}\sim p(\cdot|h_i)} \mathop{\mathbb{E}}_{\tilde{s},\tilde{\boldsymbol{h}}_{-i}\sim p(\cdot|h_i)} \mathop{\mathbb{E}}_{s'\sim \mathbf{P}\left(\cdot|\tilde{s},\boldsymbol{\pi}\left(\left\langle\tilde{\boldsymbol{h}}_{-i},h_i\right\rangle\right)\right)} \mathop{\mathbb{E}}_{h_i'\sim p\left(\cdot|\tilde{s}',\boldsymbol{\pi}\left(\left\langle\tilde{\boldsymbol{h}}_{-i},h_i\right\rangle\right),\left\langle\tilde{\boldsymbol{h}_{-i}},h_i\right\rangle\right)} V^{\pi_i}\left(h_i'\right) \\
&= \mathop{\mathbb{E}}_{\tilde{s},\tilde{\boldsymbol{h}}_{-i}\sim p(\cdot|h_i)} \mathop{\mathbb{E}}_{s'\sim \mathbf{P}\left(\cdot|\tilde{s},\boldsymbol{\pi}\left(\left\langle\tilde{\boldsymbol{h}}_{-i},h_i\right\rangle\right)\right)} \mathop{\mathbb{E}}_{h_i'\sim p\left(\cdot|\tilde{s}',\boldsymbol{\pi}\left(\left\langle\tilde{\boldsymbol{h}}_{-i},h_i\right\rangle\right),\left\langle\tilde{\boldsymbol{h}_{-i}},h_i\right\rangle\right)} V^{\pi_i}\left(h_i'\right) \\
&\qquad \text{(does not depend on } s,\boldsymbol{h}_{-i}) \\
&= \mathop{\mathbb{E}}_{\tilde{s},\tilde{\boldsymbol{h}}_{-i}\sim p(\cdot|h_i)} \mathop{\mathbb{E}}_{s'\sim \mathbf{P}\left(\cdot|\tilde{s},\boldsymbol{\pi}\left(\left\langle\tilde{\boldsymbol{h}}_{-i},h_i\right\rangle\right)\right)} \int_{h_i'} p\left(h_i' \mid \tilde{s}',\boldsymbol{\pi}\left(\left\langle\tilde{\boldsymbol{h}}_{-i},h_i\right\rangle\right),\left\langle\tilde{\boldsymbol{h}_{-i}},h_i\right\rangle\right) V^{\pi_i}\left(h_i'\right)\,\mathrm{d}h_i' \\
&\qquad \text{(definition of expectation)} \\
&= \int_{h_i'} \mathop{\mathbb{E}}_{\tilde{s},\tilde{\boldsymbol{h}}_{-i}\sim p(\cdot|h_i)} \mathop{\mathbb{E}}_{s'\sim \mathbf{P}\left(\cdot|\tilde{s},\boldsymbol{\pi}\left(\left\langle\tilde{\boldsymbol{h}}_{-i},h_i\right\rangle\right)\right)} p\left(h_i' \mid \tilde{s}',\boldsymbol{\pi}\left(\left\langle\tilde{\boldsymbol{h}}_{-i},h_i\right\rangle\right),\left\langle\tilde{\boldsymbol{h}_{-i}},h_i\right\rangle\right) V^{\pi_i}\left(h_i'\right)\,\mathrm{d}h_i' \\
&\qquad \text{(linearity)} \\
&= \int_{h_i'} p\left(h_i' \mid h_i\right) V^{\pi_i}\left(h_i'\right)\,\mathrm{d}h_i' \qquad \text{(see Lemma 3.1)} \\
&= \mathop{\mathbb{E}}_{h_i'\sim p(\cdot|h_i)} V^{\pi_i}\left(h_i'\right) \qquad \text{(definition of expectation)}
\end{aligned}$$

By using the value of A and B we get:

$$\begin{aligned}
&\mathop{\mathbb{E}}_{s,\boldsymbol{h}_{-i}\sim p(\cdot|h_i)} \left[\Delta_p(s,\langle\boldsymbol{h}_{-i},h_i\rangle)\right] \\
&= \mathop{\mathbb{E}}_{s',\boldsymbol{h}'\sim p(\cdot|h_i)} V^{\boldsymbol{\pi}}\left(s',\boldsymbol{h}'\right) - \mathop{\mathbb{E}}_{h_i'\sim p(\cdot|h_i)} V^{\pi_i}\left(h_i'\right) \\
&= \mathop{\mathbb{E}}_{h_i'\sim p(\cdot|h_i)} \mathop{\mathbb{E}}_{s',\boldsymbol{h}_{-i}'\sim p\left(\cdot|h_i,h_i'\right)} V^{\boldsymbol{\pi}}\left(s',\left\langle\boldsymbol{h}_{-i}',h_i'\right\rangle\right) - \mathop{\mathbb{E}}_{h_i'\sim p(\cdot|h_i)} V^{\pi_i}\left(h_i'\right) \qquad \text{(chain rule)} \\
&= \mathop{\mathbb{E}}_{h_i'\sim p(\cdot|h_i)} \mathop{\mathbb{E}}_{s',\boldsymbol{h}_{-i}'\sim p\left(\cdot|h_i'\right)} V^{\boldsymbol{\pi}}\left(s',\left\langle\boldsymbol{h}_{-i}',h_i'\right\rangle\right) - \mathop{\mathbb{E}}_{h_i'\sim p(\cdot|h_i)} V^{\pi_i}\left(h_i'\right) \qquad (h_i' \text{ contains } h_i) \\
&= \mathop{\mathbb{E}}_{h_i'\sim p(\cdot|h_i)} \mathop{\mathbb{E}}_{s',\boldsymbol{h}_{-i}'\sim p\left(\cdot|h_i'\right)} \left[V^{\boldsymbol{\pi}}\left(s',\left\langle\boldsymbol{h}_{-i}',h_i'\right\rangle\right) - V^{\pi_i}\left(h_i'\right)\right] \qquad \text{(linearity)}
\end{aligned}$$

Therefore we obtain:

$$\begin{aligned}
&\mathop{\mathbb{E}}_{s,\boldsymbol{h}_{-i}\sim p(\cdot|h_i)} \left[V^{\boldsymbol{\pi}}(s,\langle\boldsymbol{h}_{-i},h_i\rangle) - V^{\pi_i}(h_i)\right] \\
&\qquad = \gamma \mathop{\mathbb{E}}_{h_i'\sim p(\cdot|h_i)} \mathop{\mathbb{E}}_{s',\boldsymbol{h}_{-i}'\sim p\left(\cdot|h_i'\right)} \left[V^{\boldsymbol{\pi}}\left(s',\left\langle\boldsymbol{h}_{-i}',h_i'\right\rangle\right) - V^{\pi_i}\left(h_i'\right)\right] \qquad (3.12)
\end{aligned}$$

By applying recursively n times Equation 3.12 we obtain:

$$\mathop{\mathbb{E}}_{s,\boldsymbol{h}_{-i}\sim p(\cdot|h_i)}\left[V^{\boldsymbol{\pi}}(s,\langle \boldsymbol{h}_{-i},h_i\rangle)-V^{\pi_i}(h_i)\right]$$
$$=\gamma^n \mathop{\mathbb{E}}_{h_i^1\sim p(\cdot|h_i)}\mathop{\mathbb{E}}_{h_i^2\sim p\left(\cdot|h_i^1\right)}\dots\mathop{\mathbb{E}}_{h_i^n\sim p\left(\cdot|h_i^{n-1}\right)}\mathop{\mathbb{E}}_{s^n,\boldsymbol{h}_{-i}^n\sim p\left(\cdot|h_i^n\right)}\left[V^{\boldsymbol{\pi}}\left(s^n,\langle \boldsymbol{h}_{-i}^n,h_i^n\rangle\right)-V^{\pi_i}\left(h_i^n\right)\right] \quad (3.13)$$

We then define $R_{\max}=\max_{s\in\mathcal{S}}\max_{\boldsymbol{a}\in\boldsymbol{\mathcal{A}}}|R(s,\boldsymbol{a})|$. This allows us to bound the difference between the centralized value and the local value for all $s\in\mathcal{S},\boldsymbol{h}\in\boldsymbol{\mathcal{H}}$

$$\begin{aligned}|V^{\boldsymbol{\pi}}(s,\langle \boldsymbol{h}_{-i},h_i\rangle)-V^{\pi_i}(h_i)| &\leq |V^{\boldsymbol{\pi}}(s,\langle \boldsymbol{h}_{-i},h_i\rangle)|+|V^{\pi_i}(h_i)| && \text{(triangular inequality)}\\ &\leq \frac{R_{\max}}{1-\gamma}+\frac{R_{\max}}{1-\gamma} && \text{(upper-bound on the value)}\\ &\leq \frac{2R_{\max}}{1-\gamma}\end{aligned}$$

This allows us to bound the LHS of Equation 3.13:

$$\begin{aligned}&\left|\mathop{\mathbb{E}}_{s,\boldsymbol{h}_{-i}\sim p(\cdot|h_i)}\left[V^{\boldsymbol{\pi}}(s,\langle \boldsymbol{h}_{-i},h_i\rangle)-V^{\pi_i}(h_i)\right]\right|\\ &=\left|\gamma^n \mathop{\mathbb{E}}_{h_i^1\sim p(\cdot|h_i)}\mathop{\mathbb{E}}_{h_i^2\sim p\left(\cdot|h_i^1\right)}\dots\mathop{\mathbb{E}}_{h_i^n\sim p\left(\cdot|h_i^{n-1}\right)}\mathop{\mathbb{E}}_{s^n,\boldsymbol{h}_{-i}^n\sim p\left(\cdot|h_i^n\right)}\left[V^{\boldsymbol{\pi}}\left(s^n,\langle \boldsymbol{h}_{-i}^n,h_i^n\rangle\right)-V^{\pi_i}\left(h_i^n\right)\right]\right|\\ &\leq \gamma^n \mathop{\mathbb{E}}_{h_i^1\sim p(\cdot|h_i)}\mathop{\mathbb{E}}_{h_i^2\sim p\left(\cdot|h_i^1\right)}\dots\mathop{\mathbb{E}}_{h_i^n\sim p\left(\cdot|h_i^{n-1}\right)}\mathop{\mathbb{E}}_{s^n,\boldsymbol{h}_{-i}^n\sim p\left(\cdot|h_i^n\right)}\left|V^{\boldsymbol{\pi}}\left(s^n,\langle \boldsymbol{h}_{-i}^n,h_i^n\rangle\right)-V^{\pi_i}\left(h_i^n\right)\right|\\ &\qquad\qquad\text{(Jensen inequality)}\\ &\leq \gamma^n \mathop{\mathbb{E}}_{h_i^1\sim p(\cdot|h_i)}\mathop{\mathbb{E}}_{h_i^2\sim p\left(\cdot|h_i^1\right)}\dots\mathop{\mathbb{E}}_{h_i^n\sim p\left(\cdot|h_i^{n-1}\right)}\mathop{\mathbb{E}}_{s^n,\boldsymbol{h}_{-i}^n\sim p\left(\cdot|h_i^n\right)}\left[\frac{2R_{\max}}{1-\gamma}\right] \quad \text{(see above)}\\ &\leq \gamma^n\frac{2R_{\max}}{1-\gamma}\end{aligned}$$

As $\gamma\in]0,1[$, when $n\to+\inf$ we obtain:

$$\left|\mathop{\mathbb{E}}_{s,\boldsymbol{h}_{-i}\sim p(\cdot|h_i)}\left[V^{\boldsymbol{\pi}}(s,\langle \boldsymbol{h}_{-i},h_i\rangle)-V^{\pi_i}(h_i)\right]\right|\leq 0$$

And finally, using Eq. 3.10:

$$\mathop{\mathbb{E}}_{s,\boldsymbol{h}_{-i}\sim p(\cdot|h_i)}\left[\tilde{Q}_i(s,\langle \boldsymbol{h}_{-i},h_i\rangle,a_i)-Q^{\pi_i}(h_i,a_i)\right]=0$$

□

This result justifies using the Q-value proxy in place of the true Q-function during learning: it shows that, under fixed teammate policies, LAN optimizes an unbiased estimate of the correct learning signal. Although this guarantee does not address the challenges introduced by non-stationarity when agents update their policies concurrently, such as the need for coordinated exploration and stable learning targets, it nevertheless provides a sound foundation for learning. The next subsection examines how the structure of the Q-value proxy further supports stability and coordination when all agents update their policies simultaneously.

3.4.5 Stability and Coordination via the Q-Value Proxy

Compared to the true local Q-value Q^{π_i}, LAN's Q-value proxy $\tilde{Q}_i^{\boldsymbol{\pi}}$ has two additional desirable properties that enhance stability and coordination during learning. While these properties arise from the joint learning dynamics and cannot be isolated or tested independently, they provide insight into LAN's effectiveness.

Property 3.2: Mitigation of the Moving Target Problem

LAN, through the parallel learning of the Q-value proxies $\tilde{Q}_i^{\boldsymbol{\pi}}$ for all agents, mitigates the moving target problem.

As all agents learn concurrently, the environment perceived by each becomes non-stationary, breaking the Markov property and often causing methods like IQL to plateau prematurely. LAN mitigates this through coordinated updates: each agent's Q-value proxy combines a centralized value with a local advantage. When a local advantage is updated, the shared centralized value—computed from the same transitions—is also affected as both components are learned jointly in the same backpropagation step. Since this centralized value contributes to the Q-value targets of all agents (see Equation (3.7)), changes in one agent's learning are propagated to the others, enabling more stable and synchronized joint learning.

Property 3.3: Mitigation of Multi-Agent Credit Assignment

The Q-value proxy $\tilde{Q}_i^{\boldsymbol{\pi}}$ alleviates the multi-agent credit assignment problem.

The centralized value approximates the expected return of the joint policy. An agent can thus assess the contribution of its action by comparing the resulting Q-value to the centralized value alone. This difference (the local advantage) is learned implicitly through the Q-value proxy by construction. When applying DQN like learning to $\tilde{Q}_i^{\boldsymbol{\pi}}$,

the resulting update for the local advantage is:

$$y_{A_a} = r + \gamma \tilde{Q}^{\pi}_{i,t}\left(s', \boldsymbol{h}', \arg\max_{a'_i} \tilde{Q}^{\pi}_{i,t}\left(s', \boldsymbol{h}', a'_i\right)\right) - V^{\pi}(s, \boldsymbol{h}) \tag{3.14}$$

This form is reminiscent of COMA's counterfactual baseline [Foerster et al., 2018], and reduces credit assignment complexity by isolating the effect of each agent's choice.

3.4.6 Additional Intuitions

Beyond the formal properties above, we provide two additional intuitions that help explain LAN's effectiveness.

Intuition 1: Breaking Partial Observability. In a POMDP, a given observation-action history may correspond to multiple latent states, requiring marginalization in value estimation. In Dec-POMDPs, this complexity is amplified, as each agent must additionally marginalize over the joint histories of the other agents, i.e., over $(s, \boldsymbol{h}_{-i})$, as captured by the induced POMDP $\mathcal{P}_i$ (Definition 3.1). Since LAN's target is conditioned on the full next state and joint history, it avoids this marginalization at training time, allowing for better-informed updates. At execution time, LAN uses the local advantage to select actions based on the agent's own history, which is sufficient for decentralized execution.

Intuition 2: Reducing Learning Complexity. Value-based methods extract policies by maximizing over Q-values. While Q-values and advantages induce the same action ordering, they differ in learning complexity. Advantage functions isolate the effect of actions, while value functions require marginalization over transitions. This distinction, exploited in Dueling DQN [Wang et al., 2016], is even more useful in multi-agent settings. LAN uses the centralized value to assist in learning local advantages, allowing agents to focus on relative action quality without estimating local values directly thereby simplifying the optimization process.

3.5 Architecture

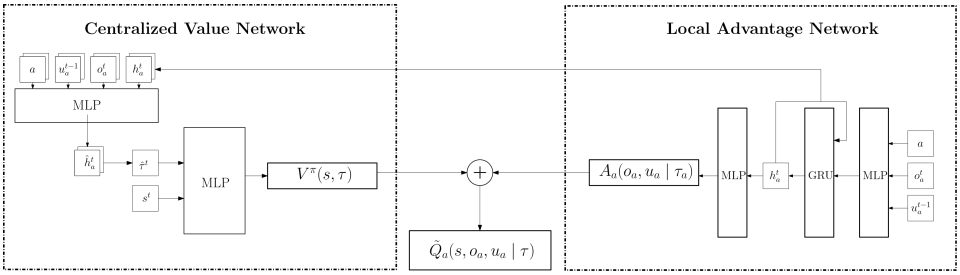


Figure 3.2: Architecture of LAN.

This section describes the architecture of LAN, which is illustrated in Figure 3.2.

To handle partial observability, LAN uses a recurrent architecture. Each agent i maintains a hidden state τ_i^t updated through a GRU that encodes its observation-action history:

$$\tau_i^t = \mathrm{GRU}_\theta(\tau_i^{t-1}, [o_i^t, a_i^{t-1}]) \tag{3.15}$$

This hidden state τ_i^t summarizes the local history h_i^t and is used to compute the local advantage function $A^{\pi_i}\left(h_i^t, a_i^t\right)$.

To construct a representation of the joint history $\boldsymbol{h}^t$ used by the centralized value function $V^{\boldsymbol{\pi}}$, LAN builds on the local features already extracted at the agent level. For each agent i, we define an enriched feature vector:

$$\tilde{\tau}_i^t = [\tau_i^t, o_i^t, a_i^{t-1}, \mathrm{id}_i] \tag{3.16}$$

where id_i is an agent identifier. This vector is then embedded using a shared feedforward network:

$$\hat{\tau}_a^t = \phi(\tilde{\tau}_a^t) \tag{3.17}$$

The joint history is then represented by summing the embedded vectors:

$$\hat{\boldsymbol{\tau}}^t = \sum_{i \in i} \hat{\tau}_i^t \tag{3.18}$$

This aggregation is scalable and ensures permutation invariance. The centralized value is then computed by combining $\hat{\boldsymbol{\tau}}^t$ with the current state s^t:

$$V^{\boldsymbol{\pi}}\left(s^t, \boldsymbol{\tau}^t\right) = f([\hat{\boldsymbol{\tau}}^t, s^t]) \tag{3.19}$$

where f is a two-layer MLP with ReLU activations.

This architecture offers two benefits:

- It allows the centralized value function to be computed from a fixed-size representation of the joint history, independent of the number of agents.
- It enables efficient training by sharing parameters across agents and without introducing an additional RNN, reducing the overall model complexity.

LAN's advantage networks share weights across agents for sample efficiency, but this is not necessary. To allow for individual policies, eventhough the advantage networks share parameters, we condition the local advantage networks on the agent's identifier id_i effectively combining efficient parameter sharing with the flexibility of individual policies. We note that the fact that agents experience different local histories already introduces a form of individualization.

As the policies in LAN are deterministic, the local advantages should, in principle, be non-positive, with the maximal advantage being zero. However, as discussed by Wang et al. [2016], explicitly enforcing this constraint, even in single-agent MDPs where the true Q-value is accessible, can negatively impact learning. Their experiments showed that applying the following normalization to the output of the advantage network leads to more stable learning:

$$A^{\pi_i}(h_i, a_i) \leftarrow A^{\pi_i}(h_i, a_i) - \frac{1}{|\mathcal{A}_i|} \sum_{a \in \mathcal{A}_i} A^{\pi_i}(h_i, a) \tag{3.20}$$

In the single-agent setting, this results in an advantage function that differs from the true one by a constant offset. In LAN, because the centralized value is shared across agents, enforcing a zero-mean constraint on the local advantages would force this offset to be shared across all agents. While this coupling might, in some cases, help promote coordinated behavior, it also introduces an additional constraint across networks that can hinder learning. Following the findings of Wang et al. [2016], we investigated both the zero-mean and non-positivity constraints in LAN and found that both significantly impaired learning.

As a result, LAN does not apply any constraint to the outputs of the local advantage network. The experimental section contains an ablation study that compares LAN with and without the mean constraint, showing that the latter performs better in practice.

3.6 Experiments

We evaluate LAN across a set of cooperative multi-agent environments designed to test learning stability, scalability, coordination, and credit assignment. Our primary benchmark is the StarCraft Multi-Agent Challenge (SMAC) [Samvelyan et al., 2019], which was the reference testbed for reinforcement learning in Dec-POMDPs at the time of this work. SMAC provides a diverse set of high-dimensional, partially observable tasks that require decentralized execution and varying degrees of coordination.

To assess the generality of our approach, we further evaluate LAN on a variant of the simple spread task from the Multi-Agent Particle Environment (MPE) [Mordatch and Abbeel, 2017], where we vary the number of agents. This allows us to test LAN's ability to transfer to environments with different dynamics and coordination structures.

We include experiments on the Checkers environment, originally introduced by VDN [Sunehag et al., 2018], which is designed to isolate the multi-agent credit assignment problem. This task provides an additional lens on LAN's effectiveness in attributing individual contributions in cooperative settings.

3.6.1 StarCraft Multi-Agent Challenge

The StarCraft Multi-Agent Challenge[1] (SMAC) [Samvelyan et al., 2019] is a set of environments that runs in the popular video game StarCraft II. SMAC does not focus on the full game but rather on micromanagement tasks where two teams of agents, possibly heterogeneous and imbalanced, fight. A match is considered won if the other team is eliminated within the task dependent time limit. Each agent only observes its surroundings and receives a team reward proportional to the damage done to the other team plus bonuses for killing an enemy and winning. The action space of each agent consists of a move action to each cardinal direction, a no-op action, and an attack action for each enemy which is replaced by a heal action for each team member for the Medivacs units. The attack/heal action only affects units within range. As the agent's observation and action space are linearly dependent on the number of agents to perform well scalability is a key issue. SMAC also provides the real state of the environment, which we use as input for the centralized value. The benchmark is composed of 14 different maps (Table 3.1) that are designed to assess different aspects of cooperation. They are ranked into 3 categories: easy, hard, and super hard maps.

Table 3.1: The different maps of SMAC.

Map Name	Ally Units	Enemy Units
2s3z	2 Stalkers & 3 Zealots	2 Stalkers & 3 Zealots
3s5z	3 Stalkers & 5 Zealots	3 Stalkers & 5 Zealots
1c3s5z	1 Colossus, 3 Stalkers & 5 Zealots	1 Colossus, 3 Stalkers & 5 Zealots
5m_vs_6m	5 Marines	6 Marines
10m_vs_11m	10 Marines	11 Marines
27m_vs_30m	27 Marines	30 Marines
3s5z_vs_3s6z	3 Stalkers & 5 Zealots	3 Stalkers & 6 Zealots
MMM2	1 Medivac, 2 Marauders & 7 Marines	1 Medivac, 3 Marauders & 8 Marines
2s_vs_1sc	2 Stalkers	1 Spine Crawler
3s_vs_5z	3 Stalkers	5 Zealots
6h_vs_8z	6 Hydralisks	8 Zealots
bane_vs_bane	20 Zerglings & 4 Banelings	20 Zerglings & 4 Banelings
2c_vs_64zg	2 Colossi	64 Zerglings
corridor	6 Zealots	24 Zerglings

3.6.2 Configuration

To ensure a fair comparison, we follow the standard experimental protocol introduced by Samvelyan et al. [2019] and adopted by QMIX and QPLEX. In particular, we:

- Use the same architecture for the local advantage network as for the decentralized utility networks in QMIX and QPLEX, consisting of a feed-forward layer with 64 units followed by a GRU with 64 units.
- Train each algorithm on a single environment for 2 million timesteps.
- Evaluate greedy policies every $10k$ timesteps over 32 episodes.
- Use the same hyperparameters across all maps.
- Disable TD(λ).

The centralized value network is composed of feed-forward layers with 128 units: one to embed each agent's input vector $\tilde{\tau}_i$, and two additional layers to process the aggregated joint history representation $\tilde{\boldsymbol{\tau}}$. Table 3.2 summarizes the hyperparameters used for training LAN.

We train LAN with at least 10 different random seeds and report the median win rate across time, along with the first and third quantiles. While in RL we typically report the average return, the evaluation method of Samvelyan et al. [2019] uses the win rate as

[1]We use version SC2.4.6.2.69232 and not SC2.4.10. Performances are not comparable between versions.

Table 3.2: LAN Training Hyperparameters

Parameter	Value
Replay buffer size	5,000 episodes
Training timesteps	2 million
Episodes per batch	32
Target update frequency	Every 200 gradient updates
Exploration schedule	ε-greedy, from 1 to 0.05 over 50k steps
Evaluation episodes	32 every 10k steps
Learning rate	$5 \cdot 10^{-4}$
Optimizer	Adam
Gradient clipping	Max norm 10
Activation function	ReLU
Number of environments	1
Number of updates per episode	2

the main metric. This means that while the win rate might stagnate, the average return which is what the agents are optimizing, can still increase.

We note that LAN does not require parameter sharing, and that each type of agent could have its own model. In that case, every agent type also needs its own embedding network to compute $\tilde{\tau}_i$.

3.6.3 Performance Analysis on SMAC

We evaluate LAN on SMAC, comparing it to IQL, VDN, QMIX, and QPLEX. These baselines represent established value-based methods under the CTDE paradigm, with increasing representational capacity and coordination ability. IQL serves as a non-centralized baseline, VDN and QMIX introduce progressively more expressive value decompositions, and QPLEX is one of the most performant factorization-based methods. For IQL, VDN, and QMIX, we used the QMIX implementation, and for QPLEX we used the official codebase.

We also introduce a variant of IQL that uses the full state as input to the Q-value function, which we refer to as *IQL (no fog)*. This variant is introduced in 3.6.5 and serves as an ablation to assess how LAN mitigates the moving target problem.

Results on representative maps.

Figure 3.3 presents the median win rates on 9 representative SMAC maps, selected to highlight the diversity and difficulty of coordination challenges. The first row features

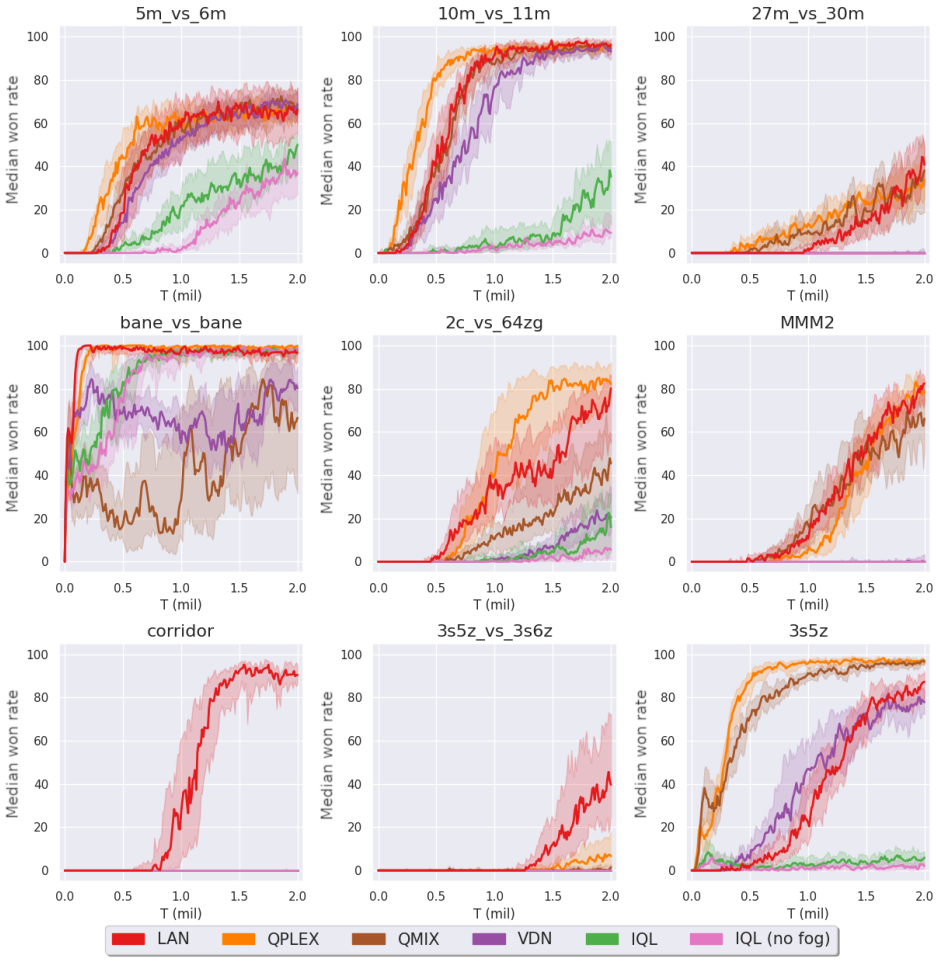


Figure 3.3: Median battle won rate during learning on 9 maps of SMAC. Each algorithm is run on at least 10 different seeds per map. Following the evaluation method of Samvelyan et al. [2019], we train the agents on 2 million steps and plot the median, with the 1st and 3rd quantiles. IQL (no fog) is introduced in Section 3.6.5.

fights between marines with an increasing number of agents and the enemy controlling more units. The second row is composed from left to right of a balanced map with 24 heterogeneous units per team, a map where 2 power-full units fight a swarm of 64 smaller enemies, and an unbalanced heterogeneous map with a medic units that as a side effect increases the action space. The last row shows the result on two super-hard maps (corridor and 3s5z_vs_3s6z) where the baselines do not reach any wins, and a map (3s5z) where LAN seems to under-perform.

Homogeneous unbalanced maps

In the maps of the first row of Figure 3.3, two unbalanced teams with homogeneous units fight against each other, with our team composed of fewer units than the enemy: in 5m_vs_6m 5 agents fight 6 enemies, in 10m_vs_11m 10 agents fight 11 enemies, and in 27m_vs_30m 27 agents fight 30 enemies. The ratio between the number of agents and the number of enemies makes the map 10m_vs_11m easier compared to the other two. In the map 27m_vs_30m, both the number of agents and the dimension of the observation and action space constitute a real challenge for MARL. In those three maps, LAN dominates IQL and performs on par with SOTA at the end of training.

First, as IQL is a natural ablation of LAN, we deduce from this experiment that the centralized value introduced by LAN does indeed help to coordinate the learning of the agents and that LAN can address the shortcomings of IQL.

Second, LAN does not only performs on par with the SOTA, and slighty outperforms the other algorithms in the more difficult map, it is also more scalable than QMIX and QPLEX in terms of parameters of its centralized component with respect to the number of agents (Table 3.3). Indeed, between 5m_vs_6m and 27m_vs_30m the number of agents is multiplied by 5.4 and the number of parameters of LAN's centralized value is only multiplied by a factor of 2, while for the centralized component of QMIX and QPLEX this factor is respectively 8.8 and 16.5.

Heterogeneous or large-action maps

The second row of Figure 3.3, is composed of two hard and one super-hard maps.

The first one, bane_vs_bane, opposes two large and balanced teams of 24 heterogeneous units. We observe that while IQL easily reaches 100% of winning rate, VDN struggles to learn and QMIX fails to learn. This behavior suggests that the representational and optimization constraints imposed by monotonic value factorization become increasingly restrictive as the number of agents grows, limiting their ability to capture the complex coordination patterns required in this setting. In contrast, IQL avoids these constraints by learning fully decentralized value functions, which appear sufficient for this task. This observation supports our claim that alternative research directions to value factorization are needed, in particular approaches that remain closer

to independent learning while explicitly addressing its known instability issues in more challenging environments. QPLEX is able to learn the perfect strategy at the cost of doubling the number of parameters compared to QMIX. LAN also learns to consequently eliminate the opposing team and reaches a perfect score with 5 times fewer parameters than QPLEX.

The second map, 2c_vs_64zg, matches two powerful agents against 64 weaker agents. The numerous enemies make the action space very large, with 70 actions, which is a known challenge in RL [Zahavy et al., 2018]. In this map, QPLEX reaches a final performance of 83% win rate followed closely by LAN with 80%, while QMIX, VDN and IQL score respectively around 50%, 20% and 15% win rate.

The third map, MMM2, features two unbalanced heterogeneous teams, with the enemy team having 2 additional units, and is the only map including medical units. While IQL and VDN do not obtain any wins, QMIX and QPLEX score 60% and 80% respectively. LAN obtains the same final performance as QPLEX.

Super-hard maps

The last row of Figure 3.3 presents LAN's performance on 2 super-hard maps alongside the easier version of one of those maps.

In the super hard map corridor, 6 agents of type 'zealot' fight a team 24 enemies of type 'zerlings'. While the SMAC paper claimed that the only solution for this map was to take advantage of the terrain (a spawning zone connected to a second zone by a corridor) to limit the number of enemies that can attack our agents, LAN discovered another solution. One agent lures part of the enemies to a remote location while the rest fights the remaining enemies. After killing the bait a fraction of the enemies attack our agents while the majority go through the corridor to reach the second zone. Our agents defeat their attackers, and after regenerating part of their shields move to the second zone to finish off the enemies. While the current SOTA flattens to zero, LAN obtains an almost perfect score with around 90% success rate.

On the next super hard map, 3s5z_vs_3s6z, LAN learns good decentralized policies with a performance at around 40%. The only other algorithm that was able to achieve any wins is QPLEX with less than 10%. The strategy is similar as the one learned in corridor, a stalker (long-range unit) baits most of the enemy's zealots (close combat units) into targeting him. It then flees far away from his teammates and sacrifices himself so that the other agents can kill the stalkers and remaining zealots. The agents can then easily kill the remaining enemies as they are no longer protected by any long-range support.

The last map of Figure 3.3, 3s5z, is the balanced version of the previous map and therefore easier. In this map, LAN reaches 87% median battle win rate, whereas VDN only scores 80%, and QMIX and QPLEX obtain 97%. This underperformance is intriguing as LAN performs better than the other algorithms in the harder version of this map. By

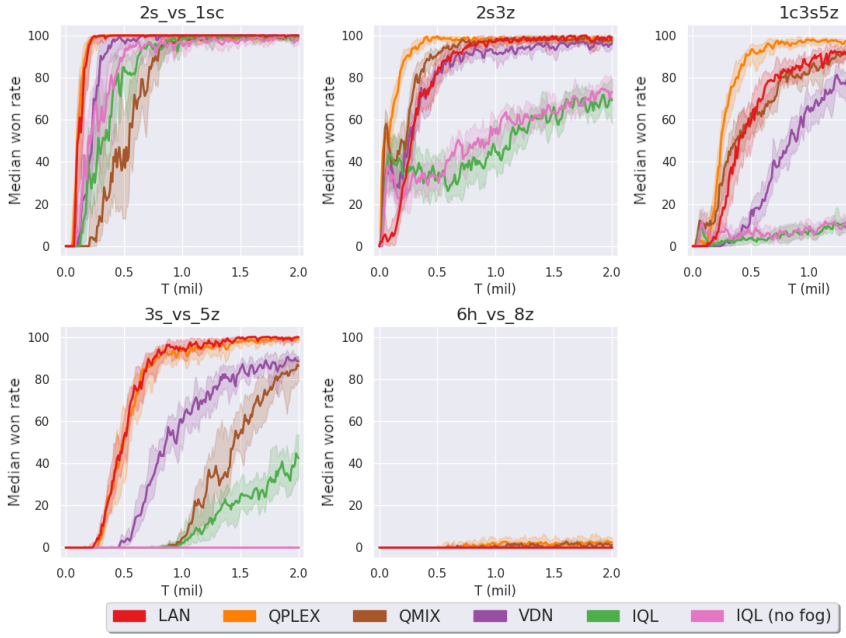


Figure 3.4: Median battle won rate during learning on the last 5 SMAC maps.

visualizing the learned policies in 3s5z we discovered that LAN converges to two different policies: a) a basic confrontation policy which is the policy learned by QMIX and QPLEX; b) a baiting strategy identical to the one learned in 3s5z_vs_3s6z. We also remark that LAN appears to still be learning and might converge to the same performance as the other QPLEX if given more time. We further analyze the implications of this multimodal behavior in the credit assignment discussion in Section 3.6.4.

LAN's strong performance on the last two super-hard maps appears linked to the emergence of coordinated strategies that involve role specialization and sacrifice, such as baiting enemies away from the team. In particular, the ability to maintain high win rates despite early agent deaths suggests that LAN effectively reinforces temporally extended behaviors, where actions that are locally disadvantageous can be credited for their delayed, team-level benefits. For instance, sacrificial behaviors occurring early in an episode can be reinforced when they consistently lead to improved outcomes for the remaining agents later on. This is in contrast with the mixing-based methods, which often fail to learn such strategies. A more detailed analysis of these behaviors, and how LAN addresses the underlying credit assignment challenges, is provided in Section 3.6.4.

Results on remaining maps.

Figure 3.4 presents the results on the remaining five SMAC maps not included in the previous subsection. On the easy map 2s_vs_1sc, all algorithms, including LAN, reach a perfect win rate. In 2s3z and 1c3s5z, LAN performs on par with QPLEX and QMIX, while IQL fails to learn an effective policy, highlighting once again the limitations of fully decentralized training. The map 3s_vs_5z shows similar trends, with LAN and QPLEX achieving perfect scores and QMIX and VDN plateauing around 85%. On the final map, 6h_vs_8z, no algorithm, including LAN, manages to win, underscoring the difficulty of this particular scenario.

Parameter efficiency.

Table 3.3: Number of parameters (x1000) of the value function in LAN vs. the mixing network in QPLEX/QMIX. The dependency of the dimension of the observation and action space in the number of agents is the only cause of the difference in the number of parameters of LAN's centralized value network in the different maps

	LAN	**QPLEX**	**QMIX**
2s3z	62	50	36
3s5z	74	90	60
1c3s5z	83	113	73
5m_vs_6m	56	43	32
10m_vs_11m	68	106	70
27m_vs_30m	111	709	283
3s5z_vs_3s6z	76	95	63
MMM2	86	136	85
2s_vs_1sc	46	18	12
3s_vs_5z	54	31	22
6h_vs_8z	61	59	42
bane_vs_bane	125	555	241
2c_vs_64zg	119	116	72
corridor	79	109	69

Table 3.3 reports the number of parameters in the centralized component of LAN compared to the mixing networks used in QPLEX and QMIX. LAN consistently maintains a smaller or comparable parameter count, with significantly better scalability in scenarios with many agents. For example, from 5m_vs_6m to 27m_vs_30m, the number of agents increases by a factor of 5.4, yet the centralized value network in LAN grows by only a

factor of 2. In contrast, QPLEX and QMIX grow by factors of 16.5 and 8.8 respectively. This demonstrates the efficiency of LAN's architecture in large-scale settings.

SMAC summary performance.

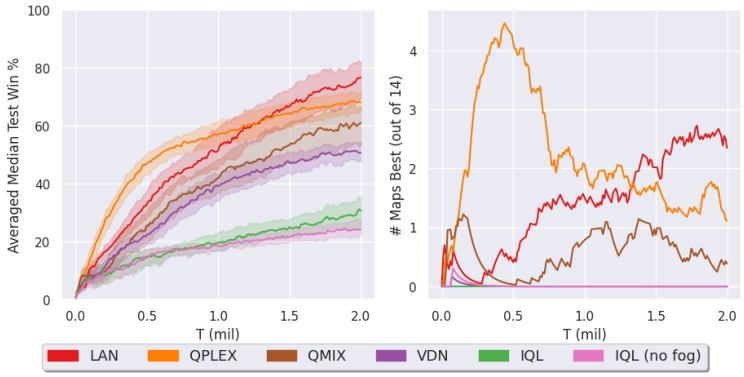


Figure 3.5: (Left) Average median test win rate across the 14 SMAC maps during training. Shaded regions denote the interquartile range. (Right) Number of maps where each algorithm achieves the highest median win rate by a margin of at least 1/32.

Following the evaluation method of Samvelyan et al. [2019], Figure 3.5 summarizes LAN's overall performance across the entire SMAC benchmark. The left panel shows the average median win rate over time, while the right panel counts the number of maps where each method performs best by at least 1/32.

IQL achieves only 30% average win rate and fails to dominate on any map, which is expected given its fully decentralized learning and susceptibility to the moving target problem. VDN and QMIX show similar learning dynamics early on, but QMIX ultimately outperforms VDN, reaching a final win rate of 60%. QPLEX learns faster than both and reaches 67% final performance in roughly half the training steps. LAN starts slower than QPLEX but surpasses it at around 1.25×10^6 steps, ultimately achieving the best overall performance with 77% win rate.

The right panel of Figure 3.5 shows that LAN outperforms all other methods on three challenging maps: corridor, 3s5z_vs_3s6z, and 5m_vs_6m. These results highlight LAN's robust average-case performance, strong generalization, and particular advantage in coordination-intensive environments.

3.6.4 Credit Assignment Analysis

Several of LAN's strongest results, particularly on super-hard SMAC maps like corridor and 3s5z_vs_3s6z, are driven by its ability to assign credit to agents whose actions only indirectly or delayedly contribute to team success. In these scenarios, LAN consistently discovers emergent strategies that involve agent specialization and sacrifice.

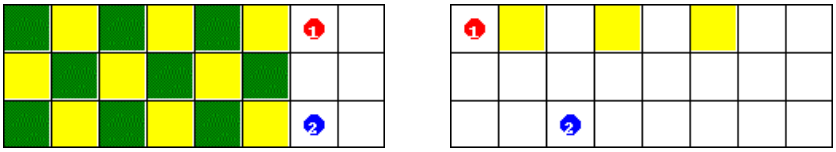
In corridor, LAN learns to sacrifice one agent early in the episode to lure a subset of enemies away from the main team. This baiting strategy enables the rest of the agents to engage fewer enemies at once and win the fight with high probability. A similar pattern emerges in 3s5z_vs_3s6z, where one stalker draws the enemy zealots far from the rest of the team before being eliminated. This creates a favorable team-fight scenario in which the remaining agents defeat the unprotected enemy units. Importantly, these strategies rely on agents performing actions that yield negative local outcomes (i.e., death) but benefit the team as a whole. The fact that such behavior is reinforced demonstrates LAN's capacity to solve the multi-agent temporal credit assignment problem.

We believe this type of coordinated behavior is easier to discover with LAN than with mixing-based algorithms due to the architecture of the Q-value proxy. In this respect, LAN is similar to multi-agent actor–critic methods that employ a shared centralized critic across agents, as its centralized value function is shared and trained jointly with local advantage functions, allowing credit to be assigned even to agents that are no longer active. In particular, dead agents can still benefit from the team reward as captured by the value network, enabling the reinforcement of self-sacrificial behavior that contributes to future team success. In contrast, factorization-based methods like QMIX and QPLEX introduce individual utility functions that agents are trained to maximize. When these individual utilities misalign with the team objective—as in the case of baiting strategies—such methods seem to struggle to learn coordinated behavior effectively.

By focusing on learning best-response policies for each agent rather than decomposing a joint Q-function into individual utilities, LAN directly optimizes for the global return. The complex strategies observed in corridor and 3s5z_vs_3s6z demonstrate that LAN can discover and reinforce temporally extended multi-agent behaviors that factorization-based methods fail to capture.

This capacity is further confirmed by an experiment on the Checkers environment [Sunehag et al., 2018], designed explicitly to probe credit assignment. The environment features two agents navigating a grid containing apples (positive reward) and lemons (negative reward), with asymmetric reward magnitudes. Apples yield $+10$ when eaten by agent 1 and $+1$ for agent 2; lemons yield -10 and -1 respectively. Crucially, both agents receive the same total team reward, which is the sum of their individual outcomes.

As shown in Figure 3.6, LAN converges to an optimal policy where agent 2 deliberately consumes only the lemons blocking the path, enabling agent 1 to collect all the apples.



(a) Initial state of the environment.

(b) Final state of the environment reached by LAN's policy.

Figure 3.6: The Checkers environment. Green boxes are apples, yielding +10 (agent 1) and +1 (agent 2); yellow boxes are lemons, yielding −10 and −1 respectively.

This outcome demonstrates that LAN assigns appropriate credit to agent 2's negative-reward actions, reinforcing them because they enable future high-reward actions by agent 1. Despite the shared team reward and local observations, LAN's Q-value proxy allows it to propagate return-based feedback effectively and learn role-asymmetric cooperation strategies.

Taken together, these findings illustrate LAN's capabilities to mitigate the multi-agent credit assignment problem.

3.6.5 Moving target problem analysis

IQL serves as a natural ablation of LAN, where the centralized value function is removed and agents are trained purely from their individual trajectories and rewards. While this simplicity makes IQL scalable and fully decentralized, it introduces a major drawback: the moving target problem. Because each agent updates its policy independently, the non-stationarity induced by simultaneously learning teammates causes instability in the learning targets. As a result, agents often fail to converge to coordinated behaviors.

In scenarios such as bane_vs_bane, where coordination is unnecessary or when agents have no mutual influence, IQL can exhibit satisfactory performance. However, the notable superiority of LAN over IQL across all maps demonstrates that LAN effectively addresses the limitations of IQL, including the challenge posed by the moving target problem.

As shown in Figure 3.5, IQL achieves only 30% average median win rate and fails to outperform any baseline on any map. It performs particularly poorly on tasks that require tight cooperation or asymmetric strategies, such as MMM2, 10m_vs_30m, and 3s5z. These results further highlight IQL's difficulty in learning under joint policy non-stationarity.

In contrast, LAN significantly alleviates this issue through its Q-value proxy, which incorporates a shared centralized value function $V^{\pi}(s, \boldsymbol{h})$ that stabilizes learning targets

during training. By conditioning the centralized component on the full state and joint history, the centralized value acts as a consistent signal across agents, reducing variance and promoting coordinated policy updates. This structure mitigates the moving target problem while still enabling decentralized execution through local advantage functions.

To confirm that LAN's improvement is not merely due to increased observability from access to the state, we conduct an additional ablation experiment where IQL is trained without the fog of war. This variant, denoted "IQL (no fog)" in Figures 3.3 and 3.5, removes partial observability by allowing all agents to see the full state at each timestep. The architecture and training setup, including the joint buffer, are otherwise kept identical.

Despite this enhanced observability, IQL (no fog) still performs worse than LAN on nearly all maps. This confirms that the benefits of LAN arise not simply from access to more information, but from the coordinated learning enabled by the Q-value proxy and centralized value function.

Interestingly, IQL with full observability (IQL no fog) performs worse than standard IQL across all maps. While increased observability might be expected to help, this result suggests that simply providing more information is not sufficient to stabilize learning in multi-agent settings. In the absence of an explicit coordination or credit assignment mechanism, full observability can in fact introduce a large amount of irrelevant or weakly informative input, increasing the difficulty of representation learning and exacerbating non-stationarity. In contrast, LAN consistently outperforms both variants. This supports the claim that LAN's performance gains stem not merely from access to privileged information, but from the coordinated learning enabled by the Q-value proxy, which mitigates instability and improves credit assignment.

In summary, LAN effectively mitigates the moving target problem through centralized training and shared value estimation. This enables it to stabilize policy learning in multi-agent settings, resulting in significantly improved performance over fully decentralized methods.

3.6.6 Ablation: Mean Advantage Proxy

As mentioned in the Architecture section, in single-agent reinforcement learning, the advantage function is often normalized by subtracting the mean across actions (Equation 3.20). In this subsection, we investigate the impact of applying a similar mean-subtraction to the advantage function in LAN, and we denote this variant as **LAN mean**. Figure 3.7 compares the performance of LAN with and without mean-subtracted advantages across all SMAC maps.

While this modification improves performance in a few settings—such as 3s5z and 27m_vs_30m—it significantly degrades learning in others. In particular, it prevents any learning from occurring on the super-hard maps corridor and 3s5z_vs_3s6z, where the

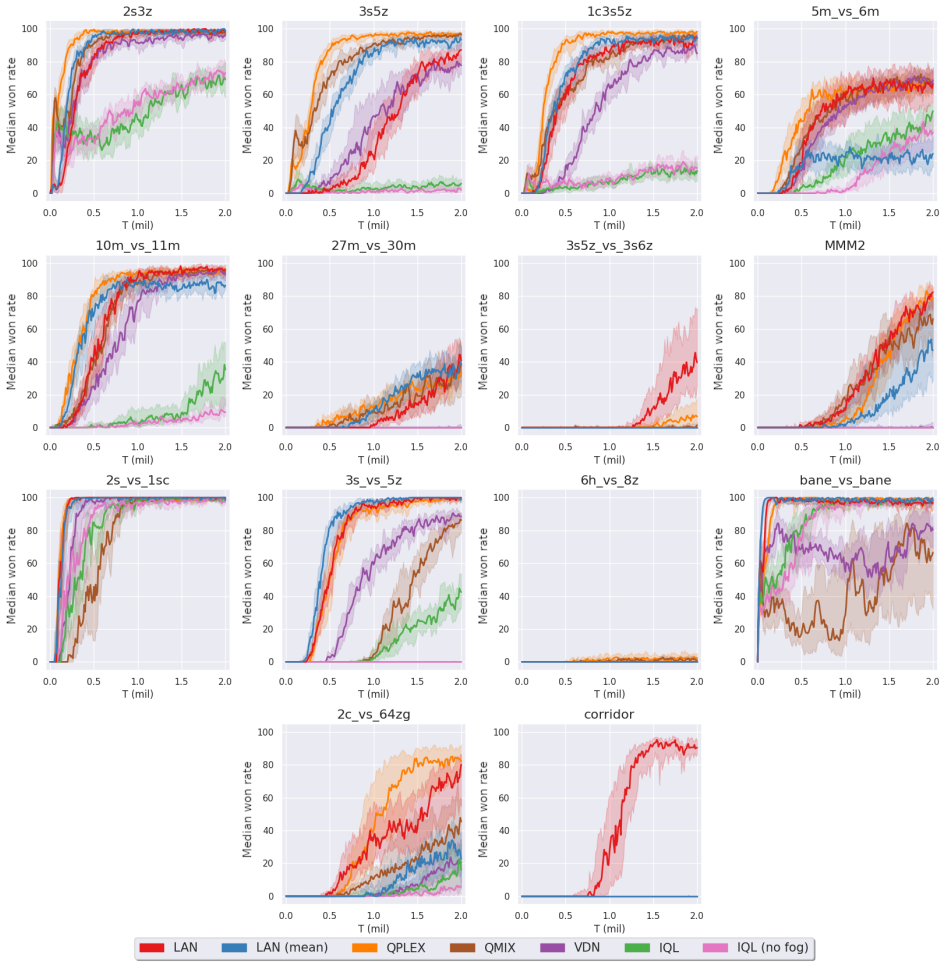


Figure 3.7: Median battle win rate during training on all SMAC maps, comparing LAN with its mean-subtracted variant (LAN mean).

original LAN achieves strong performance through emergent asymmetric strategies. It also reduces final performance on several maps including 5m_vs_6m, 2c_vs_64zg, and MMM2.

These results suggest that while mean-subtracted advantages may be beneficial in single-agent settings, their impact in multi-agent scenarios is less straightforward. In LAN, the advantage function is learned jointly with the centralized value function to support decentralized execution. Subtracting the mean may distort the alignment between the centralized value and the local advantage, especially in environments where credit is unevenly distributed or sparse. As a result, this variant fails to reinforce crucial behaviors such as baiting or role specialization, which are central to LAN's success on the most challenging tasks.

We conclude that the use of raw advantage values, as in the original LAN formulation, is better suited for cooperative multi-agent reinforcement learning, where subtle inter-agent dependencies and asymmetric contributions must be preserved during learning.

3.6.7 Generalization to MPE

To assess whether the benefits of LAN generalize beyond SMAC, we evaluate it on a modified version of the *simple spread* task from the Multi-Agent Particle Environment (MPE) Mordatch and Abbeel [2017]. This environment consists of multiple agents that must spread out and cover a set of landmarks while avoiding collisions. The original task provides full observability and has a fixed number of three agents and landmarks.

We introduce two key modifications: (i) **partial observability**, where each agent only observes other entities within a fixed radius, and (ii) **variable agent count**, evaluating performance with 3, 6, and 9 agents. The environment size is scaled proportionally with the number of agents to preserve spatial density. The reward combines a coverage component (negative distance to nearest landmark) and a collision penalty. As in SMAC, all algorithms are trained for two million steps and evaluated on 10 random seeds.

Figure 3.8 shows the learning curves for all methods across the different agent configurations. With 3 agents, LAN quickly learns the task and outperforms QPLEX and QMIX, while VDN performs poorly and IQL fails to learn altogether. As the number of agents increases to 6 and 9, LAN continues to converge faster than the other methods and reaches comparable or better final performance. This trend highlights LAN's robustness to scale and its ability to learn efficiently in partially observable environments with decentralized execution.

Overall, these results confirm that the architectural advantages of LAN—such as stable training targets and decomposability into local policies—are not specific to SMAC, but transfer effectively to other cooperative multi-agent settings with different dynamics, observation structures, and coordination challenges.

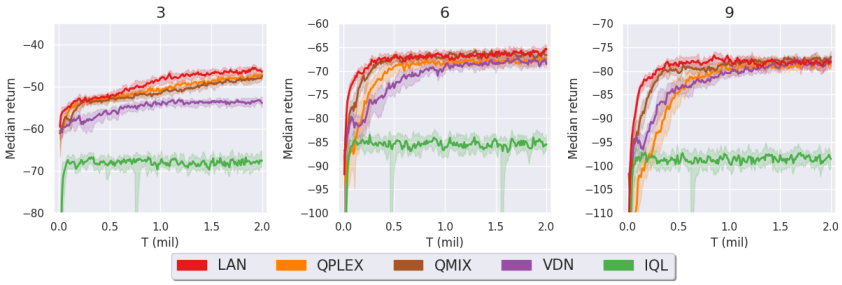


Figure 3.8: Median return during training on the modified *simple spread* environment from MPE with 3, 6, and 9 agents. Shaded regions represent the first and third quantiles.

Summary

In this section, we have demonstrated the effectiveness of LAN across a wide range of cooperative multi-agent benchmarks. On SMAC, LAN matches or outperforms state-of-the-art methods such as QPLEX on most maps, and achieves dominant performance on the most challenging tasks involving delayed credit and asymmetric cooperation. Through detailed analyses, we showed that LAN successfully mitigates the moving target problem and addresses the multi-agent credit assignment challenge via its shared centralized value function and local advantage decomposition. Additional ablations confirmed the importance of LAN's original design, and experiments on the MPE benchmark showed that these benefits generalize beyond the StarCraft domain. Together, these results establish LAN as a scalable, robust, and principled approach to learning in partially observable multi-agent environments.

3.7 Conclusion

In this chapter, we introduced *Local Advantage Networks* (LAN), a novel value-based algorithm for cooperative multi-agent reinforcement learning in Dec-POMDPs. LAN embodies the central idea of this thesis: leveraging access to state information during training to improve learning and coordination in settings characterized by partial observability and decentralized execution.

LAN follows the centralized training with decentralized execution (CTDE) paradigm by constructing a Q-value proxy for each agent. This proxy decomposes into a centralized value function—conditioned on the full state and joint history—and a local advantage function that depends only on an agent's observation-action history. The centralized component provides a shared training target across agents, addressing non-stationarity

and stabilizing the learning process. Crucially, the proxy formulation ensures that decentralized execution is preserved, as each agent's policy depends solely on its local history. The resulting architecture scales gracefully with the number of agents, since the joint value is computed over compressed hidden states rather than explicit joint inputs.

We evaluated LAN on the StarCraft Multi-Agent Challenge (SMAC) and showed that it performs competitively with or surpasses state-of-the-art value factorization methods such as QMIX and QPLEX. LAN achieved significant improvements on the most challenging scenarios, including tasks requiring temporally extended, asymmetric cooperation. These empirical results validate LAN's ability to mitigate the moving target problem and to solve the multi-agent credit assignment challenge—both of which are critical obstacles in decentralized partially observable environments. Furthermore, a generalization study on a modified version of MPE confirmed the robustness and transferability of LAN's architecture beyond SMAC.

LAN contributes a distinct perspective to the literature on multi-agent value-based methods. While much of the recent work has focused on improving the expressiveness of value factorization techniques, LAN departs from this line of research by instead leveraging state access to construct shared training signals that promote coordinated behavior. As such, LAN provides a scalable and principled alternative for learning decentralized policies in partially observable settings, and serves as a concrete instantiation of the broader thesis goal: designing algorithms that effectively leverage access to the environment's state during training to improve performance and coordination under partial observability.

3.7.1 Limitations and scope

While LAN demonstrates strong empirical performance on challenging cooperative benchmarks, a more systematic analysis is required to fully characterize the classes of environments in which it provides the largest benefits over value factorization approaches, as well as to better understand its potential limitations.

In particular, it would be valuable to assess whether value factorization methods can recover coordination strategies similar to those learned by LAN, such as temporally extended baiting or role asymmetric behaviors, and to identify the conditions under which such strategies emerge. Such analyses would contribute to a deeper understanding of the respective inductive biases of advantage based and factorization based approaches.

A further limitation of the empirical analysis arises from the evaluation protocol adopted in SMAC, which required a single set of hyperparameters to be used across all scenarios. This restricts the ability to analyze how sensitive LAN and competing baselines are to environment specific factors and may obscure performance trade offs across different coordination regimes. As a result, the reported experiments provide limited insight into how hyperparameter choices interact with environment structure,

and more targeted evaluations would be required to fully understand LAN's behavior across diverse settings.

At the time of this work, SMAC constituted the primary benchmark for Dec-POMDPs. Although SMAC offers a diverse collection of coordination tasks, it captures only a subset of possible multi agent interaction patterns. Since then, new benchmarks with different observability assumptions, coordination requirements, and dynamics have emerged, providing opportunities for a more comprehensive evaluation of LAN across a wider range of settings.

4 Online Planning in POMDPs with State-Requests

This chapter is based on the paper *Online Planning in POMDPs with State Requests* [Avalos et al., 2025], co-authored with Raphaël Avalos, Ann Nowé, and Diederik M. Roijers, published in the Reinforcement Learning Journal (RLJ) in 2024. The paper was presented at the Reinforcement Learning Conference (RLC) 2024.

Core Intuition

In partially observable decision making, agents must reason about uncertainty over the underlying state while planning into the future. When explicit state access is available at a known cost, many distinct histories may assign positive probability to the same underlying state, and resolving uncertainty can reveal that state regardless of the specific history.

This ability introduces a substantial structural redundancy in the planning problem. Once the state is revealed, future decision making depends only on the realized state and not on how it was reached. Classic tree-based online planners are not equipped to exploit this structure. By conditioning on full histories and expanding the search space as a tree, they repeatedly solve identical downstream planning problems whenever the same state can be reached through different trajectories.

The core idea behind AEMS-SR is to explicitly leverage this redundancy. By moving from a tree-structured search to a graph-based representation, belief nodes that correspond to the same fully-revealed underlying state can be merged, allowing planning computations to be reused across histories. This graph formulation introduces additional algorithmic complexity compared to tree search, but the resulting gains in efficiency and scalability make this tradeoff favorable in settings with explicit state access.

4.1 Introduction

In many real-world scenarios, full state information is occasionally accessible, but only at a significant cost. For example, an agent might rely on low-power sensors for normal operation but has the option to activate a high-accuracy sensor that consumes substantial resources, or consult a human expert, which is time-consuming or expensive. This setting challenges the agent to plan effectively under partial observability while judiciously requesting state information when the benefits outweigh the costs.

We formalize this problem setting as *Partially Observable Markov Decision Processes with State Requests (POMDP-SR)*, an extension of the classical POMDP in which the agent has access to a one-step oracle that reveals the true underlying state at a known cost. This formulation captures a broad range of scenarios, from robotic applications involving costly sensors to privacy-sensitive decision making.

While it is theoretically possible to reduce POMDP-SRs to standard POMDPs (see Section 4.4), doing so leads to a significant increase in both the time horizon and observation space, which negatively impacts the performance of conventional online planning methods such as POMCP [Silver and Veness, 2010] and AEMS [Ross et al., 2007]. These methods, based on tree-based search, are particularly sensitive to such growth, resulting in a prohibitively large branching factor.

To address this, we introduce *AEMS-SR* (**A**nytime **E**rror **M**inimization **S**earch with **S**tate **R**equests), a novel online planning algorithm tailored for POMDPs with state access. AEMS-SR departs from traditional tree search by constructing a graph-based search space, enabling it to reuse belief nodes across different trajectories and thereby avoid redundant expansions. This graph structure significantly mitigates the combinatorial explosion induced by state requests.

While we instantiate our approach using AEMS [Ross et al., 2007] due to its strong empirical performance [Ross et al., 2008], the proposed graph-based approach of state requests is not specific to AEMS and can be applied to other heuristic-based online belief space planners.

We make the following contributions:

- We formalize the POMDP-SR setting, highlighting its relevance to practical decision-making tasks with costly state access.
- We propose AEMS-SR, a graph-based online planning algorithm that maintains theoretical ε-optimality guarantees.
- We evaluate AEMS-SR on the standard Tag domain and on RobotDelivery, a novel benchmark designed for this setting, and show that it consistently outperforms existing online planning baselines.

This chapter is organized as follows. We first introduce in the Background (Section 4.2) the notations and AEMS the algorithm from which we build our approach. We then contextualize our work within the existing literature in Section 4.3. In Section 4.4, we formally define the POMDP-SR framework and discuss its relation to standard POMDPs. Section 4.5 introduces the AEMS-SR algorithm and analyzes its theoretical properties. We present empirical results in Section 4.7, demonstrating the effectiveness of AEMS-SR compared to existing online planning methods. Finally, we conclude in Section 4.8 with a discussion of limitations and future directions.

4.2 Background

4.2.1 Notation

We refer to beliefs as *corner beliefs* when the probability mass is entirely concentrated on a single state s, i.e., $b(s) = 1$ and $b(s') = 0$ for all $s' \neq s$. In these cases, we use the state s directly to refer to the belief. The *support* (Section 2.1.1) of a belief, denoted $\text{supp}(b)$, is the set of states with non-zero probability under b.

We denote the set of fringe nodes (leaf nodes without children) in a tree $\mathcal{T}$ as $\mathcal{F}(\mathcal{T})$. The depth of a node b in the tree $\mathcal{T}$ is denoted as $d_{\mathcal{T}}(b, b_0)$, where b_0 is the root node of the tree.

4.2.2 AEMS

Anytime Error Minimization Search (AEMS) [Ross et al., 2007] is an online planning algorithm that incrementally builds a search tree $\mathcal{T}$ rooted at the current belief b_0. Following the stochastic shortest path paradigm of AO* [Nilsson, 1982], AEMS maintains an upper bound $U_{\mathcal{T}}(b)$ and a lower bound $L_{\mathcal{T}}(b)$ on the optimal value function $V^*(b)$ at each node b in the tree.

At each expansion step, AEMS selects the fringe node that is expected to contribute most to reducing the error at the root. This selection is based on a heuristic that considers both the probability of reaching a node under the current best policy and the value uncertainty at that node.

Let $\hat{e}(b) = U(b) - L(b)$ be the estimated value gap, $d_{\mathcal{T}}(b, b_0)$ the depth of b from the root, and $h^b_{b_0}$ the sequence of actions and observations from b_0 to b. Let $\hat{\pi}_{\mathcal{T}}$ be the policy that chooses actions maximizing the upper bound, and $P(h^b_{b_0} \mid b_0, \hat{\pi}_{\mathcal{T}})$ be the probability of the history from b_0 to b by following the policy $\hat{\pi}_{\mathcal{T}}$. The node selected for expansion is:

$$\tilde{b}(\mathcal{T}) = \arg\max_{b \in \mathcal{F}(\mathcal{T})} \gamma^{d_{\mathcal{T}}(b, b_0)} P(h^b_{b_0} \mid b_0, \hat{\pi}_{\mathcal{T}}) \hat{e}(b) \tag{4.1}$$

This prioritization strategy allows AEMS to focus computational resources on regions of the search space that are expected to yield the most significant reductions in error, thus enabling it to provide high-quality solutions within limited time constraints.

4.3 Related Work

4.3.1 Heuristic search

Other heuristic search algorithms are notable in the realm of online planning for POMDPs. The approach by Satia and Lave [1973] employs a branch-and-bound strategy and uses a heuristic similar to AEMS, where fringe beliefs are weighted by their likelihood of observation. A key distinction, however, is that all non-dominated actions are treated as equally probable. The BI-POMDP algorithm [Washington, 1997] is more closely aligned with AO* [Nilsson, 1968] and AEMS(-SR), as it focuses on fringe nodes accessible via a greedy policy that selects actions maximizing the upper bound—akin to our Equation 4.10. Unlike AEMS, BI-POMDP does not apply additional weighting on the probability of reaching fringe nodes, and instead prioritizes node expansion based solely on the size of the error gap.

In this work, we chose to extend AEMS, given its demonstrated efficiency in prior evaluations [Ross et al., 2008]. However, the core ideas of our approach are not limited to AEMS and could be adapted to other heuristic search algorithms.

4.3.2 State requests in POMDPs

The idea of allowing an agent to request state information has seen growing interest. Bellinger et al. [2021] present the AMRL framework, in which the agent can request the *next* state at a cost. This differs from our POMDP-SR setting in two important ways: state access is delayed by one timestep, and the action space is doubled instead of decoupling state requests from environment actions. Their reinforcement learning algorithm, AMRL-Q, is inspired by Q-learning [Watkins and Dayan, 1992] and restricts policies to be state-conditioned. This approach uses a single sampled state for learning and action selection, which is known to be suboptimal as it disregards belief or history dependencies.

ACNO-MDPs [Nam et al., 2021] build on AMRL by omitting observations entirely when the agent does not request the state, simplifying the belief update process. Two reinforcement learning algorithms are proposed: *observe-before-planning*, where the agent observes the state at every timestep for a fixed duration to learn an MDP model, and then applies POMCP (Silver and Veness 2010, Section 2.3.2) on the original POMDP; and

observe-while-planning, where the agent decides dynamically when to request the state and uses either POMCP or DVRL [Igl et al., 2018] for planning and model improvement.

Krale et al. [2023] further investigate ACNO-MDPs by introducing a heuristic-based strategy to decide when to request the state. While these works address similar questions of balancing observability and cost, they differ significantly from our setting. Our approach assumes a known model and focuses on online planning with formal guarantees, including ε-optimality. Additionally, our formulation separates the decision of whether to request the state from the choice of the environment action, avoiding the doubling of the action space that occurs in AMRL.

4.4 Framework

We now formally introduce the *Partially Observable Markov Decision Process with State Requests (POMDP-SR)*, the central model studied in this chapter. This framework extends the standard POMDP by allowing the agent to request the underlying state at a cost, thereby turning access to full information into a strategic decision. We first define POMDP-SRs and highlight how their optimal policies may differ fundamentally from both MDPs and POMDPs. We then present an equivalent POMDP formulation, which enables the application of existing planning techniques but also exposes the computational challenges that motivate the development of AEMS-SR.

4.4.1 POMDP with State Requests

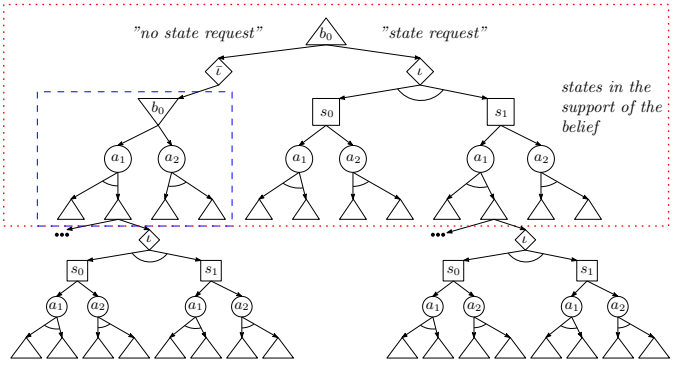
Definition 4.1: POMDP with State Requests

A *Partially Observable Markov Decision Process with State Requests* (POMDP-SR) is defined as a tuple $\mathcal{P}_{\text{SR}} = \langle \mathcal{P}, c \rangle$, where $\mathcal{P}$ is a standard POMDP and $c > 0$ is the cost associated with requesting the true underlying state. At each timestep, the agent first chooses whether to request the state, denoted by a binary decision $a^{\iota} \in \{\bar{\iota}, \iota\}$. If $a^{\iota} = \iota$, the true state s_t is revealed immediately incurring a cost c. Then, regardless of the request decision, the agent selects an environment action a^{t}.

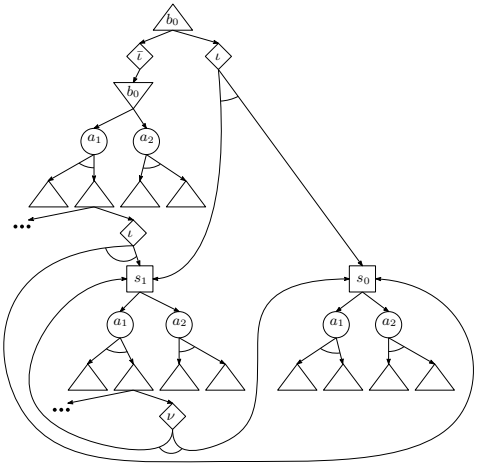
Property 4.1: Optimal Action Divergence in POMDP-SR

There exist environments in which the optimal action in the MDP and the POMDP coincide, but the optimal action in the corresponding POMDP-SR differs.

This arises from the fact that the agent operates under the anticipation of potential future state requests. In such contexts, a suboptimal action in the MDP and the POMDP can



(a) AO Tree. The box with red dashes corresponds to an expansion in a $\mathcal{P}_{\text{SR}}$. For comparison, the box with blue dashes would match the expansion in a $\mathcal{P}$.



(b) AO Graph

Figure 4.1: Tree and Graph representation after three successive expansions (the expanded beliefs are in green). Beliefs before selecting state request depicted by upward triangles, beliefs before environmental action by downward triangles, (not-)request state actions by diamonds, environmental actions by circles, and corner beliefs by rectangles. Some nodes are hidden for readability.

become the optimal one in the POMDP-SR when combined with a future request the state, achieving a return that is sub-optimal for the MDP but significantly better than the one of a POMDP.

Property 4.1 highlights how the integration of state requests fundamentally shifts the dynamics of decision-making in POMDPs. Example 4.1 proves this property.

Example 4.1

To illustrate Property 4.1, Fig. 4.2 shows an environment where the MDP and POMDP optimal action is different than the POMDP-SR one for some c.

States that yield the same observation are grouped in dashed boxes. The agent starts in state s_0, and the available environment actions are a_1 (left) and a_2 (right).

- In the fully observable MDP, the optimal action is a_1, yielding a return of 10.
- In the POMDP, due to uncertainty over the initial state, the expected return of a_1 drops to 5, and this remains the optimal action.

Let us see how in POMDP-SR the agent can anticipate future state requests and how this changes the optimal action. If the agent takes action a_1, it can follow the optimal POMDP policy and achieve a return of 5 without requesting the state, or it needs to request the state twice to reach the 10 return, incurring a cost of $2c$. If the agent takes action a_2, it can obtain a return of 9 by requesting the state once, incurring a cost of c.

$$Q^{\text{SR}}(s_0, a_1) = \max(5, 10 - 2c)$$
$$Q^{\text{SR}}(s_0, a_2) = 9 - c$$

Hence, a_2 becomes optimal whenever:

$$9 - c > \max(5, 10 - 2c)$$

This inequality holds when $c \in [1, 4]$, in which case the optimal action in the POMDP-SR differs from both the MDP and the POMDP.

This example highlights the anticipatory reasoning in POMDP-SR: actions may be chosen not for their immediate reward, but for the strategic advantage they enable through future state access.

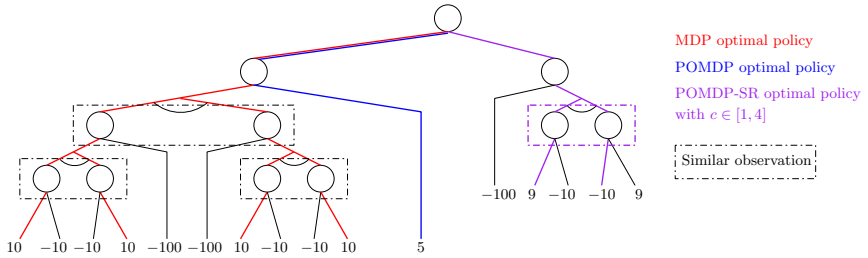


Figure 4.2: Example environment where the MDP and POMDP optimal action is a_1, but under POMDP-SR the optimal action is a_2 when $c \in [1, 4]$. Circles denote states; observations are indistinguishable within dashed boxes.

4.4.2 Equivalent POMDP

In this section, we introduce a transformation that converts a POMDP-SR into an equivalent POMDP to allow the use of existing POMDP planning techniques.

Definition 4.2: Equivalent POMDP

A POMDP-SR $\mathcal{P}_{\text{SR}} = \langle \mathcal{P}, c \rangle$ can be transformed into an equivalent POMDP $\mathcal{P}' = \langle \tilde{\mathcal{S}}, \tilde{\Omega}, \tilde{\mathcal{A}}, \tilde{\mathbf{A}}, \tilde{\mathbf{P}}, \tilde{\mathcal{O}}, \tilde{\mathcal{R}}, \tilde{\gamma} \rangle$ by separating the state request decision from the environment action, resulting in a two-phase model with variable action spaces. This transformation enables the use of standard POMDP planning techniques on POMDP-SRs.

The components of the equivalent POMDP are defined as follows:

- **State space:** $\tilde{\mathcal{S}} \equiv \{0, 1\} \times \mathcal{S}$ augments the original state space with a binary phase indicator. We write s^i to denote state s in phase $i \in \{0, 1\}$, where $i = 0$ denotes the state request phase and $i = 1$ the environment action phase.
- **Observation space:** $\tilde{\Omega} = \Omega \cup \mathcal{S} \cup \{o^*\}$ includes:
  - the original observations,
  - the full state space (used when the state is requested),
  - a special observation o^* that serves as a placeholder when the state is not requested.
- **Action space:** $\tilde{\mathcal{A}} = \{\iota, \bar{\iota}\} \cup \mathcal{A}$ contains state request decisions $\{\iota, \bar{\iota}\}$ (see Definition 4.1) and environment actions. The set of legal actions at a state

is defined by the function $\tilde{\mathbf{A}}$:

$$\tilde{\mathbf{A}}(s^0) = \{\iota, \bar{\iota}\} \qquad \tilde{\mathbf{A}}(s^1) = \mathcal{A}$$

Since all states in the support of a belief share the same phase i, this action legality can be naturally extended to beliefs.

- **Transition function:** The transition dynamics $\tilde{\mathbf{P}}$ are defined for $s, s' \in \mathcal{S}$ as:

$$\tilde{\mathbf{P}}(s'^0 \mid s^1, a) = \mathbf{P}(s' \mid s, a) \qquad \tilde{\mathbf{P}}(s'^1 \mid s^0, a) = 1_s(s')$$

$$\tilde{\mathbf{P}}(s'^i \mid s^i, a) = 0 \quad \text{(all other cases)}$$

where $1_s(s')$ is the indicator function returning 1 if $s' = s$, 0 otherwise. This structure ensures that the agent transitions from the request phase to the action phase and vice versa, and that the environment dynamics are preserved during the action phase.

- **Observation function:** $\tilde{\mathcal{O}}$ is defined for legal actions only (δ_x is a Dirac delta centered at x):

$$\tilde{\mathcal{O}}(s'^0, a) = \mathcal{O}(s', a) \qquad \tilde{\mathcal{O}}(s'^1, \iota) = \delta_{s'} \qquad \tilde{\mathcal{O}}(s'^1, \bar{\iota}) = \delta_{o^*}$$

- **Reward function:** $\tilde{\mathcal{R}}$ is defined over legal actions as:

$$\tilde{\mathcal{R}}(s^0, a) = -1_\iota(a) \cdot c/\sqrt{\gamma} \qquad \tilde{\mathcal{R}}(s^1, a) = \mathcal{R}(s, a)$$

- **Discount factor:** $\tilde{\gamma} = \sqrt{\gamma}$

Remark 4.1

An alternative to this two-phase model with variable action spaces would be to transform the action space to $\mathcal{A}' = \{\iota, \bar{\iota}\} \times \mathcal{A}$, effectively combining the state request decision and the environment action into a single joint action at each timestep. While this approach would avoid doubling the number of timesteps, it does not yield an equivalent model to a POMDP-SR due to the discounting factor. The transformation described above ensures that the structure and properties of the POMDP-SR are retained.

Notation

Similar to the state notation, we denote beliefs containing only states of one type as b^0 and b^1 for the request and environment action phases, respectively.

Equivalent POMDP Complexity

Transforming a POMDP-SR into its equivalent POMDP enables the use of classic POMDP planning algorithms, but this approach may prove inefficient. One inefficiency arises from the lack of support for variable action spaces in classic implementations, necessitating the use of deterrent penalties for illegal action which impacts the algorithm's complexity. For offline methods like PBVI, doubling the state space and augmenting the observation space to include the state space lead to a steep increase in complexity. Heuristic search algorithms like AEMS face an even more challenging situation as doubling the horizon in POMDP-SR and requiring a new sub-tree for every state in the support of the belief result in an exponential expansion of the search tree (Fig. 4.1a). This growth underscores the fundamental issue: current planning algorithms are ill-suited to effectively handle the unique complexities introduced by POMDP-SR scenarios.

4.5 Online planning: AEMS-SR

In this section, we present our new method Anytime Error Minimization Search for POMDP-SRs (AEMS-SR) which adapts AEMS to our framework.

As outlined in Section 4.4, the introduction of state requests leads to an exponential increase in the size of the search tree. This growth is primarily due to two factors: the doubling of timesteps and the generation of a new subtree for each state in the belief's support. Consequently, each expansion of belief in a POMDP-SR, illustrated by the red box in Fig. 4.1a, adds $(1+|\text{supp}(b)|)\cdot|\mathcal{A}|\cdot|\Omega|$ nodes, demanding an equal number of belief updates. This is in stark contrast to classic POMDPs, where only $|\mathcal{A}|\cdot|\Omega|$ nodes are added per expansion, as illustrated by the blue box in Fig. 4.1a.

Upon examining the search tree in POMDP-SR scenarios, illustrated in Fig. 4.1a, we observe that many nodes are similar. This redundancy is particularly pronounced when subsequent beliefs overlap significantly, such as when the agent has to deal with position uncertainty or potential action failure. In these cases, the search tree often contains identical subtrees that are redundantly expanded, impairing search efficiency.

The challenge of repetitive subtree expansions is not unique to POMDP-SR; it is a known issue in both POMDPs and MDPs. Techniques like transposition tables [Childs et al., 2008] have been used to address this problem, offering computational trade-offs that can

be beneficial in certain environments but are less practical for continuous spaces such as beliefs.

To deal with these, AEMS-SR employs a rooted cyclic graph, denoted as $\mathcal{G}$, with the current belief b_0 as its root, for the search replacing the conventional tree structure. A rooted cyclic graph is defined as a regular cyclic graph where every node can be reached from its root, and where the root does not have any parents. This shift to a cyclic graph necessitates the development of novel heuristic and algorithmic solutions to adeptly manage the added complexities of cyclicity.

4.5.1 AEMS-Loop

We first introduce AEMS-Loop, the extension of AEMS to cyclic graphs, and theoretically prove its completeness and $\varepsilon-$optimiality, meaning that the algorithm will always return a solution that is $\varepsilon-$close to the optimal solution given enough time.

Similar to other online tree search algorithms, we rely on upper and lower bounds, denoted as $U(b)$ and $L(b)$, of the optimal value function $V^*(b)$ that are computed offline.

$$U(b) \geq V^*(b) \geq L(b), \forall b \in \mathcal{B}$$

These bounds are used to initialize the values of the fringe nodes $\mathcal{F}(\mathcal{G})$ of the graph $\mathcal{G}$, which are then propagated to the root b_0 through dynamic programming by recursively applying the Bellman equations (Definition 2.4) much like in Value Iteration (Section 2.2.2). We denote $U_\mathcal{G}(b)$ and $L_\mathcal{G}(b)$ as the upper and lower bound values of a belief b in the graph $\mathcal{G}$ obtained through this process. The following equations define how to compute them with τ being the belief update function (Property 2.3).

$$U_\mathcal{G}(b,a) = R(b,a) + \gamma \sum_{o \in \Omega} P(o \mid b,a) U_\mathcal{G}(\tau(b,a,o)) \tag{4.2}$$

$$L_\mathcal{G}(b,a) = R(b,a) + \gamma \sum_{o \in \Omega} P(o \mid b,a) L_\mathcal{G}(\tau(b,a,o)) \tag{4.3}$$

$$U_\mathcal{G}(b) = \begin{cases} U(b) & \text{if } b \in \mathcal{F}(\mathcal{G}) \\ \max_{a \in \mathcal{A}} U_\mathcal{G}(b,a) & \text{otherwise} \end{cases} \tag{4.4}$$

$$L_\mathcal{G}(b) = \begin{cases} L(b) & \text{if } b \in \mathcal{F}(\mathcal{G}) \\ \max_{a \in \mathcal{A}} L_\mathcal{G}(b,a) & \text{otherwise} \end{cases} \tag{4.5}$$

Therefore successive expansions of the graph allows to improve the bounds and reduce the error gap at the root: $e_\mathcal{G}(b_0) = U_\mathcal{G}(b_0) - L_\mathcal{G}(b_0)$.

Working with a cyclic graph introduces the possibility of multiple paths, potentially an infinite number, between the root b_0 and any fringe node $b \in \mathcal{F}(\mathcal{G})$. This requires to define the following objects.

Definition 4.3: Path and Path Set

We define $\Phi_{\mathcal{G}}(b_0, b)$ as the set of paths in $\mathcal{G}$ that start from b_0 and terminate at b. A path $h \in \Phi_{\mathcal{G}}(b_0, b)$ is a sequence of tuples $(b_i, a_i, o_{i+1})_{i<T}$.

Given a policy π, the probability of following a specific path is:

$$P(h \mid b_0, \pi) = \prod_{i=0}^{T-1} P(o_{i+1} \mid b_i, a_i) \cdot \pi(a_i \mid b_i) \tag{4.6}$$

Definition 4.4: Cumulative Probability

We define $\Psi_{\pi}^{\mathcal{G}}(b_0, b)$ as the sum over all possible paths between the root b_0 and a fringe $b \in \mathcal{F}(\mathcal{G})$ of the probability of observing the path discounted by the length of the path $d(h)$. We drop the subscript $\mathcal{G}$ when the dependency is clear.

$$\Psi_{\pi}^{\mathcal{G}}(b_0, b) = \sum_{h \in \Phi_{\mathcal{G}}(b_0, b)} \gamma^{d(h)} \mathbf{P}(h \mid b_0, \pi) \tag{4.7}$$

This generalization is necessary to adapt AEMS from a tree-based to a graph-based setting, as beliefs may be reachable through multiple, differently weighted paths.

Theorem 4.1

In any rooted graph $\mathcal{G}$ with root b_0 where values are computed according to Eq. 4.5 using a lower bound value function L with error $e(b) = V^*(b) - L(b)$, the error on the root belief state is bounded by:

$$\hat{e}_{\mathcal{G}}(b_0) = V^*(b) - L_{\mathcal{G}}(b_0) \leq \sum_{b \in \mathcal{F}(\mathcal{G})} \Psi_{\pi^*}(b_0, b) e(b) \tag{4.8}$$

where $e(b) = V^*(b) - L(b)$.

Proof sketch. We use a similar proof as AEMS on an enrolling of the tree of size n to obtain an upper bound composed of two elements:

(i) the discounted probabilities of observing a path of size at most n from the root to one of the replicas of an element in $\mathcal{F}(\mathcal{G})$;

(ii) the discounted probabilities of other paths which are of size n.

By making $n \to +\infty$, (i) converges to the term in the theorem and (ii) to 0. □

Proof. Consider an arbitrary node $b \in \mathcal{G} \setminus \mathcal{F}(\mathcal{G})$ in a graph $\mathcal{G}$ that is not a fringe, and $a_b^* = \arg\max_a Q^*(b, a)$ the optimal action.
By definition $\gamma \in [0, 1)$.

If $\gamma = 0$, then $L(b) = V^*(b) = r\big(b, a_b^*\big)$ and $e_{\mathcal{G}}(b) = 0$ which conclude the proof. Therefore let us focus on $\gamma \in (0, 1)$.

By definition of $L_{\mathcal{G}}$ we have $L_{\mathcal{G}}\big(b, a_b^*\big) \leq L_{\mathcal{G}}(b)$.

$$
\begin{aligned}
e_{\mathcal{G}}(b) &= V^*(b_0) - L_{\mathcal{G}}(b_0) \\
&\leq V^*(b_0) - L_{\mathcal{G}}(b_0, a_b^*) \\
&\leq r\big(b, a_b^*\big) + \gamma \sum_{o \in \Omega} P\big(o|b, a_b^*\big) V^*\big(\tau\big(b, a_b^*, o\big)\big) \\
&\qquad - \left(r\big(b, a_b^*\big) + \gamma \sum_{o \in \Omega} P\big(o|b, a_b^*\big) L_{\mathcal{G}}\big(\tau\big(b, a_b^*, o\big)\big) \right) \\
&\leq \gamma \sum_{o \in \Omega} P\big(o|b, a_b^*\big) e_{\mathcal{G}}\big(\tau\big(b, a_b^*, o\big)\big)
\end{aligned}
$$

This result in the following inequality:

$$
e_{\mathcal{G}}(b) \leq \begin{cases} e(b) & \text{if } b \in \mathcal{F}(\mathcal{G}) \\ \gamma \sum_{o \in \Omega} P\big(o|b, a_b^*\big) e_{\mathcal{G}}\big(\tau\big(b, a_b^*, o\big)\big) & \text{otherwise} \end{cases} \tag{4.9}
$$

Let us now consider $n \in \mathbb{N}^*$, and unroll the rooted graph $\mathcal{G}$ into a tree $\mathcal{T}_n$ with root b_0 and with maximum depth n. We define the following elements:

- Similar to our Definition 4.3 of Φ, we define, for all $b' \in \mathcal{G}$, $\Phi_n(b_0, b')$ the set of paths starting from b_0 and finishing in any of the replicas of b' in $\mathcal{T}_n$. It is important to note that the length of the paths in Φ_n are bounded by n. We have

$$
\lim_{n \to +\infty} \Phi_n\big(b_0, b'\big) = \Phi(b_0, b)
$$

  the (possibly infinite) set of paths between b_0 and b' in G.
- For any $b' \in \mathcal{G}$ and any policy π,

$$
\Psi_\pi^n\big(b_0, b'\big) = \sum_{h \in \Phi_n(b_0, b')} \gamma^{d(h)} \mathbf{P}(h|b_0, \pi)
$$

- $\mathcal{F}_n$ as the fringes nodes of $\mathcal{T}_n$
- $\mathcal{F}_n^{\mathcal{G}}$ is the set of nodes of $\mathcal{F}_n$ that are replicas of a node in $\mathcal{F}(\mathcal{G})$
- $\bar{\mathcal{F}}_n^{\mathcal{G}} = \mathcal{F}_n \setminus \mathcal{F}_n^{\mathcal{G}}$
- $e_n = e_{\mathcal{T}_n}$

We note that Equation 4.9 hold for any graph including $\mathcal{T}_n$. Therefore, by solving the recurrence in $\mathcal{T}_n$ for b_0 we obtain (as in the proof of Theorem 1 of AEMS):

$$\begin{aligned} e_n(b_0) &\leq \sum_{b \in \mathcal{F}_n} \gamma^{d(b_0,b)} P\left(h_{b_0}^b | b_0, \pi^*\right) e(b) \\ &\leq \sum_{b \in \mathcal{F}_n^{\mathcal{G}}} \gamma^{d(b_0,b)} P\left(h_{b_0}^b | b_0, \pi^*\right) e(b) + \sum_{b \in \bar{\mathcal{F}}_n^{\mathcal{G}}} \gamma^{d(b_0,b)} P\left(h_{b_0}^b | b_0, \pi^*\right) e(b) \\ &\leq \sum_{b \in \mathcal{F}(\mathcal{G})} \Psi_{\pi^*}^n(b_0, b) e(b) + \sum_{b \in \bar{\mathcal{F}}_n^{\mathcal{G}}} \gamma^{d(b_0,b)} P\left(h_{b_0}^b | b_0, \pi^*\right) e(b) \end{aligned}$$

We note that for all $b \in \bar{\mathcal{F}}_n^{\mathcal{G}}$ the associated history h in $\mathcal{T}_n$ is of length n. Indeed, if the history size was shorter than n, b would also be a fringe in $\mathcal{G}$ which is impossible by definition of $\bar{\mathcal{F}}_n^{\mathcal{G}}$. It follows that:

$$\begin{aligned} e_n(b_0) &\leq \sum_{b \in \mathcal{F}(\mathcal{G})} \Psi_{\pi^*}^n(b_0, b) e(b) + \gamma^n \sum_{b \in \bar{\mathcal{F}}_n^{\mathcal{G}}} P\left(h_{b_0}^b | b_0, \pi^*\right) e(b) \\ &\leq \sum_{b \in \mathcal{F}(\mathcal{G})} \Psi_{\pi^*}^n(b_0, b) e(b) + \gamma^n \sup_{b'} e\left(b'\right) \\ &\qquad \text{(the sup exists because } V^* \text{ is bounded above and } L \text{ bellow)} \end{aligned}$$

In the limit, when $n \to +\infty$ we have $e_n \to e_{\mathcal{G}}$, $\Psi^n \to \Psi$, and $\gamma^n \to 0$ as $\gamma \in (0, 1)$, leading to

$$e_{\mathcal{G}}(b_0) \leq \sum_{b \in \mathcal{F}(\mathcal{G})} \Psi_{\pi^*}(b_0, b) e(b)$$

□

Theorem 4.1 gives an upper bound on the contribution of each fringe node to the error at the root, extending AEMS's Theorem 1 [Ross et al., 2007] to cyclic graphs.

Remark 4.2

Assuming a tree structure, which implies $|\Phi(b_0, b)| = 1$ for any fringe belief $b \in \mathcal{F}(\mathcal{T})$, recovers the original theorem of Ross et al. [2007].

Similar to AEMS, this theorem provides a robust method for choosing the next belief to expand to rapidly minimize root error: prioritize expanding the belief with the greatest estimated contribution $\arg\max_{b \in \mathcal{F}(\mathcal{G})} \Psi_{\pi^*}(b_0, b) e(b)$.

However, as the optimal policy π^* and value function V^* are unknown, we need to approximate them to compute $\Psi_{\pi^*}(b_0, b)$ and $e(b)$. We denote the approximation of π^* as $\hat{\pi}_{\mathcal{G}}$. As in AEMS, we employ the following two approximations:

$$\hat{\pi}_{\mathcal{G}}(b, a) = \begin{cases} 1 & \text{if } a = \arg\max_{a'} U_{\mathcal{G}}(b, a) \\ 0 & \text{otherwise} \end{cases} \tag{4.10}$$

$$\hat{e}(b) = U(b) - L(b) \geq e(b) \tag{4.11}$$

While other approximations $\hat{\pi}_{\mathcal{G}}$ are possible, we selected the one presented in Equation 4.10 because of its empirical performance in AEMS and its simplicity. Using those two approximations, we can leverage Theorem 4.1 to define the following heuristic.

Definition 4.5: AEMS-Loop Heuristic

The AEMS-Loop heuristic for next belief expansion is defined as follows:

$$\tilde{b}(\mathcal{G}) = \arg\max_{b \in \mathcal{F}(\mathcal{G})} \Psi_{\hat{\pi}_{\mathcal{G}}}(b_0, b) \hat{e}(b) \tag{4.12}$$

In the rest of this section, we will show that this heuristic is sound and leads to a complete and ε−optimal algorithm.

Definition 4.6

We define the approximate error contribution of a fringe node $b \in \mathcal{F}(\mathcal{G})$ on the value at the root b_0 as

$$E(b, b_0, \mathcal{G}) = \Psi_{\hat{\pi}_{\mathcal{G}}}(b_0, b) \hat{e}(b)$$

Lemma 4.1

In any graph $\mathcal{G}$, the approximate error contribution $E(b, b_0, \mathcal{G})$ of a belief node b is bounded by $E(b, b_0, \mathcal{G}) \leq \gamma^{d_b} \sup_{b'} \hat{e}(b')$ with $d_b = \min_{h \in \Phi_{\mathcal{G}}(b_0, b)} d(h)$.

Proof.

$$\begin{aligned} E(b, b_0, \mathcal{G}) &= \Psi_{\hat{\pi}_\mathcal{G}}(b_0, b)\hat{e}(b) \\ &= \sum_{h \in \Phi(b_0, b)} \gamma^{d(h)} \mathbf{P}(h \mid b_0, \hat{\pi}_\mathcal{G})\hat{e}(b) \\ &\leq \gamma^{d_b} \sum_{h \in \Phi(b_0, b)} \mathbf{P}(h \mid b_0, \hat{\pi}_\mathcal{G})\hat{e}(b) \\ &\leq \gamma^{d_b} \sup_{b'} \hat{e}\left(b'\right) \sum_{h \in \Phi(b_0, b)} \mathbf{P}(h \mid b_0, \hat{\pi}_\mathcal{G}) \\ &\leq \gamma^{d_b} \sup_{b'} \hat{e}\left(b'\right) \end{aligned}$$

□

Definition 4.7: Accessible Fringe Nodes and Possible Histories

We define:

- the set of accessible fringe nodes of a graph $\mathcal{G}$ under $\hat{\pi}_\mathcal{G}$ as

$$\hat{\beta}(\mathcal{G}) = \{b \mid b \in \mathcal{F}(\mathcal{G}) \text{ and } \exists h \in \Phi_\mathcal{G}(b_0, b), P(h \mid b_0, \hat{\pi}_\mathcal{G}) > 0\}$$

- the set of possible histories

$$\hat{\zeta}(b_0, \mathcal{G}) = \{h \mid P(h \mid b_0, \hat{\pi}_\mathcal{G}) > 0\}$$

- the set of possible histories between the root belief b_0 and b, for any $b \in \mathcal{G}$, as

$$\Phi^p_\mathcal{G}(b_0, b) = \{h^b_{b_0} \mid h^b_{b_0} \in \Phi_\mathcal{G}(b_0, b) \bigcap \hat{\zeta}(b_0, \mathcal{G})\}$$

Definition 4.8: Observation Probability

For all histories $h = (b_i, a_i, o_{i+1})_{i<T}$, where $T = d(h)$ is the length of the history, we define the observation probability

$$P(h \mid b_0) = \Pi_{i=0}^{T-1} P(o_{i+1} \mid b_i, a_i)$$

Lemma 4.2

Given U bounded above and L bounded bellow such as $U(b) \geq V^*(b) \geq L(b)$, and $\hat{e}(b) = U(b) - L(b)$ for all $b \in \mathcal{B}$, then for any graph $\mathcal{G}$, $\varepsilon > 0$ and $D \in \mathbb{N}^*$ such that $\gamma^D \sup_b \hat{e}(b) \leq \varepsilon$, if for all $b \in \hat{\beta}(\mathcal{G})$ and for all $h \in \Phi^p_{\mathcal{G}}(b_0, b)$, either $d(h) > D$ or there exists an ancestor $b' \in h$ such that $\hat{e}_{\mathcal{G}}(b') \leq \varepsilon$, then $\hat{e}_{\mathcal{G}}(b_0) \leq \varepsilon$.

Proof. For any graph $\mathcal{G}$, and any belief b that is not a fringe belief node $b \in \mathcal{G} \setminus \mathcal{F}(\mathcal{G})$. We define as $\hat{a}^{\mathcal{G}}_b = \arg\max_{a \in \mathcal{A}} U_{\mathcal{G}}(b, a)$.

$$
\begin{aligned}
\hat{e}_{\mathcal{G}}(b) &= U_{\mathcal{G}}(b) - L_{\mathcal{G}}(b) \\
&\leq U_{\mathcal{G}}\big(b, \hat{a}^{\mathcal{G}}_b\big) - L_{\mathcal{G}}\big(b, \hat{a}^{\mathcal{G}}_b\big) \\
&\leq r\big(b, \hat{a}^{\mathcal{G}}_b\big) + \gamma \sum_{o \in \Omega} P\big(o \mid b, \hat{a}^{\mathcal{G}}_b\big) U_{\mathcal{G}}\big(\tau\big(b, \hat{a}^{\mathcal{G}}_b, o\big)\big) \\
&\qquad - \left(r\big(b, \hat{a}^{\mathcal{G}}_b\big) + \gamma \sum_{o \in \Omega} P\big(o \mid b, \hat{a}^{\mathcal{G}}_b\big) L_{\mathcal{G}}\big(\tau\big(b, \hat{a}^{\mathcal{G}}_b, o\big)\big)\big) \right) \\
&\leq \gamma \sum_{o \in \Omega} P\big(o \mid b, \hat{a}^{\mathcal{G}}_b\big) \hat{e}_{\mathcal{G}}\big(\tau\big(b, \hat{a}^{\mathcal{G}}_b, o\big)\big)
\end{aligned}
$$

We obtain the following upper bound for $b \in \mathcal{G}$ on $\hat{e}_{\mathcal{G}}(b)$:

$$
\hat{e}_{\mathcal{G}}(b) \leq \begin{cases} \hat{e}(b) & \text{if } b \in \mathcal{F}(\mathcal{G}) \\ \varepsilon & \text{if } \hat{e}_{\mathcal{G}}(b) \leq \varepsilon \\ \gamma \sum_{o \in \Omega} P\big(o \mid b, a^*_b\big) \hat{e}_{\mathcal{G}}\big(\tau\big(b, \hat{a}^{\mathcal{G}}_b, o\big)\big) & \text{otherwise} \end{cases} \tag{4.13}
$$

We define :

- $\hat{e}_{\mathcal{G}}\big(h^b_{b_0}\big) = \hat{e}_{\mathcal{G}}(b)$
- $A(\mathcal{G})$ is the set of possible histories $h^b_{b_0} \in \hat{\zeta}(b_0, \mathcal{G})$ of length $d\big(h^b_{b_0}\big) < D$ such that $\hat{e}_{\mathcal{G}}(b) \leq \varepsilon$.
- $B(\mathcal{G})$ is the set of possible histories $h^b_{b_0} \in \hat{\zeta}(b_0, \mathcal{G})$ of length $d\big(h^b_{b_0}\big) < D$ such that $b \in \mathcal{F}(\mathcal{G})$, and that for all intermediary $b' \in h^b_{b_0}$, the partial history $h^{b'}_{b_0} \notin A(\mathcal{G})$.
- $C(\mathcal{G})$ is the set of all possible histories $h^b_{b_0} \in \hat{\zeta}(b_0, \mathcal{G})$ of size at least D, that do not belong to $B(\mathcal{G})$ and for all intermediary $b' \in h^b_{b_0}$, the partial history $h^{b'}_{b_0} \notin A(\mathcal{G})$.

As for all $b \in \hat{\beta}(\mathcal{G})$ and for all $h \in \Phi^p_{\mathcal{G}}(b_0, b)$ either $d(h) > D$ or there exists an ancestor $b' \in h$ such that $\hat{e}_{\mathcal{G}}(b') \leq \varepsilon$, $B(\mathcal{G})$ is empty.

By unfolding the recurrence above we obtain:

$$\begin{aligned}
\hat{e}_{\mathcal{G}}(b_0) &= \sum_{h^b_{b_0} \in A(\mathcal{G})} \gamma^{d\left(h^b_{b_0}\right)} P\left(h^b_{b_0}|b_0\right) \hat{e}_{\mathcal{G}}(b) + \sum_{h^b_{b_0} \in C(\mathcal{G})} \gamma^{d\left(h^b_{b_0}\right)} P\left(h^b_{b_0}|b_0\right) \hat{e}_{\mathcal{G}}(b) \\
&\leq \varepsilon \sum_{h^b_{b_0} \in A(\mathcal{G})} P\left(h^b_{b_0}|b_0\right) + \sum_{h^b_{b_0} \in C(\mathcal{G})} \gamma^{d\left(h^b_{b_0}\right)} P\left(h^b_{b_0}|b_0\right) \hat{e}_{\mathcal{G}}(b) \\
&\leq \varepsilon \sum_{h^b_{b_0} \in A(\mathcal{G})} P\left(h^b_{b_0}|b_0\right) + \gamma^D \sup_b \hat{e}(b) \sum_{h^b_{b_0} \in C(\mathcal{G})} P\left(h^b_{b_0}|b_0\right) \\
&\leq \varepsilon \sum_{h^b_{b_0} \in A(\mathcal{G})} P\left(h^b_{b_0}|b_0\right) + \varepsilon \sum_{h^b_{b_0} \in C(\mathcal{G})} P\left(h^b_{b_0}|b_0\right) \\
&\leq \varepsilon \sum_{h^b_{b_0} \in A(\mathcal{G}) \cup C(\mathcal{G})} P\left(h^b_{b_0}|b_0\right) \\
&\leq \varepsilon
\end{aligned}$$

□

Theorem 4.2

Given U bounded above and L bounded below, with the property that $\forall b \in \mathcal{B}$, $U(b) \geq V^*(b) \geq L(b)$, and let $\hat{e}(b) = U(b) - L(b)$. If $\gamma \in [0, 1)$ and $\inf_{b, \mathcal{G} | \hat{e}_{\mathcal{G}}(b) > \varepsilon} \hat{\pi}_{\mathcal{G}}\left(b, \hat{a}^{\mathcal{G}}_b\right) > 0$ for $\hat{a}^{\mathcal{G}}_b = \arg\max_{a \in \mathcal{A}} U_{\mathcal{G}}(b, a)$, then the AEMS-Loop algorithm using heuristic $\tilde{b}(\mathcal{G})$ is complete and ε-optimal.

Proof. Consider an arbitrary $\varepsilon > 0$ and the current root belief b_0.

If $\gamma = 0$, then after one expansion $\hat{e}_{\mathcal{G}}(b_0) = 0$ since $U_{\mathcal{G}}(b_0) = L_{\mathcal{G}}(b_0) = \max_{a \in \mathcal{A}} r(b_0, a)$. And therefore AEMS-Loop is complete and $\varepsilon-$optimal.

Lets focus on $\gamma \in (0, 1)$.
Because U is bounded above and L bellow, $\sup_b \hat{e}(b)$ exists and there exists $D \in \mathbb{N}$ such that $\gamma^D \sup_b \hat{e}(b) < \varepsilon$.
We define the following elements:

- $\mathcal{A}^{\mathcal{G}}\left(h^b_{b_0}\right)$ the set of ancestors beliefs of b in the history $h^b_{b_0}$
- $\mathcal{A}^{\mathcal{G}}(b) = \bigcup_{h^b_{b_0} \in \Phi^p_{\mathcal{G}}(b_0, b)} \mathcal{A}^{\mathcal{G}}\left(h^b_{b_0}\right)$ the set of ancestors beliefs of b across possible histories.
- $\hat{e}^{min}_{\mathcal{G}}(A) = \min_{b \in A} \hat{e}_{\mathcal{G}}(b)$ for any finite set of beliefs A.

- $\mathcal{G}_b = \{\mathcal{G} \mid \mathcal{G} \text{ is finite}, b \in \hat{\beta}(b_0, \mathcal{G}), \max_{h \in \Phi^p_{\mathcal{G}}(b_0, b)} \hat{e}^{min}_{\mathcal{G}}\left(\mathcal{A}^{\mathcal{G}}(h)\right) > \varepsilon\}$
  Intuitively, $\mathcal{G}_b$ is the set of finite graphs $\mathcal{G}'$ with root b_0 for which b is an accessible fringe node, i.e. that can be attained with non zero probability under the policy $\hat{\pi}_{\mathcal{G}'}$, and for which there exist an history $h^b_{b_0}$ such that all the ancestors beliefs b' in that history have an approximate error gap $\hat{e}_{\mathcal{G}'}(b') > \varepsilon$. The existence of the max relies on the graph being finite.
- $\mathcal{B} = \{b \mid \hat{e}(b) \inf_{\mathcal{G} \in \mathcal{G}_b} \sum_{h \in \Phi_{\mathcal{G}}(b_0, b) | d(h) \leq D} P\left(h^b_{b_0} \mid b_0, \hat{\pi}_{\mathcal{G}}\right) > 0\}$
  The assumption $\inf_{b, \mathcal{G} | \hat{e}_{\mathcal{G}}(b) > \varepsilon} \hat{\pi}_{\mathcal{G}}\left(b, \hat{a}^{\mathcal{G}}_b\right) > 0$ ensures that $\mathcal{B}$ contains all the beliefs states b within depth D such that (i) $\hat{e}(b) > 0$, (ii) there exists a finite graph $\mathcal{G}$ where $b \in \hat{\beta}(b_0, \mathcal{G})$ and for which there exists an history $h^b_{b_0} \in \Phi^p_{\mathcal{G}}(b_0, b)$ such that all ancestors $b' \in \mathcal{A}\left(h^b_{b_0}\right)$ have $\hat{e}_{\mathcal{G}}(b') > \varepsilon$.

As there are only a finite number of beliefs for which there exists an history of size smaller than D, $\mathcal{B}$ is finite. This allows us to define

$$E_{min} = \min_{b \in \mathcal{B}} \hat{e}(b) \inf_{\mathcal{G} \in \mathcal{G}_b} \sum_{h^b_{b_0} \in \Phi_{\mathcal{G}}(b_0, b)} \gamma^{d\left(h^b_{b_0}\right)} P\left(h^b_{b_0} | b_0, \hat{\pi}_{\mathcal{G}}\right)$$

By construction $E_{min} > 0$.
We also know that for any graph $\mathcal{G}$, all beliefs $b \in \mathcal{B} \cap \hat{\beta}(b_0, \mathcal{G})$ have an approximate error contribution $E(b, b_0, \mathcal{G}) \geq E_{min}$

$$E(b, b_0, \mathcal{G}) = \Psi_{\hat{\pi}_{\mathcal{G}}}(b_0, b)\hat{e}(b) = \sum_{h^b_{b_0} \in \Phi(b_0, b)} \gamma^{d\left(h^b_{b_0}\right)} P\left(h^b_{b_0} | b_0, \hat{\pi}_{\mathcal{G}}\right) \hat{e}(b) \geq E_{\min}$$

As $\gamma \in (0, 1)$ and $E_{min} > 0$, there exist $D' \in \mathbb{N}^+$ such that $\gamma^{D'} \sup_b \hat{e}(b) < E_{min}$. Therefore, we know from Lemma 4.1 that AEMS-Loop cannot expand any node of depth D' or more before expanding a graph $\mathcal{G}$ where $\mathcal{B} \cap \hat{\beta}(b_0, \mathcal{G}) = \emptyset$.
As there exist a finite number of belief nodes for which an history starting from b_0 of length at most D' exists, it is clear that AEMS-Loop will reach such a graph $\mathcal{G}$ after a finite number of expansions.
Since, for this graph $\mathcal{G}$, $\mathcal{B} \cap \hat{\beta}(b_0, \mathcal{G}) = \emptyset$ we have that for all beliefs $b \in \hat{\beta}(b_0, \mathcal{G})$ the possible histories $h^b_{b_0} \in \Phi^p_{\mathcal{G}}(b_0, b)$ with length $d\left(h^b_{b_0}\right) \leq D$ have

$$\hat{e}^{min}_{\mathcal{G}}\left(\mathcal{A}^{\mathcal{G}}\left(h^b_{b_0}\right)\right) < \varepsilon$$

Therefore, Lemma 4.2 ensures that $\hat{e}_{\mathcal{G}}(b_0) < \varepsilon$ and consequently AEMS-Loop will terminate with an ε-optimal solution in a finite number of expansions as $\hat{e}_{\mathcal{G}}$ is an upper bound of $e_{\mathcal{G}}$. □

Theorem 4.2 establishes the completeness and $\varepsilon-$optimality of AEMS-Loop for any policy $\hat{\pi}_{\mathcal{G}}$ assigning non-zero probability to the upper-bound maximizing action, such as the one defined in Eq.4.10.

4.5.2 Algorithm

This subsection explains how AEMS-SR (Alg. 3), a practical implementation of AEMS-Loop adapted to POMDP-SR, works in practice.

While the graph structure enables consolidating identical belief nodes to prevent redundant work, fully implementing this strategy would require comparing each new belief against all existing beliefs in the current graph. Such an approach would lead to scalability issues similar to those that render graphs less practical in general MDPs and POMDPs. To maintain tractability while still achieving our main goal of mitigating the exponential growth in the search tree, we restrict the capacity for multiple parents to corner beliefs (Section 4.2.1).

AEMS-SR starts with a graph $\mathcal{G}$ containing only the root belief b_0. The algorithm then iterates through the following steps until the time limit is reached:

1. Find the fringe belief $\tilde{b}(\mathcal{G}) \in \mathcal{F}(\mathcal{G})$ that maximizes Eq. 4.12, this corresponds to Alg. 4
2. Update the graph $\mathcal{G}$ by expanding the belief $\tilde{b}(\mathcal{G})$, for the request the state action ι we reuse the existing corner beliefs creating a graph similar to Fig. 4.1b
3. Update the upper and lower bound value to match Eq.4.4-4.5

Algorithm 3 shows the pseudocode of AEMS-SR.

Definition 4.9: Origin and Unique Path

For all fringe beliefs $b \in \mathcal{F}(\mathcal{G})$, we define the *origin* function $\xi_{\mathcal{G}}(b)$, which returns the first ancestor among the corner beliefs, or b_0 if such an ancestor does not exist. Additionally, we define h_b^{ξ} as the unique path between the origin $\xi(b)$ and the belief b which does not contain a state request action ι. The fact that $b \in \mathcal{F}(\mathcal{G})$ ensures the existence of the path, while the uniqueness is guaranteed by construction as only the corner beliefs can have multiple parents.

Computing Ψ

Our heuristic (Eq. (4.12)) requires to compute $\Psi(b_0, b)$ (Eq. (4.7)) for all fringe beliefs $b \in \mathcal{F}(\mathcal{G})$. While straightforward in a tree structure, it becomes complex in a graph due

Algorithm 3: AEMS-SR: Anytime Error Minimization Search with State Requests

input: t: Maximum time, ε: Error threshold
while *not EnvironmentTerminated()*
  Initialize $\mathcal{G}$ with initial belief state b_0^0 as root
  $t_0 \leftarrow$ Time()
  while *Time() -* $t_0 \leq t$ *and not Solved*(b_0^0, ε)
    // Calls Algorithm 4
    $b^* \leftarrow$ GetBeliefToExpand($\mathcal{G}$)
    Expand(b^*)
    UpdateAncestors(b^*)
  $\hat{a}_0 \leftarrow \arg\max_{a \in \{\bar{\iota}, \iota\}} L_{\mathcal{G}}(b_0^0, a)$
  if $\hat{a}_0 = \iota$ **then**
    $s \leftarrow$ GetState()
    $\hat{a}_1 \leftarrow \arg\max_{a \in \mathcal{A}} L_{\mathcal{G}}(s, a)$
  else
    $\hat{a}_1 \leftarrow \arg\max_{a \in \mathcal{A}} L_{\mathcal{G}}(b_0^1, a)$
  DoAction($\hat{a}_1$); $o \leftarrow$ GetObservation()
  $b_0^0 \leftarrow \tau(s, \hat{a}_1, o)$ if $\hat{a}_0 = \iota$ else $\tau\left(b_0^0, \bar{\iota}, \hat{a}_1, o\right)$
  // Potentially improve the bounds (see Section 4.6.4)

to potentially infinite paths between b_0 and b. We address this challenge by leveraging the fact that only corner beliefs can have multiple parents to obtain Eq. (4.14). As the second part of the equation is straightforward to compute, our focus shifts to calculating $\Psi(b_0, s)$ for all corner beliefs s.

$$\Psi(b_0, b) = \Psi(b_0, \xi(b)) P\left(h_b^{\xi} \mid \xi(b), \hat{\pi}_{\mathcal{G}}\right) \tag{4.14}$$

We start by reducing the graph $\mathcal{G}$, such as the one in Fig. 4.1b, to $\bar{\mathcal{G}}$ containing only the root belief and corner beliefs. The set of nodes of $\bar{\mathcal{G}}$ is defined as $N' = \{b_0\} \cup \mathcal{S}$. An edge connects a node $b \in N'$ to a corner belief $s \in \mathcal{S}$ if there is at least one *direct path* h between the two nodes in the original graph $\mathcal{G}$, with a direct path defined as a path where only the last action is a request the state ι. This edge is weighted by the sum of the discounted cumulative probabilities over direct paths:

$$\text{edge}(b, s) = \sum_{\substack{h \in \Phi(b,s) \\ h \text{ is direct}}} \gamma^{d(h)} \mathbf{P}(h \mid b, \pi_{\mathcal{G}})$$

For nodes $n, n' \in N'$, we define $\bar{\Psi}(n, n')$ as 0 if there is no edge between n and n', and as the weight of the edge otherwise. As the paths starting in b_0 and ending in a corner belief s are either direct or pass through another corner belief s', we can write for all corner beliefs s:

$$\Psi(b_0, s) = \sum_{s' \in \mathcal{S}} \Psi\big(b_0, s'\big)\bar{\Psi}\big(s', s\big) + \bar{\Psi}(b_0, s) \tag{4.15}$$

We can rewrite Eq. (4.15) more compactly into Eq. (4.16) using matrix notations by defining the matrix $\bar{\Psi}_{s,s'} = \bar{\Psi}(s, s')$, and $\bar{\Psi}_{b_0}$ as the vector with components equal to $\bar{\Psi}(b_0, s)$. From which we obtain the solution given in Eq. (4.17).

$$\Psi_{b_0} = \bar{\Psi}\Psi_{b_0} + \bar{\Psi}_{b_0} \tag{4.16}$$

$$\Psi_{b_0} = (I - \bar{\Psi})^{-1}\bar{\Psi}_{b_0} \tag{4.17}$$

> *Remark 4.3*
>
> This solution could be seen as constructing the Markov chain corresponding to the corner beliefs and finding the stationary distribution of the policy $\hat{\pi}_{\mathcal{G}}$.

We can now compute $\Psi(b_0, b)$ for all fringes $b \in \mathcal{F}(\mathcal{G})$ (Eq. (4.14)), and expand $\tilde{b}(\mathcal{G})$ (Eq. 4.12).

Algorithm 4: getBeliefNodeToExpand

input: Graph $\mathcal{G}$, root belief b_0
if $b_0 \in \mathcal{F}(\mathcal{G})$ **then**
  return b_0
/* *Call to Algorithm 5 that traverse the graph to compute* $\bar{\Psi}, \bar{\Psi}_{b_0}, \mathcal{F}(\mathcal{G})$ */
$\mathcal{V}, \bar{\Psi}, \bar{\Psi}_{b_0}, \mathcal{F}(\mathcal{G}) \leftarrow \text{GWalk}(b_0, \{\}, 0_{|\mathcal{S}|\times|\mathcal{S}|}, 0_{|\mathcal{S}|}, \{\})$
$\Psi_{b_0} = (I_{|\mathcal{S}|} - \bar{\Psi})^{-1}\bar{\Psi}_{b_0}$ (Eq. 4.17)
bestE = $-\infty$
for $b \in \hat{\mathcal{F}}$
  E = $(U(b) - L(b)) \cdot \Psi(b_0, b)$ *(Eq. 4.14)*
  if $E > bestE$ **then**
    bestE = E
    $b_{\text{best}} = b$
return b_{best}

Algorithm 5: GWalk: computing $\bar{\Psi}$ and $\bar{\Psi}_{b_0}$

input: belief b, visitedStates $\mathcal{V}$, stateMatrix M, $\bar{p}$, fringeNodes $\hat{\mathcal{F}}$, Graph $\mathcal{G}$, root belief b_0

if $b \in \mathcal{F}(\mathcal{G})$ **then**
  $\hat{\mathcal{F}} = \hat{\mathcal{F}} \cup \{b\}$
else if $\pi(b) = \iota$ **then**
  for $s \in supp(b)$
    if $\xi(b) = b_0$ **then**
      $\bar{p}[s]$ += $b[s] * \bar{\Psi}(\xi(b), b)$
    else
      $M_{\xi(b),s}$+ = $b[s] * \bar{\Psi}(\xi(b), b)$
    if $s \notin \mathcal{V}$ **then**
      $\mathcal{V} = \mathcal{V} \cup \{s\}$
      for $o \in \mathcal{O}(s, \pi(s))$
        $\mathcal{V}, M, \bar{p}, \hat{\mathcal{F}} \leftarrow$ GWalk$(\tau(s, \pi(s), o), \mathcal{V}, M, \bar{p}, \hat{\mathcal{F}})$
else
  for $o \in \mathcal{O}(b, \pi(b))$
    $\mathcal{V}, M, \bar{p}, \hat{\mathcal{F}} \leftarrow$ GWalk$(\tau(b, \pi(b), o), \mathcal{V}, M, \bar{p}, \hat{\mathcal{F}})$
return $\mathcal{V}, M, \bar{p}, \hat{\mathcal{F}}$

Algorithm 4 includes the pseudo-code to obtain the next belief node to expand which calls Algorithm 5 to obtain $\bar{\Psi}$ and $\bar{\Psi}_{b_0}$. This computation enables the calculation of Ψ_{b_0} and returns $\tilde{b}(\mathcal{G})$ for expansion.

Update ancestors

Updating the ancestors consists of leveraging the knowledge gained through a belief expansion to update the lower and upper bound in the graph $L_{\mathcal{G}}$ and $U_{\mathcal{G}}$ by enforcing Equations 4.4 and 4.5. As explained in LAO* [Hansen and Zilberstein, 2001], this requires running dynamic programming. When working with stochastic trees, such as AEMS, the dynamic programming results in updating the parents sequentially until reaching the root, guaranteeing convergence in a number of steps equal to the depth of the tree. In contrast, when dealing with cyclic graphs, a full dynamic programming update is needed, which can be computationally expensive. To address this challenge, LAO* proposes a method to limit the number of nodes to update by focusing on a subset of nodes. In our

implementation, instead of traversing the graph to determine the set of nodes to consider for dynamic programming, we update the parents recursively until the updates become smaller than a threshold (10^{-6} in our experiments), and we maintain a queue to deal with the cyclic structure.

The cycle of identifying the next belief to expand, expanding it, and backtracking the lower and upper bounds continues until the predefined time limit is reached or the error gap at the root, $\hat{e}_{\mathcal{G}}(b_0)$, becomes less than ε. Subsequently, the agent determines whether to request the state and selects an environmental action, by maximizing the lower bound $L_{\mathcal{G}}$. Then the agent obtains a new observation, triggering the reinitialization of the graph with the updated belief.

4.6 Bounds

AEMS-SR requires a lower and upper bound L, U for Eq. 4.4, 4.5 and 4.11. Those bounds, computed offline, are represented as a set Γ of $|\mathcal{A}|$ α-vectors, one for each action a and denoted as α_a. Evaluation of the bounds on a belief b is given by $\max_{\alpha\in\Gamma} \langle b, \alpha\rangle$. Typically, the lower bound is derived from Blind policies [Hauskrecht, 1997] that consistently select the same action. For the upper bound, the two main algorithms are QMDP [Cassandra et al., 1997] and FIB [Hauskrecht, 2000]. As the agent retains the option not to request the state, the ability to request the state cannot reduce the expected return but it may potentially increase it. Therefore, ensuring the validity of the upper bounds in POMDP-SRs is crucial.

4.6.1 Q-MDP

Q-MDP is constructed by assuming that the uncertainty about the state will disappear after the next action and corresponds to solving the underlying MDP. The $\alpha-$vector α_a is the fixed point of Eq. 4.18, and $\alpha_a(s)$ corresponds to the Q-Value $Q(s, a)$ of the underlying MDP.

$$\alpha_a(s) = \mathcal{R}(s, a) + \gamma \sum_{s'\in S} \mathbf{P}\left(s' \mid s, a\right) \max_{\alpha_{a'}\in\Gamma} \alpha_{a'}\left(s'\right) \tag{4.18}$$

Lemma 4.3

The Q-MDP upper bound of a standard POMDP $\mathcal{P}$ is an upper bound for the equivalent POMDP $\mathcal{P}'$ of $\mathcal{P}_{\text{SR}}$.

Proof. In the underlying MDP $\mathcal{M}'$ of the Equivalent POMDP, the optimal policy will avoid state requests due to the penalty and the full observability. Consequently, the

optimal policies of both MDPs only differ by the inclusion of non-request actions, which carry zero reward and thus do not affect the expected return. Hence, for any $s \in \mathcal{S}$, the value of the optimal policy in $\mathcal{M}$ at s matches the one in $\mathcal{M}'$ at s^0. Therefore QMDP is an upper bound for the values of $\mathcal{P}_{\text{SR}}$. □

4.6.2 Fast Informed Bound

Fast Informed Bound (FIB) considers the partial observability at the next step which provides a tighter upper bound compared to Q-MDP. The associated α-vectors are constructed by iteratively applying the operator associated to Eq. 4.19.

$$\alpha_a(s) = \mathcal{R}(s,a) + \gamma \sum_{o \in \Omega} \max_{\alpha_{a'} \in \Gamma} \sum_{s' \in \mathcal{S}} P\big(s', o \mid s, a\big)\alpha_{a'}\big(s'\big) \tag{4.19}$$

Lemma 4.4

The Fast Informed Bound upper bound of a standard POMDP $\mathcal{P}$ is not guaranteed to be an upper bound for $\mathcal{P}_{\text{SR}}$'s equivalent POMDP $\mathcal{P}'$.

Proof. Consider a POMDP $\mathcal{P}$ with two states, a uniform probability transition function, a unique observation, and two actions such that

$$\mathcal{R}(s_1, a_1) = \mathcal{R}(s_2, a_2) = 1 \qquad \mathcal{R}(s_1, a_2) = \mathcal{R}(s_2, a_1) = -1$$

Q-MDP returns $(\frac{\gamma}{1-\gamma}+1, \frac{\gamma}{1-\gamma}-1)$ and $(\frac{\gamma}{1-\gamma}-1, \frac{\gamma}{1-\gamma}+1)$ as α-vectors, which correspond to the uncertainty of the initial state ± 1 and observing to the following states. Conversely, FIB returns $(1,-1)$ and $(-1,1)$ as α-vectors corresponding to the first reward followed by an expected future return of 0. Let us now consider the POMDP-SR $\mathcal{P}_{\text{SR}} = \langle \mathcal{P}, c \rangle$ and the policy that always request the state to then select the optimal action. This policy has an expected discounted return equal to $\frac{1-c}{1-\gamma}$. Setting the cost c to 0.1 proves that FIB is not an upper bound for POMDP-SR. □

4.6.3 FIB-SR

Adapting FIB to POMDP-SR involves introducing an additional α-vector, α_c, corresponding to the action of requesting the state. FIB-SR alternates between updating α-vectors for environmental actions using Eq. 4.19 with $\Gamma = \{\alpha_a, \forall a \in \mathcal{A}\} \cup \alpha_c$ and updating α_c using Eq. 4.20. This process reflects paying the cost c to observe the state and then selecting the environmental action.

$$\alpha_c(s) = -c + \max_{a \in \mathcal{A}} \alpha_a(s) \tag{4.20}$$

4.6.4 Improving the bounds during learning

In traditional POMDPs, maintaining offline-computed bounds unchanged during online phases is standard practice. While updating these bounds could enhance the algorithm's efficiency over successive time steps and episodes, this approach is generally not feasible. The primary obstacle is the need to store additional alpha vectors, which would diminish the efficiency of computing bounds for new beliefs and increase memory demands. Conversely, in POMDP-SRs, corner beliefs present an opportunity to update bounds efficiently. For every corner belief s in the graph $\mathcal{G}$ and action a, $U_{\mathcal{G}}(s, a)$ provides a tighter bound than $\alpha_a(s)$ computed offline. Therefore, by replacing $\alpha_a(s)$ with the value of $U_{\mathcal{G}}(s, a)$, we can update the upper bound without requiring additional memory. The same approach applies to lower bounds, resulting in a practical and efficient solution.

4.7 Experiments

Many existing POMDP benchmarks, such as RockSample [Smith and Simmons, 2004], feature partial observability that can be permanently resolved with a single state request, rendering the planning problem trivial in the POMDP-SR setting. We instead evaluate AEMS-SR on two environments where partial observability is is restored at the next timestep: *Tag* [Pineau et al., 2006], a standard benchmark, and *RobotDelivery*, a novel domain specifically designed for evaluating POMDP-SRs.

4.7.1 Environments

RobotDelivery

RobotDelivery is a grid-world environment (Fig. 4.3) featuring a main room (width 3, length $2n + 1$) with, at the top, n corridors, each two units long and leading to a package pickup point. At the beginning of each episode, the agent starts at (A) and a package is in one of the pickup-points, with equal probability. Its mission is to collect the package and deliver it to point (D), receiving a reward of 1 for each successful delivery. After each delivery, there is a probability e that no new packages will spawn. Package spawning occurs with probability $1-t$ into the waiting area (W) and with probability t in one of the pickup points. If a package is in the waiting area, it has a probability t of being transferred to a pickup point. The waiting area forces the agent to time its state request as requesting the state when the package is in (W) would force the agent to request it again. The agent has four possible actions (up, down, left, right). Except when moving into a package location or delivery location, actions have a failure probability f causing the agent to remain stationary. There are 5 observations, the first four indicate the number of walls surrounding the agent (0-3), and the fifth occurs when going up to a pickup point with

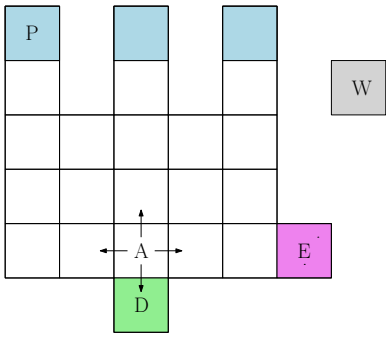


Figure 4.3: RobotDelivery ($n = 3$). The agent (A) needs to pick up the hidden package (P) from one of the three possible locations (blue) and deliver it to the delivery cell (D, green). W (grey) is the waiting area for packages, and E (violet) is the exit. After delivery, a new package appears in W with probability $1 - e$, moving uniformly to pickup points with probability t. The agent can exit through E at any time.

a package or going down into the delivery zone. The fact that the last observation can occur in two different situations does not create any ambiguity and is meant to simplify the observation space. The episode ends when the agent exits through (E), which can be done at any time, yielding a reward of 1.

Tag

Tag is a grid-world environment ($|S| = 842, |\Omega| = 30$) where the agent chases a moving prey [Pineau et al., 2006]. The agent observes its own position if not in the same tile as the prey and a special observation otherwise. There are 5 actions, 4 corresponding to cardinal directions, each resulting in a reward of -1, and one action to tag the prey yielding -10 if not in the same tile and $+10$ otherwise. Successfully tagging the prey terminates the episode. The prey observes both positions, staying in place with a 0.2 probability, and only moves to increase its distance from the agent.

4.7.2 Setup

We evaluate AEMS-SR on RobotDelivery environments with 3, 5, and 7 corridors, corresponding to state spaces of 133, 305, and 541, respectively, and on Tag. Parameters are:

- **RobotDelivery:** $\gamma = 0.99$, $f = 0.1$, $t = 0.8$, $e = 1/3$, leading to an expected number of 3 packages per episode.

- **Tag:** $\gamma = 0.95$.

Each algorithm is run with time budgets per decision of 0.1s, 0.5s, and 1s. We compare AEMS-SR to POMCP and AEMS, both operating on the equivalent POMDP. For fairness:

- Our POMCP implementation extends the AI-Toolbox [Bargiacchi et al., 2020] to support variable action spaces, avoiding illegal-action penalties.
- AEMS is re-implemented to avoid doubling the state space allowing for faster belief updates, making the belief update time identical to AEMS-SR.

For Tag, we use a simplified version of the environment with 842 states (vs. 870 in the original) by collapsing multiple terminal states into a single one. We evaluate each algorithm over 4000 episodes (10 initial positions × 400 rollouts). Initial states are consistent across all algorithms.

We measure the following metrics:

- Discounted Return (R): average return per episode.
- Number of Expansions (NE): number of node expansions per step.
- Error Reduction (ER): $1 - \frac{U_{\mathcal{G}}(b_0) - L_{\mathcal{G}}(b_0)}{U(b_0) - L(b_0)}$.

> *Remark 4.4*
>
> The return and error reduction are not necessarily correlated: an agent may reduce uncertainty by proving that all reachable outcomes yield low return, resulting in low value but high confidence.

4.7.3 Results

Results without Improving the Bounds

Table 4.1 shows results with unrefined bounds. In RobotDelivery, POMCP consistently chooses to exit early, resulting in average returns below 1. In Tag, it generally fails to tag the prey, scoring worse than −17.

AEMS behaves similarly in RobotDelivery, failing to deliver packages. AEMS-SR, in contrast, successfully delivers at least once in R-3 and R-5, and achieves multiple deliveries at higher compute budgets. For instance, in R-3 with 0.5s, AEMS-SR reaches an average return of 2.49. In the harder R-7 setting, it reverts to the exit strategy, highlighting the challenge of deeper planning caused by the increased number of states.

In Tag, AEMS performs better than POMCP but is outperformed by AEMS-SR. Despite having a higher ER than AEMS-SR in Tag, AEMS episodes are longer and spend more steps in regions where ER is near zero.

Table 4.1: Experiment results comparing POMCP, AEMS and AEMS-SR with a time limit of 0.1s, 0.5s and 1s on Robot Delivery (with 3, 5 and 7 corridors and cost $c = 0.1$) and Tag (cost $c = 1$). AEMS and AEMS-SR use the FIB-SR upper bound and **do not improve the offline bounds during planning**. We report the mean and standard error for the return and the Error Reduction (ER), and the mean for the Number of Expansions (NE).

		Return			Number of Expansions		Error Reduction (%)	
	T	POMCP	AEMS	AEMS-SR	AEMS	AEMS-SR	AEMS	AEMS-SR
RobotDelivery 3	0.1	0.97±0.00	0.98±0.00	**1.38±0.01**	55	**2525**	3.8±0.0	**24.1±0.4**
	0.5	0.97±0.00	0.98±0.00	**2.49±0.03**	111	**11150**	4.4±0.0	**42.4±0.1**
	1.0	0.97±0.00	0.98±0.00	**2.50±0.03**	152	**20035**	4.6±0.0	**43.7±0.1**
RobotDelivery 5	0.1	0.95±0.00	0.96±0.00	**1.00±0.01**	29	**1292**	3.2±0.0	**7.6±0.2**
	0.5	0.95±0.00	0.96±0.00	**1.01±0.01**	59	**6668**	3.7±0.0	**8.6±0.2**
	1.0	0.95±0.00	0.96±0.00	**1.45±0.01**	80	**13347**	4.0±0.0	**32.3±0.3**
RobotDelivery 7	0.1	**0.93±0.00**	**0.93±0.00**	**0.93±0.00**	17	**696**	2.5±0.0	**6.2±0.0**
	0.5	0.93±0.00	**0.94±0.00**	**0.94±0.00**	37	**4024**	3.4±0.0	**6.9±0.0**
	1.0	0.93±0.00	**0.94±0.00**	**0.94±0.00**	51	**8071**	3.6±0.0	**7.1±0.0**
Tag	0.1	-17.43±0.06	-6.30±0.06	**-4.66±0.06**	20	**67**	54.7±0.1	**58.8±0.1**
	0.5	-17.45±0.06	-5.35±0.08	**-4.56±0.09**	56	**566**	**67.8±0.1**	64.2±0.1
	1.0	-17.47±0.06	-5.35±0.09	**-4.50±0.09**	79	**1124**	**69.6±0.1**	64.9±0.1

AEMS-SR achieves up to two orders of magnitude more expansions than AEMS. The reason is structural: AEMS builds a full subtree per belief support element (Figure 4.1a), making belief updates costly. AEMS-SR reuses corner beliefs thanks to the graph structure, drastically reducing redundant computation. This confirms the empirical benefit of using a graph instead of a tree for planning.

Results with Improving the Bounds

Table 4.2 reports results with online bound improvement (as described in Section 4.6). For AEMS, this has little effect due to limited expansion depth and high update cost.

In contrast, AEMS-SR shows strong gains. In R-7, the agent no longer exits immediately: with 0.5s, the average return exceeds 2. Even with only 0.1s, it achieves 1.84.

Some AEMS-SR runs terminate early, achieving their ε goal before exhausting the time budget, resulting in lower NE. The improved bounds also raise ER, since it is computed from the original offline bounds. This indicates better planning efficiency.

Overall, AEMS-SR outperforms both AEMS and POMCP across environments. Its graph-based representation and ability to exploit online bound refinement make it particularly effective in POMDP-SR settings.

Table 4.2: Experiment results comparing AEMS and AEMS-SR with a time limit of 0.1s, 0.5s and 1s on Robot Delivery (with 3, 5 and 7 corridors and cost $c = 0.1$) and Tag (cost $c = 1$). AEMS and AEMS-SR use the FIB-SR upper bound and **improve the offline bounds during planning**. We report the mean and standard error for the return and the Error Reduction (ER), and the mean for the Number of Expansions (NE).

		Return		Number of Expansions		Error Reduction (%)	
	T	AEMS	AEMS-SR	AEMS	AEMS-SR	AEMS	AEMS-SR
RobotDelivery 3	0.1	0.98±0.00	**2.33±0.02**	54	**2170**	3.8±0.0	**72.6±0.5**
	0.5	0.98±0.00	**2.11±0.01**	111	**8106**	4.4±0.0	**77.2±0.5**
	1.0	0.98±0.00	**2.03±0.01**	151	**13279**	5.7±0.2	**79.2±0.5**
RobotDelivery 5	0.1	0.96±0.00	**2.17±0.02**	29	**1252**	3.2±0.0	**52.2±0.4**
	0.5	0.96±0.00	**2.25±0.02**	59	**5853**	3.7±0.0	**64.8±0.6**
	1.0	0.96±0.00	**2.21±0.02**	80	**10581**	4.0±0.0	**67.4±0.6**
RobotDelivery 7	0.1	0.94±0.00	**1.84±0.02**	17	**713**	2.5±0.0	**41.1±0.5**
	0.5	0.94±0.00	**2.14±0.02**	37	**3729**	3.4±0.0	**53.2±0.5**
	1.0	0.94±0.00	**2.16±0.02**	52	**7227**	3.6±0.0	**56.5±0.5**
Tag	0.1	-6.11±0.06	**-4.83±0.07**	20	**68**	60.8±0.1	**70.6±0.1**
	0.5	-5.41±0.09	**-4.26±0.09**	55	**571**	73.7±0.1	**80.8±0.1**
	1.0	-5.23±0.09	**-4.53±0.09**	78	**1144**	74.6±0.1	**82.2±0.1**

4.7.4 Additional Experiments

Effect of Bound Choice

Table 4.3 presents results using the Q-MDP upper bound instead of FIB-SR. Generally, both upper bounds yield similar average results. A notable exception is observed in R-3-1, where AEMS with improved bounds achieves an average return of 1.04, suggesting a shift away from the exit-directly strategy in certain instances. This further underscores the benefit of improving bounds during the online phase.

Increased Stochasticity

Table 4.4 presents the results for a modified RobotDelivery version with a movement failure probability $f = 0.2$, also using the Q-MDP upper bound for ease of comparison. AEMS's performance remains unaffected by this change, with variations in return due to the longer expected exit time. AEMS-SR shows lower returns compared to the $f = 0.1$ scenario. This outcome was expected given that the increasing f results in expending the support of the beliefs and on increasing the necessary time to pickup and deliver

Table 4.3: Results for RobotDelivery for POMCP, AEMS and AEMS-SR with 3, 5 and 7 corridors, a time limit of 0.1s, 0.5 and 1s, a cost $c = 0.1$, probability of failure of movement $f = 0.1$, probability transfer from the waiting area $t = 0.8$, expected number of packages $e = 3$. POMCP results are averaged on 400 runs. AEMS and AEMS-SR use the Q-MDP upper bound and their results are averaged over 800 runs. We report the mean and standard error to the mean.

			Not Improve Bounds						Improve Bounds					
		Return	Return		NU		ER (%)		Return		NU		ER (%)	
		POMCP	AEMS	AEMS-SR	AEMS	AEMS-SR	AEMS	AEMS-SR	AEMS	AEMS-SR	AEMS	AEMS-SR	AEMS	AEMS-SR
	T	±0.00	±0.00		±0		±0.0				±0			
R-3	0.1	0.97	0.98	**1.37±0.01**	56	**2473±3**	3.9	**22.6±0.4**	0.98±0.00	**2.33±0.02**	56	**2209±7**	4.0±0.0	**71.6±0.5**
	0.5	0.97	0.98	**2.48±0.03**	113	**10624±26**	4.5	**42.1±0.1**	0.98±0.00	**2.12±0.02**	113	**7994±86**	4.5±0.0	**78.1±0.5**
	1.0	0.97	0.98	**2.47±0.03**	156	**19142±49**	4.8	**41.7±0.1**	1.04±0.01	**2.00±0.01**	152	**13671±188**	15.8±0.7	**79.7±0.5**
R-5	0.1	0.95	0.96	**1.00±0.01**	28	**1287±2**	3.2	**7.7±0.2**	0.96±0.00	**2.21±0.02**	29	**1251±4**	3.3±0.0	**53.0±0.4**
	0.5	0.95	0.96	**1.01±0.01**	58	**6637±8**	3.8	**8.7±0.2**	0.96±0.00	**2.26±0.02**	59	**5918±28**	3.8±0.0	**65.9±0.6**
	1.0	0.95	0.96	**1.54±0.01**	79	**13348±14**	4.0	**35.7±0.2**	0.96±0.00	**2.21±0.02**	80	**10652±72**	4.1±0.0	**67.4±0.6**
R-7	0.1	0.93	**0.94**	**0.94±0.00**	17	**704±1**	2.5	**6.3±0.0**	0.94±0.00	**1.98±0.02**	17	**713±2**	2.6±0.0	**46.3±0.4**
	0.5	0.93	**0.94**	**0.94±0.00**	37	**4029±4**	3.4	**7.0±0.0**	0.93±0.00	**2.17±0.02**	37	**3785±13**	3.4±0.0	**53.9±0.5**
	1.0	0.93	**0.94**	**0.94±0.00**	51	**8042±8**	3.7	**7.2±0.0**	0.94±0.00	**2.16±0.02**	51	**7313±32**	3.7±0.0	**57.2±0.5**

Table 4.4: Results for RobotDelivery for POMCP, AEMS and AEMS-SR with 3, 5 and 7 corridors, a time limit of 0.1s, 0.5 and 1s, a cost $c = 0.1$, probability of failure of movement $\boldsymbol{f = 0.2}$, probability transfer from the waiting area $t = 0.8$, expected number of packages $e = 3$. POMCP results are averaged on 400 runs. AEMS and AEMS-SR use the Q-MDP upper bound and their results are averaged over 800 runs. We report the mean and standard error to the mean.

			Not Improve Bounds						Improve Bounds					
		Return	Return		NU		ER (%)		Return		NU		ER (%)	
		POMCP	AEMS	AEMS-SR	AEMS	AEMS-SR	AEMS	AEMS-SR	AEMS	AEMS-SR	AEMS	AEMS-SR	AEMS	AEMS-SR
	T	±0.00	±0.00		±0		±0.0		±0.00		±0		±0.0	
R-3	0.1	0.97	0.98	**1.15±0.01**	55	**2289±5**	3.6	**13.6±0.4**	0.98	**2.43±0.03**	55	**2240±7**	3.6	**59.6±0.5**
	0.5	0.97	0.98	**2.45±0.03**	111	**10410±29**	4.1	**39.5±0.1**	0.97	**2.33±0.02**	112	**8627±66**	4.1	**70.4±0.5**
	1.0	0.97	0.98	**2.48±0.03**	151	**18972±63**	4.3	**41.8±0.1**	0.97	**2.23±0.02**	152	**14465±149**	4.3	**73.0±0.5**
R-5	0.1	0.94	0.95	**0.95±0.00**	28	**1302±2**	3.0	**6.3±0.0**	0.95	**0.99±0.01**	28	**1337±2**	3.1	**38.5±0.7**
	0.5	0.95	0.95	**0.97±0.00**	57	**6121±8**	3.6	**7.7±0.1**	0.95	**2.10±0.03**	58	**6105±22**	3.7	**50.4±0.6**
	1.0	0.95	0.95	**1.04±0.01**	78	**11433±27**	3.8	**11.0±0.3**	0.95	**2.19±0.02**	79	**11608±50**	3.9	**58.9±0.5**
R-7	0.1	0.92	**0.93**	**0.93±0.00**	16	**725±1**	2.4	**6.0±0.0**	**0.93**	**0.93±0.00**	16	**732±1**	2.5	**6.3±0.0**
	0.5	0.92	**0.93**	**0.93±0.00**	35	**4025±4**	3.2	**7.3±0.0**	0.93	**1.55±0.02**	36	**4023±11**	3.3	**39.3±0.7**
	1.0	0.92	**0.93**	**0.93±0.00**	49	**7711±7**	3.4	**7.7±0.0**	0.93	**1.89±0.02**	50	**7553±27**	3.5	**43.4±0.6**

Table 4.5: Results for RobotDelivery for POMCP, AEMS and AEMS-SR with 3, 5 and 7 corridors, a time limit of 0.1s, 0.5 and 1s, a cost $\boldsymbol{c} = \mathbf{0.25}$, probability of failure of movement $f = 0.2$, probability transfer from the waiting area $t = 0.8$, expected number of packages $e = 3$. POMCP results are averaged on 400 runs. AEMS and AEMS-SR use the Q-MDP upper bound and their results are averaged over 800 runs. We report the mean and standard error to the mean.

			Not Improve Bounds						Improve Bounds					
		Return	Return		NU		ER (%)		Return		NU		ER (%)	
		POMCP	AEMS	AEMS-SR	AEMS	AEMS-SR	AEMS	AEMS-SR	AEMS	AEMS-SR	AEMS	AEMS-SR	AEMS	AEMS-SR
	T	±0.00	±0.00	±0.00	±0		±0.0		±0.00		±0		±0.0	
R-3	0.1	0.93	**0.98**	**0.98**	54	**2416±3**	3.9	**6.9±0.0**	0.98	**2.38±0.03**	55	**2308±5**	3.9	**75.7±0.2**
	0.5	0.94	**0.98**	**0.98**	110	**12043±14**	4.5	**7.9±0.0**	0.98	**2.31±0.03**	111	**11220±29**	4.5	**78.7±0.2**
	1.0	0.94	0.98	**1.65**	150	**24867±27**	4.8	**37.6±0.1**	0.98	**2.42±0.03**	151	**22689±52**	4.8	**80.8±0.2**
R-5	0.1	0.93	**0.96**	**0.96**	28	**1317±2**	3.2	**6.7±0.0**	**0.96**	**0.96±0.00**	28	**1331±2**	3.2	**6.7±0.0**
	0.5	0.93	**0.96**	**0.96**	57	**6973±8**	3.8	**7.8±0.0**	**0.96**	**0.96±0.00**	59	**7051±7**	3.8	**7.9±0.0**
	1.0	0.93	**0.96**	**0.96**	79	**14020±15**	4.0	**8.3±0.0**	**0.96**	**0.96±0.00**	80	**14185±13**	4.1	**8.3±0.0**
R-7	0.1	0.92	**0.93**	**0.93**	17	**721±1**	2.5	**6.3±0.0**	**0.94**	0.93±0.00	17	**726±1**	2.6	**6.4±0.0**
	0.5	0.92	**0.94**	0.93	37	**4198±4**	3.4	**7.7±0.0**	**0.94**	**0.94±0.00**	37	**4231±4**	3.4	**7.7±0.0**
	1.0	0.92	**0.94**	**0.94**	50	**8567±7**	3.7	**8.2±0.0**	**0.94**	**0.94±0.00**	51	**8638±8**	3.7	**8.2±0.0**

packages. Nonetheless, AEMS-SR still outperforms AEMS and POMCP, both consistently adopting the exit-immediately strategy.

Higher State Request Cost

Table 4.5 details results for the RobotDelivery environment with a state request cost of $c = 0.25$, maintaining other parameters as in the main paper. Given our use of a greedy policy to approximate π^*, increasing the cost requires a more significant reduction in the no-request action's upper bound for AEMS-SR to consider state requests. This makes the problem more difficult. In contrast to POMCP and AEMS, which consistently opt for immediate exits, AEMS-SR still manages to deliver packages in R-3-1 without bounds improvement and in all R-3 instances if updating the offline bounds.

Stability Issues

We note that AEMS-SR's performance does not uniformly improve with increased compute time per step. Upon examining the results, we identified instances where AEMS-SR, while carrying a package in the corridor, consistently chooses the right action over the down action, resulting in the agent staying in place. This behavior occurs because both actions yield nearly identical optimal values, differing only by a factor of γ, and that the dynamic programming approach used in backtracking converges to an ε-solution, potentially introducing minor errors that can persist over time due to bound updates.

4.8 Conclusion

To address environments in which an agent can obtain full state information before each action at a cost, we introduced the POMDP with State Requests framework. Within this framework, we presented AEMS-SR, a principled online planning algorithm that mitigates the exponential growth of tree based search in POMDP SR by exploiting a cyclic graph structure. We proved that AEMS-SR is complete and ε optimal, and we empirically evaluated its performance on RobotDelivery, a novel benchmark designed for POMDP-SR, as well as on Tag. The results demonstrate that AEMS-SR consistently outperforms established online planning methods such as AEMS and POMCP in POMDP-SR settings.

As directions for future work, we aim to develop policies $\hat{\pi}_{\mathcal{G}}$ that are explicitly adapted to the structure of POMDP SR. We also plan to investigate whether the graph based search principles underlying AEMS SR can be extended to other subclasses of POMDPs.

4.8.1 Limitations and scope

The scope of AEMS-SR is limited to the class of problems typically addressed by online planning methods, which remain small in scale compared to the environments targeted by deep reinforcement learning. While the proposed graph based formulation significantly improves efficiency within this regime, it does not by itself address the challenges posed by high dimensional state spaces, long horizons, or continuous observations.

In addition, AEMS SR assumes access to the true underlying state at a known cost, an assumption that may not hold in many practical environments at execution time. An important direction for future work is therefore to investigate how costly access to partial states or richer obseravtions that do not fully reveal the state, could be incorporated into the same graph based planning framework.

Finally, the current formulation models information access through an explicit per query cost. Other forms of access constraints, such as fixed budgets or limits on the number of queries over an episode, are common in practice and may induce different planning tradeoffs. Extending the graph based strategy of AEMS SR to accommodate alternative access models, while preserving its ability to exploit structural redundancy, is a promising avenue for future research.

5 Learning Belief Models via Wasserstein Belief Updater

This chapter is based on the paper *The Wasserstein Believer: Learning Belief Updates for Partially Observable Environments through Reliable Latent Space Models* [Avalos et al., 2024], co-authored with Raphaël Avalos, Florent Delgrange, Guillermo Pérez, Ann Nowé, and Diederik M. Roijers, published in the proceedings of the 2024 International Conference on Learning Representations (ICLR).

Core Intuition

In partially observable reinforcement learning, the agent must compress its interaction history into a representation that is sufficient for decision-making. While recurrent neural networks can learn to encode histories in both supervised and reinforcement learning settings, the latter poses a fundamental challenge: the agent does not know a priori which aspects of the history matter for long-term success. What must be remembered is determined by consequences that are only revealed through interaction, making unguided history compression difficult to learn effectively.

Belief states provide a principled solution to this problem. By maintaining a posterior distribution over the underlying environment state, they define a sufficient statistic for optimal control under partial observability. However, exact belief updates require full knowledge of the environment dynamics and scale poorly with the size or continuity of the state space, rendering them impractical in most reinforcement learning settings.

The core idea behind the Wasserstein Belief Updater is to approximate belief-based reasoning with theoretical guarantees. WBU leverages a training protocol where the agent can observe ground-truth states during training but must rely solely on partial observations at execution time. It learns both a latent Wasserstein Auto-Encoded MDP model with an observation function and a belief updater that maintains distributions over this latent space. By training these components in a coordinated round-robin fashion, WBU ensures they remain synchronized and recovers a state-like representation on which classical reinforcement learning algorithms can be applied.

5.1 Introduction

A central objective of this thesis is to investigate how access to the true environment state during training or execution can be used to improve decision-making under partial observability. When the agent cannot directly observe the environment state, its performance hinges on how effectively it can encode or infer relevant information from the interaction history. Previous chapters explored two distinct strategies for addressing this challenge.

In the LAN framework, we relied on recurrent neural networks to compress histories into fixed-length vectors that could serve as inputs to a value-based policy. This approach is scalable and widely used, but it offers no guarantees that the resulting representation is sufficient for optimal decision-making. In contrast, the AEMS-SR planner operated in belief space by explicitly maintaining a probability distribution over states. This enabled principled planning under uncertainty, but required full knowledge of the environment model and does not scale to more than a few hundred states.

Ideally, we would like to combine the strengths of both approaches: to bring the benefits of belief-based reasoning used in planning to reinforcement learning, which can scale to larger problems and learn from experience. However, in RL settings, the model of the environment is generally unknown, and exact belief updates are intractable in large or continuous state spaces. Even when a model is available or learned, the computational cost of maintaining beliefs over it remains prohibitive.

In this chapter, we introduce the *Wasserstein Belief Updater* (WBU), a model-based reinforcement learning algorithm that leverages access to the true state during training to make belief-style reasoning tractable in large-scale environments. Rather than attempting to perform belief updates directly over the environment's state space, WBU learns a compact latent model of the environment dynamics and leverages it to train a separate belief update mechanism. While the latent model itself remains too complex for exact belief tracking, it serves as a reliable foundation for supervising the learning of an approximate belief updater that operates efficiently in latent space.

The latent model is trained using a Wasserstein auto-encoding objective (WAE-MDP, Delgrange et al. [2023], Section 2.5.3), which trains the latent model so that states with similar future rewards and transition dynamics are close together in latent space, as formalized by bisimulation metrics (Definition 2.12). This yields theoretical guarantees that the latent space preserves behavioral equivalence, ensuring that the model remains suitable for policy optimization. Building on this model, WBU trains a feedforward belief update network that predicts the next belief from the previous belief, action, and observation. This belief updater is trained via a tractable variational proxy to the Wasserstein distance (Section 2.1.3), and comes with guarantees that the resulting latent

beliefs are sufficient for learning the optimal value function. Figure 5.1 provides an overview of the WBU framework.

WBU thus enables scalable and principled learning under partial observability. It does not rely on recurrence, nor does it require access to the environment model during execution. Instead, it uses state access during training to learn the latent model. This allows us to recover the benefits of belief-based reasoning in RL, with theoretical guarantees and without sacrificing scalability. To our knowledge, WBU is the first reinforcement learning method for partially observable environments that provides formal guarantees on the sufficiency of its learned representations for near-optimal control.

The remainder of this chapter is structured as follows. We begin by reviewing related work in latent dynamics modeling and belief learning. We then present the WBU framework in detail, including the learning procedures for the latent model and the belief updater. Finally, we demonstrate empirically that WBU can learn high-quality policies without recurrent architectures or access to the environment model at test time.

5.2 Related Work

A wide range of approaches have been proposed to tackle partial observability in reinforcement learning, often by learning implicit or explicit representations of the interaction history. Early work on Predictive State Representations (PSRs) formalized the idea of representing state through predictions of future observations [Littman and Sutton, 2001]. These methods provided strong theoretical foundations for belief modeling but were computationally limited and have seen little application in large-scale reinforcement learning. One common direction, exemplified by methods like R-A2C and DVRL [Igl et al., 2018], relies on recurrent architectures to compress histories into fixed-size vectors. DVRL augments this approach with auxiliary objectives based on variational autoencoders and particle filtering, aiming to approximate beliefs over hidden states. Relatedly, DPFRL [Ma et al., 2020] embeds a differentiable particle filter into the policy to maintain an explicit belief over time via learned discriminative updates. However, the resulting representations come without guarantees, and by modelling beliefs as independent Gaussians it limits its expressivity and applicability in complex environments.

Other latent dynamics approaches, such as PlaNet [Hafner et al., 2019c] and Dreamer [Hafner et al., 2019b], learn powerful predictive models but do not explicitly maintain beliefs or provide guarantees that their latent representations are sufficient for decision-making. WBU differs by explicitly training a belief updater and grounding its latent space in bisimulation-based guarantees.

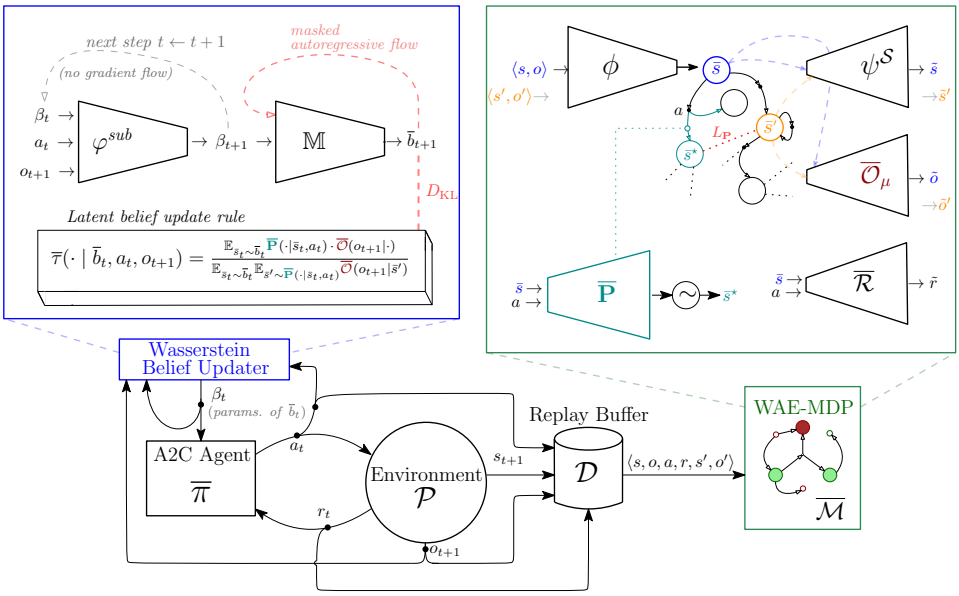


Figure 5.1: *WBU framework.* The WAE-MDP is presented in Sect. 5.3, and WBU in Sect. 5.4. Learning the different components is done in a round-robin fashion. The WAE-MDP learns from data collected by the RL agent and stored in a Replay Buffer. WBU uses the transition function $\overline{\mathbf{P}}_\theta$ and observation decoder $\overline{\mathcal{O}}_\theta$ of the WAE-MDP to learn to approximate the belief update rule. The agent learns a policy conditioned on the resulting *sub-belief* β_t (i.e., the parameters of the latent belief $\overline{b}_t$).

Other methods focus on richer distributional representations. FORBES [Chen et al., 2022] leverages normalizing flows to model the belief distribution more flexibly. Yet, instead of conditioning the policy on the full belief, it uses a single sampled latent state as input. This design choice discards the uncertainty information that is crucial for decision-making under partial observability. While using a sufficiently large set of samples could approximate the full belief, this work does not take that approach, and the single-sample strategy can lead to suboptimal behavior in complex environments. Related approaches target specific settings, such as image-based motor control tasks [Lee et al., 2020] or synthetic POMDPs where observations are generated by masking states with Gaussian noise [Wang and Tan, 2021]. These works typically hard-code the structure of partial observability and do not aim for general-purpose solutions.

An alternative perspective on learning generative models for control under partial observability is provided by active inference [Friston et al., 2006; Friston, 2010], which formulates action selection as minimizing expected free energy in a generative model. Recent work by Çatal et al. [2020] demonstrates that the required generative models can be learned using deep latent state space models, enabling active inference in high-dimensional observation spaces. While this approach addresses similar challenges to world model-based reinforcement learning, it differs from WBU in that control objectives are encoded through preferences over latent states rather than reward functions, and the learned representations are not explicitly constrained to preserve bisimulation-based control equivalence.

Closer to our setting are works that assume access to the environment state during training. In multi-agent RL, this assumption is common under the Centralized Training with Decentralized Execution (CTDE) paradigm (Section 2.4). In single-agent settings, it has been explored in kernel-POMDPs [Nishiyama et al., 2012], where state access is used to learn reproducing kernel Hilbert space (RKHS) embeddings of transitions, and more recently in Lee et al. [2023], who use it to derive sample complexity and regret bounds for tabular POMDPs under specific function approximation classes. However, these approaches either operate in simplified settings or focus exclusively on statistical guarantees, and do not learn structured latent representations suitable for downstream belief estimation or policy optimization.

Leveraging privileged training information more generally has been investigated in works like Lambrechts et al. [2024], where auxiliary signals are used to regularize RNNs. Yet, the resulting representations still lack abstraction guarantees, and belief tracking is never made explicit. In contrast, our approach learns an explicit belief updater over a structured latent model, with formal guarantees that the learned belief distributions suffice for value learning.

From a theoretical standpoint, several recent works have studied value difference bounds and their connection to bisimulation metrics [Gelada et al., 2019; Delgrange et al., 2022]. These results provide important foundations for our own guarantees, but have so

far been restricted to fully observable MDPs. In WBU, we extend this reasoning to latent belief models trained under partial observability. We show that local value difference bounds can ensure that these representations support sound belief updates.

Lastly, while RQL-AIS [SeyedSalehi et al., 2023] provides an analysis of recurrent Q-learning with value difference bounds over hidden RNN states, their results are primarily theoretical and rely on global metrics that are impractical to optimize in large-scale problems. In contrast, our guarantees rely on local, on-policy losses that are tractable to compute and minimize during training. This makes WBU both practically scalable and theoretically grounded, setting it apart from previous approaches to representation learning in POMDPs.

A distinctive feature of WBU is its reliance on full state information during training. This assumption aligns with the broader theme of this thesis, which investigates how privileged information can be used to supervise representation learning. In WBU, state access provides the supervision necessary to construct a latent space with formal guarantees, offering a principled baseline for future approaches that operate with weaker assumptions.

5.3 Learning the Dynamics

To enable belief-based reasoning in large-scale environments, our approach relies on learning a latent-space model of the POMDP dynamics. Unlike traditional reinforcement learning agents, which must learn purely from interaction without access to environment internals, we assume access to the true environment state during training. This privileged information allows us to learn an explicit latent model of the environment, which in turn enables us to approximate belief updates in a tractable space.

> **Assumption 5.1: Training-Time State Access**
>
> During training, the agent has access to the true environment state $s \in \mathcal{S}$ at each time step. This access is used solely for model learning and is not available during execution.

Given a POMDP $\mathcal{P} = \langle \mathcal{M}, \Omega, \mathcal{O} \rangle$ with underlying MDP $\mathcal{M} = \langle \mathcal{S}, \mathcal{A}, \mathbf{P}, \mathcal{R}, s_I, \gamma \rangle$, we use this assumption to learn a structured latent model of the environment dynamics, including both the transition dynamics of $\mathcal{M}$ and the observation function $\mathcal{O}$.

5.3.1 The Latent POMDP Encoding

Our approach begins by transforming the original POMDP into an equivalent form that makes the observation process deterministic. This enables learning both the transition and observation components within a unified framework.

Definition 5.1: Observation-Augmented POMDP

We define the *observation-augmented POMDP* as $\mathcal{P}^{\uparrow} = \langle \mathcal{M}_{\Omega}, \Omega, \mathcal{O}^{\uparrow} \rangle$ where:

- $\mathcal{M}_{\Omega} = \langle \mathcal{S}_{\Omega}, \mathcal{A}, \mathbf{P}_{\Omega}, \mathcal{R}_{\Omega}, (s_I, o_I), \gamma \rangle$ is a refined MDP with:
  - $\mathcal{S}_{\Omega} = \mathcal{S} \times \Omega$;
  - The transition function is given by
$$\mathbf{P}_{\Omega}((s', o') \mid (s, o), a) = \mathbf{P}(s' \mid s, a) \cdot \mathcal{O}(o' \mid s', a)$$
  - The reward function satisfies $\mathcal{R}_{\Omega}((s, o), a) = \mathcal{R}(s, a)$;
- The observation function $\mathcal{O}^{\uparrow} : \mathcal{S}_{\Omega} \to \Omega$ is the deterministic projection $\mathcal{O}^{\uparrow}((s, o)) = o$.

The POMDPs $\mathcal{P}$ and $\mathcal{P}^{\uparrow}$ are equivalent [Chatterjee et al., 2016]: $\mathcal{P}^{\uparrow}$ captures the stochasticity of $\mathcal{O}$ in its transition function through the refinement of the state space, further *yielding a deterministic observation function*, only dependent on refined states.

Henceforth, we reduce the problem of learning a latent space model of $\mathcal{P}$ to learning a WAE-MDP from $\mathcal{M}_{\Omega}$. Precisely, we learn a latent MDP $\overline{\mathcal{M}} = \left\langle \overline{\mathcal{S}}, \mathcal{A}, \overline{\mathbf{P}}, \overline{\mathcal{R}}, \bar{s}_I, \gamma \right\rangle$ linked to $\mathcal{M}_{\Omega}$ via the embedding $\phi : \mathcal{S}_{\Omega} \to \overline{\mathcal{S}}$. The latent MDP $\overline{\mathcal{M}}$ encodes the observation dynamics through $\overline{\mathbf{P}}$ and enables to learn the deterministic observation function $\mathcal{O}^{\uparrow}$ through the decoder $\Psi : \overline{\mathcal{S}} \to \mathcal{S}_{\Omega}$, by decomposing the latter in two networks $\Psi^{\mathcal{S}} : \overline{\mathcal{S}} \to \mathcal{S}$ and $\overline{\mathcal{O}}_{\mu} : \overline{\mathcal{S}} \to \Omega$, further yielding $\Psi(\bar{s}) = \langle \Psi^{\mathcal{S}}(\bar{s}), \overline{\mathcal{O}}_{\mu}(\bar{s}) \rangle$ for all $\bar{s} \in \overline{\mathcal{S}}$. This way, the WAE-MDP learns all the components of $\mathcal{P}^{\uparrow}$, the latter being equivalent to $\mathcal{P}$. With this model, we construct a *latent POMDP* $\overline{\mathcal{P}}_{\theta} = \langle \overline{\mathcal{M}}_{\theta}, \Omega, \overline{\mathcal{O}}_{\theta} \rangle$, where $\overline{\mathcal{O}}_{\theta}$ is a Dirac measure with impulse $\overline{\mathcal{O}}_{\mu}$.

As with any POMDP, the belief update function $\overline{\mathcal{T}}$ of $\overline{\mathcal{P}}_{\theta}$ allows reasoning over beliefs to optimize return (see Property 2.3 for the formal definition of belief updates and Definition 2.9 for the belief-MDP formulation). Formally, assuming the latent belief at time step $t \geq 0$ is $\bar{b}_t \in \Delta(\overline{\mathcal{S}}) = \overline{\mathcal{B}}$, a_t is executed, and then o_{t+1} observed, $\bar{b}_t$ is updated according to $\overline{\mathcal{T}}(\bar{b}_t, a_t, o_{t+1}) = \bar{b}_{t+1}$ where $\bar{b}_{t+1}$ is defined as follows for all $\bar{s}_{t+1} \in \overline{\mathcal{S}}$.

$$\bar{b}_{t+1}(\bar{s}_{t+1}) = \frac{\mathbb{E}_{\bar{s}_t \sim \bar{b}_t} \overline{\mathbf{P}}_{\theta}(\bar{s}_{t+1} \mid \bar{s}_t, a_t) \cdot \overline{\mathcal{O}}_{\theta}(o_{t+1} \mid \bar{s}_{t+1})}{\mathbb{E}_{\bar{s}_t \sim \bar{b}_t} \mathbb{E}_{\bar{s}' \sim \overline{\mathbf{P}}_{\theta}(\cdot \mid \bar{s}_t, \bar{a}_t)} \overline{\mathcal{O}}_{\theta}(o_{t+1} \mid \bar{s}')}. \tag{5.1}$$

Latent policies. Given *any* history $h \in \mathcal{H}$, running a latent policy $\overline{\pi} : \overline{\mathcal{B}} \to \Delta(\mathcal{A})$ in $\mathcal{P}$ is possible by converting h into a latent belief $\overline{\mathcal{T}}^*(h) = \overline{b}$ and executing the action prescribed by $\overline{\pi}(\cdot \mid \overline{b})$. In theory, once the latent POMDP model is trained, one could use its exact belief update (Eq. 5.1) to maintain beliefs and compute an optimal policy conditioned on them, as in classical POMDP planning. However, evaluating this exact belief update requires integration over the entire latent state space, which is computationally intractable in large environments.

As a solution, we propose to leverage the access to the dynamics of $\overline{\mathcal{M}}_\theta$ to learn a *latent belief encoder* $\Phi : \overline{\mathcal{B}} \times \mathcal{A} \times \Omega \to \overline{\mathcal{B}}$ that approximates the belief update function by minimizing $D\big(\overline{\mathcal{T}}^*(h), \Phi^*(h)\big)$ for *some* discrepancy D and $h \in \mathcal{H}$ drawn from *some* distribution. The belief encoder Φ thus enables to learn a policy $\overline{\pi}$ conditioned on latent beliefs to optimize the return in $\mathcal{P}$: *given the current history h, the next action to play is given by* $a \sim \overline{\pi}(\cdot \mid \Phi^*(h))$.[1]

Two key questions arise:

i Does the WAE-MDP encoding induce a latent POMDP with behaviors, under the policy $\overline{\pi}$, close to $\mathcal{P}$?

ii Is the history representation induced by Φ suitable for optimizing the expected return in $\mathcal{P}$?

Our guarantees hinge on the history distribution and chosen discrepancy. The next section details our theoretical analysis of the required distributions and losses to answer (i) and (ii).

5.3.2 Losses and Theoretical Guarantees

Lemma 5.1

There exists a well defined stationary distribution $\mathcal{H}_{\overline{\pi}} \in \Delta(\mathcal{H})$ over histories likely to be seen at the limit of the interaction when $\overline{\pi}$ is executed in $\mathcal{P}$

Proof. As mentioned in Definition 2.7 all POMDPs can be represented as History MDPs, where the state space is the set of histories $\mathcal{H}$. Following Remark 2.1, by ensuring ergodicity (which is the case for episodic RL processes), MDPs have a well defined stationary distribution over states and thus over histories in the case of History MDPs and their associated POMDPs. □

Local losses

The objective function of the WAE-MDP incorporates *local losses* [Gelada et al., 2019] that minimize the expected distance between the original and latent reward and transition functions:

$$L_{\mathcal{R}} = \mathop{\mathbb{E}}_{s,o,a\sim\mathcal{H}_{\bar{\pi}}} \left|\mathcal{R}(s,a) - \overline{\mathcal{R}}_\theta(\phi_\iota(s,o),a)\right|, \quad L_{\mathbf{P}} = \mathop{\mathbb{E}}_{s,o,a\sim\mathcal{H}_{\bar{\pi}}} \mathcal{W}_{\bar{d}}\left(\phi_\iota \mathbf{P}_\Omega(\cdot \mid s,o,a), \overline{\mathbf{P}}_\theta(\cdot \mid \phi_\iota(s,o),a)\right);$$

and both are optimized *locally*, i.e., under $\mathcal{H}_{\bar{\pi}}$, where $s, o, a \sim \mathcal{H}_{\bar{\pi}}$ is a shorthand for (i) $h \sim \mathcal{H}_{\bar{\pi}}$ so that o is the last observation of h, (ii) $s \sim \mathcal{T}^*(h)$, and (iii) $a \sim \bar{\pi}(\cdot \mid \Phi^*(h))$.[1]

Furthermore, $\phi_\iota \mathbf{P}_\Omega(\cdot \mid s,o,a)$ is the distribution of transitioning to $\langle s', o'\rangle \sim \mathbf{P}_\Omega(\cdot \mid s,o,a)$, then embedding it to the latent space $\bar{s}' = \phi_\iota(s',o')$, $\bar{d}$ is a metric on $\overline{\mathcal{S}}$, and $\mathcal{W}_{\bar{d}}$ is the associated Wasserstein distance (Section 2.1.3). In practice, the ability of observing states during learning enables the optimization of those local losses without the need of explicitly storing histories. Instead, we simply store the transitions of $\mathcal{M}_\Omega$ encountered while executing $\bar{\pi}$. We also introduce an *observation loss*, using the total variation distance d_{TV} (Dudley [2018]; Section 2.1.2), which allows learning $\overline{\mathcal{O}}_\theta$:

$$L_{\mathcal{O}} = \mathop{\mathbb{E}}_{s,o,a\sim\mathcal{H}_{\bar{\pi}}} \mathop{\mathbb{E}}_{s'\sim\mathbf{P}(\cdot\mid s,a)} d_{TV}\left(\mathcal{O}\big(\cdot \mid s',a\big), \mathop{\mathbb{E}}_{o'\sim\mathcal{O}(\cdot\mid s',a)} \overline{\mathcal{O}}_\theta\big(\cdot \mid \phi_\iota\big(s',o'\big)\big)\right). \tag{5.2}$$

Belief losses

We instantiate the discrepancy measure D introduced above as the Wasserstein distance between the true latent belief update and our belief encoder.

In addition, we introduce the following *reward and transition regularizers* to reconcile the behaviors obtained in the fully observable latent model $\overline{\mathcal{M}}$ and the partially observable one $\overline{\mathcal{P}}$:

$$L_{\overline{\mathcal{T}}} = \mathop{\mathbb{E}}_{h\sim\mathcal{H}_{\bar{\pi}}} \mathcal{W}_{\bar{d}}\left(\overline{\mathcal{T}}^*(h), \Phi^*(h)\right), \quad L^{\Phi}_{\overline{\mathcal{R}}} = \mathop{\mathbb{E}}_{h,s,o,a\sim\mathcal{H}_{\bar{\pi}}} \mathop{\mathbb{E}}_{\bar{s}\sim\Phi^*(h)} \left|\overline{\mathcal{R}}_\theta(\phi_\iota(s,o),a) - \overline{\mathcal{R}}_\theta(\bar{s},a)\right|,$$
$$L^{\Phi}_{\overline{\mathbf{P}}} = \mathop{\mathbb{E}}_{h,s,o,a\sim\mathcal{H}_{\bar{\pi}}} \mathop{\mathbb{E}}_{\bar{s}\sim\Phi^*(h)} \mathcal{W}_{\bar{d}}\left(\overline{\mathbf{P}}_\theta(\cdot \mid \phi_\iota(s,o),a), \overline{\mathbf{P}}_\theta(\cdot \mid \bar{s},a)\right). \tag{5.3}$$

$L^{\Phi}_{\overline{\mathcal{R}}}$ and $L^{\Phi}_{\overline{\mathbf{P}}}$ aim at regularizing Φ and minimize the gap between the rewards (resp. transition probabilities) that are expected when drawing states from the current belief compared to those actually observed. Again, the ability to observe the states during training enables optimizing those losses without explicitly requiring the states to execute the policy.

[1] Analogous to $\mathcal{T}^*$, we define Φ^* as the recursive application of Φ along histories.

The belief loss and the related two regularizers can be optimized *on-policy*, i.e., coupled with the optimization of the latent policy $\bar{\pi}$ that is used to generate the episodes.

Value difference bounds

We provide guarantees concerning the *agent behaviors in* $\mathcal{P}$, when the policies are *conditioned on latent beliefs*. To do so, we formalize the behaviors of the agent through *value functions*. For a specific policy π, the value of a history is the expected return that would result from continuing to follow the policy from the latest point reached in that history: $V^{\pi}(h) = \mathbb{E}_{\pi}\left[\sum_{t=0}^{\infty} \gamma^t r_t \mid b_I = \overline{\mathcal{T}}^*(h)\right]$. Similarly, we write $\overline{V}^{\bar{\pi}}$ for the values of the latent policy $\bar{\pi}$ in $\overline{\mathcal{P}}_{\theta}$.

Suppose the agent uses a latent policy whose inputs are produced by Φ, we claim that when the losses are minimized to zero, then

(i) the latent model almost surely mimics the original environment, and

(ii) our belief representation captures the value function.

Precisely, let $\mathcal{L} = L_{\mathcal{R}} + L^{\Phi}_{\overline{\mathcal{R}}} + \mathrm{R}_{\mathrm{MAX}} L_{\overline{\mathcal{T}}} + \gamma K_{\overline{V}} \cdot (L_{\mathbf{P}} + L^{\Phi}_{\overline{\mathbf{P}}} + L_{\overline{\mathcal{T}}} + L_{\mathcal{O}})$, with $\mathrm{R}_{\mathrm{MAX}} = \left\|\overline{\mathcal{R}}\right\|_{\infty}$ and $K_{\overline{V}} = {}^{\mathrm{R}_{\mathrm{MAX}}}/_{1-\gamma}$, then:

Before stating the proof we need to introduce some properties.

Property 5.1: Lipschitz constant and total variation constants [Müller, 1997; Sriperumbudur et al., 2009]

Let $f : \mathcal{X} \to \mathbb{R}$, so that d is a metric on $\mathcal{X}$. Assume that f is K-Lipschitz, i.e., $f \in \mathcal{L}ipsch^K_d$ Eq. (2.2), then for any two distributions $P, Q \in \Delta(\mathcal{X})$,

$$\left| \mathop{\mathbb{E}}_{x_1 \sim P} f(x_1) - \mathop{\mathbb{E}}_{x_2 \sim Q} f(x_2) \right| \leq K \cdot \mathcal{W}_d(P, Q) \tag{5.4}$$

In the case where $d = \mathbb{1}_{\neq}$, for *any* bounded function $g : \mathcal{X} \to Y$ with $Y \subseteq \mathbb{R}$, we have the following with $K_Y \geq \sup_{x \in \mathcal{X}} |g(x)|$:

$$\left| \mathop{\mathbb{E}}_{x_1 \sim P} g(x_1) - \mathop{\mathbb{E}}_{x_2 \sim Q} g(x_2) \right| \leq K_Y \cdot \mathcal{W}_{\mathbb{1}_{\neq}}(P, Q) = K_Y \cdot d_{TV}(P,\ Q) \tag{5.5}$$

In both cases, we call the smallest K (resp. K_Y) the *integral probability metric (IPM) constant* of the function f (resp. g).

Lemma 5.2: Wasserstein in expectation

For any function $f : \mathcal{Y} \times \mathcal{X} \to \mathbb{R}$ so that $\mathcal{X}$ is equipped with the metric d, consider the function $g_y : \mathcal{X} \to \mathbb{R}$ defined as $g_y(x) = f(y, x)$. Assume that for any $y \in \mathcal{Y}$, g_y is Lipschitz (or simply bounded if $d = \mathbb{1}_{\neq}$) and let K be the IPM constant of g_y. Then, let $\mathcal{D} \in \Delta(\mathcal{Y})$ be a distribution over $\mathcal{Y}$ and $P, Q \in \Delta(\mathcal{X})$ be two distributions over $\mathcal{X}$, we have

$$\mathop{\mathbb{E}}_{y\sim\mathcal{D}} \left| \mathop{\mathbb{E}}_{x_1\sim P} f(y, x_1) - \mathop{\mathbb{E}}_{x_2\sim Q} f(y, x_2) \right| \leq K \cdot \mathcal{W}_d(P, Q)$$

Proof. The proof is straightforward by construction of g_y:

$$\begin{aligned}
&\mathop{\mathbb{E}}_{y\sim\mathcal{D}} \left| \mathop{\mathbb{E}}_{x_1\sim P} f(y, x_1) - \mathop{\mathbb{E}}_{x_2\sim Q} f(y, x_2) \right| \\
=& \mathop{\mathbb{E}}_{y\sim\mathcal{D}} \left| \mathop{\mathbb{E}}_{x_1\sim P} g_y(x_1) - \mathop{\mathbb{E}}_{x_2\sim Q} g_y(x_2) \right| \\
\leq& \mathop{\mathbb{E}}_{y\sim\mathcal{D}} \left[K \cdot \mathcal{W}_d(P, Q) \right] && \text{(by Property 5.1)} \\
=& K \cdot \mathcal{W}_d(P, Q)
\end{aligned}$$

□

Lemma 5.3

Let $\mathcal{P} = \langle \mathcal{M}, \Omega, \mathcal{O} \rangle$ be a POMDP with underlying MDP $\mathcal{M} = \langle \mathcal{S}, \mathcal{A}, \mathbf{P}, \mathcal{R}, s_I, \gamma \rangle$. Assume that $\mathcal{P}$ is finite, i.e., $\mathcal{S}$, $\mathcal{A}$, and Ω are finite sets. Then, $V^\star$ is almost Lipschitz continuous.

Proof. Define $\mathcal{V}$ as the set of real-valued bounded functions $V : \mathcal{B} \to \mathbb{R}$ and $U : \mathcal{B} \times \mathcal{A} \times \mathcal{V} \to \mathcal{V}$ as

$$U(b, a, V) = \mathcal{R}_{\mathcal{B}}(b, a) + \mathop{\mathbb{E}}_{b'\sim\mathbf{P}_{\mathcal{B}}(\cdot|b,a)} \left[\gamma V(b') \right].$$

The Bellman update operator is defined as $\mathcal{U} : \mathcal{V} \to \mathcal{V}$ as $(\mathcal{U}V)(b) = \max_{a\in\mathcal{A}} U(b, a, V)$ and is an isotone mapping that is a contraction under the supremum norm with fixed point $V^\star$, i.e., $V^\star = \mathcal{U}V^\star$ [Puterman, 1994; Hauskrecht, 2000; Sutton and Barto, 1998]. Furthermore, for any initial value function $V_0 \in \mathcal{V}$, the sequence resulting from value iteration (VI), $V_{i+1} = \mathcal{U}V_i$, converges to $V^\star$ (with linear convergence rate γ Puterman [1994]): for any $\epsilon' > 0$, there is a $i \in \mathbb{N}$ so that for all $j \geq i$, $\left\| V_j - V^\star \right\|_\infty \leq \epsilon'$. Now, let $\epsilon > 0$; in particular, the latter statement holds for $\epsilon' = \epsilon/2$. Since the convergence of VI holds for any initial value, we assume that $V_0 \in \mathcal{V}$

has been chosen as a *piecewise linear convex* (PWLC) function. Then, for all $i \geq 0$, V_i is also PWLC [Sondik, 1978; Smallwood and Sondik, 1973; Hauskrecht, 2000]. This means that V_i can be defined a finite collection of linear functions. Thus, V_i is also $2K$-Lipschitz: one just need take $2K$ as the higher slope of these functions (in absolute value). In consequence, for any pair of beliefs $b_1, b_2 \in \mathcal{B}$,

$$
\begin{aligned}
&|V^\star(b_1) - V^\star(b_2)| \\
=&|V^\star(b_1) - V_i(b_1) + V_i(b_1) - V_i(b_2) + V_i(b_2) - V^\star(b_2)| \\
\leq&|V^\star(b_1) - V_i(b_1)| + |V_i(b_1) - V_i(b_2)| + |V_i(b_2) - V^\star(b_2)| && \text{(Triangular inequality)} \\
\leq&2\epsilon' + |V_i(b_1) - V_i(b_2)| && \text{(by the convergence of VI)} \\
=&\epsilon + |V_i(b_1) - V_i(b_2)| \\
\leq&\epsilon + K \cdot d_{TV}(b_1,\, b_2), && \text{(since } d_{TV}(b_1,\, b_2) = {}^1\!/_2 \left\|b_1 - b_2\right\|_1\text{)}
\end{aligned}
$$

which means that $V^\star$ is almost Lipschitz, by definition. □

Lemma 5.3 can be applied to $\overline{\mathcal{P}}$. To see why, notice first the state space of $\overline{\mathcal{P}}$ is finite. Second, $\overline{\mathcal{O}}$ is deterministic by definition (it consists of the Dirac centered on the observation produced by $\overline{\mathcal{O}}_\mu$), so the observations of $\overline{\mathcal{P}}$ can be limited to the union of the supports of $\overline{\mathcal{O}}(\cdot \mid \bar{s})$, ranging over all latent states $\bar{s} \in \overline{\mathcal{S}}$. The resulting set is also finite. Note that the same reasoning holds for any latent observation function with finite support. Therefore, we obtain the following corollary.

Corollary 5.1

When the WAE-MDP is in the zero-temperature limit (Section 2.5.3), the optimal latent value function of $\overline{\mathcal{P}}$ is almost Lipschitz-continuous.

Theorem 5.1: Model quality

For any latent policy $\bar{\pi} : \overline{\mathcal{B}} \to \Delta(\mathcal{A})$, the values of $\mathcal{P}$ and $\overline{\mathcal{P}}$ are bounded by the local and belief losses in average when, in both models, the actions are produced by $\bar{\pi}$, which is conditioned on the latent belief induced by Φ, i.e., $a \sim \bar{\pi}(\cdot \mid \Phi^*(h))$:

$$
\mathop{\mathbb{E}}_{h \sim \mathcal{H}_{\bar{\pi}}} \left| V^{\bar{\pi}}(h) - \overline{V}^{\bar{\pi}}(h) \right| \leq \frac{\mathcal{L}}{1-\gamma}. \tag{5.6}
$$

Proof. The plan of the proof is as follows:

1. We exploit the fact that the value function can be defined as the fixed point of the Bellman's equations;

2. We repeatedly apply the triangular and the Jensen's inequalities to end up with inequalities which reveal mean discrepancies for either rewards or value functions;
3. We exploit the fact that the WAE-MDP is in the zero-temperature limit to bound those discrepancies by Wasserstein (see Porperty 5.1 and the related discussion);
4. The last two points allow highlighting the L_1- norm and Wasserstein terms in the local and belief losses;
5. Finally, we set up the inequalities to obtain a discounted next value difference term, and we exploit the stationary property of $\mathcal{H}_{\overline{\pi}}$ to fall back on the original, discounted, absolute value difference term;
6. Putting all together, we end up with an inequality only composed of constants, multiplied by losses that we aim at minimizing.

Concretely, the absolute value difference can be bounded by:

$$\begin{aligned}
&\mathop{\mathbb{E}}_{h \sim \mathcal{H}_{\overline{\pi}}} \left| V^{\overline{\pi}}(h) - \overline{V}^{\overline{\pi}}(h) \right| \\
&= \mathop{\mathbb{E}}_{h \sim \mathcal{H}_{\overline{\pi}}} \left| \mathop{\mathbb{E}}_{s \sim \mathcal{T}^*(h)} \mathop{\mathbb{E}}_{a \sim \overline{\pi}(\cdot \mid \Phi^*(h))} \left[\mathcal{R}(s, a) + \gamma \mathop{\mathbb{E}}_{s' \sim \mathbf{P}(\cdot \mid s, a)} \mathop{\mathbb{E}}_{o' \sim \mathcal{O}(\cdot \mid s', a)} V^{\pi}\left(h \cdot a \cdot o'\right) \right] \right. \\
&\qquad \left. - \mathop{\mathbb{E}}_{\bar{s} \sim \overline{\mathcal{T}}^*(h)} \mathop{\mathbb{E}}_{a \sim \overline{\pi}(\cdot \mid \Phi^*(h))} \left[\overline{\mathcal{R}}_{\theta}(\bar{s}, a) + \gamma \mathop{\mathbb{E}}_{\bar{s}' \sim \overline{\mathbf{P}}_{\theta}(\cdot \mid \bar{s}, a)} \mathop{\mathbb{E}}_{o' \sim \overline{\mathcal{O}}_{\theta}(\cdot \mid \bar{s}')} \overline{V}^{\overline{\pi}}\left(h \cdot a \cdot o'\right) \right] \right| \\
&\leq \mathop{\mathbb{E}}_{h \sim \mathcal{H}_{\overline{\pi}}} \mathop{\mathbb{E}}_{a \sim \overline{\pi}(\cdot \mid \Phi^*(h))} \left[\left| \mathop{\mathbb{E}}_{s \sim \mathcal{T}^*(h)} \mathcal{R}(s, a) - \mathop{\mathbb{E}}_{\bar{s} \sim \overline{\mathcal{T}}^*(h)} \overline{\mathcal{R}}_{\theta}(\bar{s}, a) \right| \right. \\
&\quad \left. + \gamma \left| \mathop{\mathbb{E}}_{s \sim \mathcal{T}^*(h)} \mathop{\mathbb{E}}_{s' \sim \mathbf{P}(\cdot \mid s, a)} \mathop{\mathbb{E}}_{o' \sim \mathcal{O}(\cdot \mid s', a)} V^{\pi}\left(h \cdot a \cdot o'\right) - \mathop{\mathbb{E}}_{\bar{s} \sim \overline{\mathcal{T}}^*(h)} \mathop{\mathbb{E}}_{\bar{s}' \sim \overline{\mathbf{P}}_{\theta}(\cdot \mid \bar{s}, a)} \mathop{\mathbb{E}}_{o' \sim \overline{\mathcal{O}}_{\theta}(\cdot \mid \bar{s}')} \overline{V}^{\overline{\pi}}\left(h \cdot a \cdot o'\right) \right| \right]
\end{aligned}$$

(Jensen's and Triangular inequality)

For the sake of clarity, we split the RHS into two parts.

Part 1: Reward bounds

$$\mathop{\mathbb{E}}_{h \sim \mathcal{H}_{\overline{\pi}}} \mathop{\mathbb{E}}_{a \sim \overline{\pi}(\cdot \mid \Phi^*(h))} \left| \mathop{\mathbb{E}}_{s \sim \mathcal{T}^*(h)} \mathcal{R}(s, a) - \mathop{\mathbb{E}}_{\bar{s} \sim \overline{\mathcal{T}}^*(h)} \overline{\mathcal{R}}_{\theta}(\bar{s}, a) \right|$$

To simplify the notation, we focus on the absolute part by setting $h \sim \mathcal{H}_{\bar{\pi}}$, o as the last observation of h, and $a \sim \bar{\pi}(\cdot \mid \Phi^*(h))$.

$$
\begin{aligned}
&\left| \mathop{\mathbb{E}}_{s\sim\mathcal{T}^*(h)} \mathcal{R}(s,a) - \mathop{\mathbb{E}}_{\bar{s}\sim\bar{\mathcal{T}}^*(h)} \bar{\mathcal{R}}_\theta(\bar{s},a) \right| \\
=&\left| \mathop{\mathbb{E}}_{s\sim\mathcal{T}^*(h)} \left[\mathcal{R}(s,a) - \bar{\mathcal{R}}_\theta(\phi_\iota(s,o),a)\right] + \mathop{\mathbb{E}}_{s\sim\mathcal{T}^*(h)} \mathop{\mathbb{E}}_{\bar{s}\sim\bar{\mathcal{T}}^*(h)} \left[\bar{\mathcal{R}}_\theta(\phi_\iota(s,o),a) - \bar{\mathcal{R}}_\theta(\bar{s},a)\right] \right| \\
&\qquad \text{(the state embedding function } \phi_\iota \text{ comes into play)} \\
\leq& \mathop{\mathbb{E}}_{s\sim\mathcal{T}^*(h)} \left|\mathcal{R}(s,a) - \bar{\mathcal{R}}_\theta(\phi_\iota(s,o),a)\right| + \left| \mathop{\mathbb{E}}_{s\sim\mathcal{T}^*(h)} \mathop{\mathbb{E}}_{\bar{s}\sim\bar{\mathcal{T}}^*(h)} \left[\bar{\mathcal{R}}_\theta(\phi_\iota(s,o),a) - \bar{\mathcal{R}}_\theta(\bar{s},a)\right] \right| \\
&\qquad \text{(Triangular and Jensen's inequality)} \\
=& \mathop{\mathbb{E}}_{s\sim\mathcal{T}^*(h)} \left|\mathcal{R}(s,a) - \bar{\mathcal{R}}_\theta(\phi_\iota(s,o),a)\right| \\
&\quad + \left| \mathop{\mathbb{E}}_{s\sim\mathcal{T}^*(h)} \mathop{\mathbb{E}}_{\bar{s}\sim\bar{\mathcal{T}}^*(h)} \mathop{\mathbb{E}}_{\bar{s}_\perp\sim\Phi^*(h)} \left[\left[\bar{\mathcal{R}}_\theta(\phi_\iota(s,o),a) - \bar{\mathcal{R}}_\theta(\bar{s}_\perp,a)\right] + \left[\bar{\mathcal{R}}_\theta(\bar{s}_\perp,a) - \bar{\mathcal{R}}_\theta(\bar{s},a)\right] \right] \right| \\
&\qquad \text{(the belief encoder } \Phi \text{ comes into play)} \\
=& \mathop{\mathbb{E}}_{s\sim\mathcal{T}^*(h)} \left|\mathcal{R}(s,a) - \bar{\mathcal{R}}_\theta(\phi_\iota(s,o),a)\right| + \mathop{\mathbb{E}}_{s\sim\mathcal{T}^*(h)} \mathop{\mathbb{E}}_{\bar{s}_\perp\sim\Phi^*(h)} \left|\bar{\mathcal{R}}_\theta(\phi_\iota(s,o),a) - \bar{\mathcal{R}}_\theta(\bar{s}_\perp,a)\right| \\
&\quad + \mathop{\mathbb{E}}_{\bar{s}\sim\bar{\mathcal{T}}^*(h)} \mathop{\mathbb{E}}_{\bar{s}_\perp\sim\Phi^*(h)} \left|\bar{\mathcal{R}}_\theta(\bar{s}_\perp,a) - \bar{\mathcal{R}}_\theta(\bar{s},a)\right| \qquad \text{(Triangular and Jensen Inequalities)}
\end{aligned}
$$

By reintroducing the expectation over histories and actions we recognize the definitions of $L_{\mathcal{R}}$ and $L^{\Phi}_{\bar{\mathcal{R}}}$ Eq. (5.3).

$$
\begin{aligned}
&\mathop{\mathbb{E}}_{h\sim\mathcal{H}_{\bar{\pi}}} \mathop{\mathbb{E}}_{a\sim\bar{\pi}(\cdot|\Phi^*(h))} \left| \mathop{\mathbb{E}}_{s\sim\mathcal{T}^*(h)} \mathcal{R}(s,a) - \mathop{\mathbb{E}}_{\bar{s}\sim\bar{\mathcal{T}}^*(h)} \bar{\mathcal{R}}_\theta(\bar{s},a) \right| \\
\leq& L_{\mathcal{R}} + L^{\Phi}_{\bar{\mathcal{R}}} + \mathop{\mathbb{E}}_{h,o\sim\mathcal{H}_{\bar{\pi}}} \mathop{\mathbb{E}}_{a\sim\bar{\pi}(\cdot|\Phi^*(h))} \left| \mathop{\mathbb{E}}_{\bar{s}\sim\bar{\mathcal{T}}^*(h)} \mathop{\mathbb{E}}_{\bar{s}_\perp\sim\Phi^*(h)} \left[\bar{\mathcal{R}}_\theta(\bar{s}_\perp,a) - \bar{\mathcal{R}}_\theta(\bar{s},a)\right] \right| \\
\leq& L_{\mathcal{R}} + L^{\Phi}_{\bar{\mathcal{R}}} + \mathop{\mathbb{E}}_{h\sim\mathcal{H}_{\bar{\pi}}} \mathrm{R}_{\mathrm{MAX}} \mathcal{W}_{\bar{d}} \left(\bar{\mathcal{T}}^*(h), \Phi^*(h)\right) \qquad \text{(as } \lambda \to 0\text{, by Lem. 5.2 and Prop. 5.1)} \\
=& L_{\mathcal{R}} + L^{\Phi}_{\bar{\mathcal{R}}} + \mathrm{R}_{\mathrm{MAX}} L_{\bar{\mathcal{T}}};
\end{aligned}
$$

Part 2: Next value bounds

$$
\mathop{\mathbb{E}}_{h\sim\mathcal{H}_{\bar{\pi}}} \mathop{\mathbb{E}}_{a\sim\bar{\pi}(\cdot|\Phi^*(h))} \left| \mathop{\mathbb{E}}_{s\sim\mathcal{T}^*(h)} \mathop{\mathbb{E}}_{s'\sim\mathrm{P}(\cdot|s,a)} \mathop{\mathbb{E}}_{o'\sim\mathcal{O}(\cdot|s',a)} V^{\bar{\pi}}\left(h\cdot a\cdot o'\right) \right.
$$

$$- \mathop{\mathbb{E}}_{\bar{s}\sim\overline{\mathcal{T}}^*(h)} \mathop{\mathbb{E}}_{\bar{s}'\sim\overline{\mathbf{P}}_\theta(\cdot|\bar{s},a)} \mathop{\mathbb{E}}_{o'\sim\overline{\mathcal{O}}_\theta(\cdot|\bar{s}')} \overline{V}^{\bar{\pi}}\big(h\cdot a\cdot o'\big)\Big|$$

As before, to simplify the notation, we focus on the absolute part by setting $h \sim \mathcal{H}_{\bar{\pi}}$, o as the last observation of h, and $a \sim \bar{\pi}(\cdot \mid \Phi^*(h))$. And we leverage the equality between $s', o' \sim \mathbf{P}_\Omega(\cdot \mid s, o, a)$ and $s' \sim \mathbf{P}(\cdot \mid s, a)$ followed by $o' \sim \mathcal{O}(\cdot \mid s', a)$. We define a similar notation for the latent POMDP $\mathcal{P}_\theta$ with $\overline{\mathbf{P}}_\theta(\bar{s}' \mid \bar{s}, a)\overline{\mathcal{O}}_\theta(o' \mid \bar{s}') = \overline{\mathbf{P}}_{\theta,\Omega}(\bar{s}', o' \mid \bar{s}, a)$. $\bar{s}', o' \sim \overline{\mathbf{P}}_{\theta,\Omega}(\cdot \mid \bar{s}, a)$.

$$\begin{aligned}
&\left|\mathop{\mathbb{E}}_{s\sim\mathcal{T}^*(h)} \mathop{\mathbb{E}}_{s',o'\sim\mathbf{P}_\Omega(\cdot|s,o,a)} V^{\bar{\pi}}\big(h\cdot a\cdot o'\big) - \mathop{\mathbb{E}}_{\bar{s}\sim\overline{\mathcal{T}}^*(h)} \mathop{\mathbb{E}}_{\bar{s}',o'\sim\overline{\mathbf{P}}_{\theta,\Omega}(\cdot|\bar{s},a)} \overline{V}^{\bar{\pi}}\big(h\cdot a\cdot o'\big)\right| \\
=&\left|\mathop{\mathbb{E}}_{s\sim\mathcal{T}^*(h)} \mathop{\mathbb{E}}_{s',o'\sim\mathbf{P}_\Omega(\cdot|s,o,a)} \left[V^{\bar{\pi}}\big(h\cdot a\cdot o'\big) - \mathop{\mathbb{E}}_{\hat{o}'\sim\overline{\mathcal{O}}_\theta(\cdot|\phi_\iota(s',o'))} \overline{V}^{\bar{\pi}}\big(h\cdot a\cdot \hat{o}'\big)\right]\right. \\
&\left. + \left[\mathop{\mathbb{E}}_{s\sim\mathcal{T}^*(h)} \mathop{\mathbb{E}}_{\bar{s}',\sim\phi\mathbf{P}_\Omega(\cdot|s,o,a)} \mathop{\mathbb{E}}_{\hat{o}'\sim\overline{\mathcal{O}}_\theta(\cdot|\bar{s}')} \overline{V}^{\bar{\pi}}\big(h\cdot a\cdot \hat{o}'\big) - \mathop{\mathbb{E}}_{\bar{s}\sim\overline{\mathcal{T}}^*(h)} \mathop{\mathbb{E}}_{\bar{s}',o'\sim\overline{\mathbf{P}}_{\theta,\Omega}(\cdot|\bar{s},a)} \overline{V}^{\bar{\pi}}\big(h\cdot a\cdot o'\big)\right]\right| \\
&\text{(the state embedding function } \phi_\iota \text{ comes into play, as well as the latent} \\
&\text{observation function } \overline{\mathcal{O}}_\theta) \\
\leq&\left|\mathop{\mathbb{E}}_{s\sim\mathcal{T}^*(h)} \mathop{\mathbb{E}}_{s',o'\sim\mathbf{P}_\Omega(\cdot|s,o,a)} \left[V^{\bar{\pi}}\big(h\cdot a\cdot o'\big) - \mathop{\mathbb{E}}_{\hat{o}'\sim\overline{\mathcal{O}}_\theta(\cdot|\phi_\iota(s',o'))} \overline{V}^{\bar{\pi}}\big(h\cdot a\cdot \hat{o}'\big)\right]\right| \qquad (5.7) \\
&+\left|\mathop{\mathbb{E}}_{s\sim\mathcal{T}^*(h)} \mathop{\mathbb{E}}_{\bar{s}',\sim\phi\mathbf{P}_\Omega(\cdot|s,o,a)} \mathop{\mathbb{E}}_{\hat{o}'\sim\overline{\mathcal{O}}_\theta(\cdot|\bar{s}')} \overline{V}^{\bar{\pi}}\big(h\cdot a\cdot \hat{o}'\big) - \mathop{\mathbb{E}}_{\bar{s}\sim\overline{\mathcal{T}}^*(h)} \mathop{\mathbb{E}}_{\bar{s}',o'\sim\overline{\mathbf{P}}_{\theta,\Omega}(\cdot|\bar{s},a)} \overline{V}^{\bar{\pi}}\big(h\cdot a\cdot o'\big)\right| \\
&\text{(Triangular inequality)}
\end{aligned}$$

We first focus on the first absolute value difference in Eq. (5.7), which we can rewrite as follows:

$$\begin{aligned}
&\left|\mathop{\mathbb{E}}_{s\sim\mathcal{T}^*(h)} \mathop{\mathbb{E}}_{s',o'\sim\mathbf{P}_\Omega(\cdot|s,o,a)} \left[V^{\bar{\pi}}\big(h\cdot a\cdot o'\big) - \mathop{\mathbb{E}}_{\hat{o}'\sim\overline{\mathcal{O}}_\theta(\cdot|\phi_\iota(s',o'))} \overline{V}^{\bar{\pi}}\big(h\cdot a\cdot \hat{o}'\big)\right]\right| \\
\leq&\left|\mathop{\mathbb{E}}_{s\sim\mathcal{T}^*(h)} \mathop{\mathbb{E}}_{s',o'\sim\mathbf{P}_\Omega(\cdot|s,o,a)} \left[V^{\bar{\pi}}\big(h\cdot a\cdot o'\big) - \overline{V}^{\bar{\pi}}\big(h\cdot a\cdot o'\big)\right]\right| \\
&+\left|\mathop{\mathbb{E}}_{s\sim\mathcal{T}^*(h)} \mathop{\mathbb{E}}_{s',o'\sim\mathbf{P}_\Omega(\cdot|s,o,a)} \left[\overline{V}^{\bar{\pi}}\big(h\cdot a\cdot o'\big) - \mathop{\mathbb{E}}_{\hat{o}'\sim\overline{\mathcal{O}}_\theta(\cdot|\phi_\iota(s',o'))} \overline{V}^{\bar{\pi}}\big(h\cdot a\cdot \hat{o}'\big)\right]\right| \\
&\text{(Triangular inequality)}
\end{aligned}$$

$$\leq \mathop{\mathbb{E}}_{s\sim\mathcal{T}^*(h)} \mathop{\mathbb{E}}_{s',o'\sim\mathbf{P}_\Omega(\cdot|s,o,a)} \left|V^{\bar{\pi}}\left(h\cdot a\cdot o'\right) - \overline{V}^{\bar{\pi}}\left(h\cdot a\cdot o'\right)\right|$$
$$+ \mathop{\mathbb{E}}_{s\sim\mathcal{T}^*(h)} \mathop{\mathbb{E}}_{s'\sim\mathbf{P}(\cdot|s,a)} \left|\mathop{\mathbb{E}}_{o'\sim\mathcal{O}(\cdot|s',a)} \left[\overline{V}^{\bar{\pi}}\left(h\cdot a\cdot o'\right) - \mathop{\mathbb{E}}_{\hat{o}'\sim\overline{\mathcal{O}}_\theta(\cdot|\phi_\iota(s',o'))} \overline{V}^{\bar{\pi}}\left(h\cdot a\cdot \hat{o}'\right)\right]\right|$$
(by definition of $\mathbf{P}_\Omega$ and the Jensen's inequality)

$$\leq \mathop{\mathbb{E}}_{s\sim\mathcal{T}^*(h)} \mathop{\mathbb{E}}_{s',o'\sim\mathbf{P}_\Omega(\cdot|s,o,a)} \left|V^{\bar{\pi}}\left(h\cdot a\cdot o'\right) - \overline{V}^{\bar{\pi}}\left(h\cdot a\cdot o'\right)\right|$$
$$+ K_{\overline{V}}\, d_{TV}\left(\mathcal{O}\big(\cdot \mid s', a\big), \mathop{\mathbb{E}}_{o'\sim s',a} \overline{\mathcal{O}}_\theta\big(\cdot \mid \phi_\iota\big(s', o'\big)\big)\right) \qquad \text{(cf. Prop. 5.1 and Lem 5.2)}$$

Let us now focus on the second absolute value difference in Eq. (5.7), which we can rewrite as follows:

$$= \left| \mathop{\mathbb{E}}_{s\sim\mathcal{T}^*(h)} \left[\mathop{\mathbb{E}}_{\bar{s}',\sim\phi\mathbf{P}_\Omega(\cdot|s,o,a)} \mathop{\mathbb{E}}_{\hat{o}'\sim\overline{\mathcal{O}}_\theta(\cdot|\bar{s}')} \overline{V}^{\bar{\pi}}\left(h\cdot a\cdot \hat{o}'\right) - \mathop{\mathbb{E}}_{\bar{s}'\sim\overline{\mathbf{P}}_\theta(\cdot|\phi_\iota(s,o),a)} \mathop{\mathbb{E}}_{\hat{o}'\sim\overline{\mathcal{O}}_\theta(\cdot|\bar{s}')} \overline{V}^{\bar{\pi}}\left(h\cdot a\cdot \hat{o}'\right) \right] \right.$$
$$\left. + \left[\mathop{\mathbb{E}}_{s\sim\mathcal{T}^*(h)} \mathop{\mathbb{E}}_{\bar{s}'\sim\overline{\mathbf{P}}_\theta(\cdot|\phi_\iota(s,o),a)} \mathop{\mathbb{E}}_{o'\sim\overline{\mathcal{O}}_\theta(\cdot|\bar{s}')} \overline{V}^{\bar{\pi}}\left(h\cdot a\cdot o'\right) - \mathop{\mathbb{E}}_{\bar{s}\sim\overline{\mathcal{T}}^*(h)} \mathop{\mathbb{E}}_{\bar{s}',o'\sim\overline{\mathbf{P}}_{\theta,\Omega}(\cdot|\bar{s},a)} \overline{V}^{\bar{\pi}}\left(h\cdot a\cdot o'\right) \right] \right|$$
(the latent MDP dynamics, modeled by $\overline{\mathbf{P}}_\theta$, come into play)

$$\leq \mathop{\mathbb{E}}_{s\sim\mathcal{T}^*(h)} \left| \mathop{\mathbb{E}}_{\bar{s}'\sim\phi_\iota\mathbf{P}_\Omega(\cdot|s,o,a)} \mathop{\mathbb{E}}_{o'\sim\overline{\mathcal{O}}_\theta(\cdot|\bar{s}')} \overline{V}^{\bar{\pi}}\left(h\cdot a\cdot o'\right) - \mathop{\mathbb{E}}_{\bar{s}'\sim\overline{\mathbf{P}}_\theta(\cdot|\phi_\iota(s,o),a)} \mathop{\mathbb{E}}_{o'\sim\overline{\mathcal{O}}_\theta(\cdot|\bar{s}')} \overline{V}^{\bar{\pi}}\left(h\cdot a\cdot o'\right) \right|$$
$$+ \left| \mathop{\mathbb{E}}_{s\sim\mathcal{T}^*(h)} \mathop{\mathbb{E}}_{\bar{s}'\sim\overline{\mathbf{P}}_\theta(\cdot|\phi_\iota(s,o),a)} \mathop{\mathbb{E}}_{o'\sim\overline{\mathcal{O}}_\theta(\cdot|\bar{s}')} \overline{V}^{\bar{\pi}}\left(h\cdot a\cdot o'\right) - \mathop{\mathbb{E}}_{\bar{s}\sim\overline{\mathcal{T}}^*(h)} \mathop{\mathbb{E}}_{\bar{s}',o'\sim\overline{\mathbf{P}}_{\theta,\Omega}(\cdot|\bar{s},a)} \overline{V}^{\bar{\pi}}\left(h\cdot a\cdot o'\right) \right|$$
(Triangular and Jensen's inequality)

$$\leq \mathop{\mathbb{E}}_{s\sim\mathcal{T}^*(h)} K_{\overline{V}} \cdot \mathcal{W}_{\bar{d}}\left(\phi_\iota\mathbf{P}_\Omega(\cdot \mid s, a), \overline{\mathbf{P}}_\theta(\cdot \mid \phi_\iota(s, o), a)\right)$$
$$+ \left| \mathop{\mathbb{E}}_{s\sim\mathcal{T}^*(h)} \mathop{\mathbb{E}}_{\bar{s}'\sim\overline{\mathbf{P}}_\theta(\cdot|\phi_\iota(s,o),a)} \mathop{\mathbb{E}}_{o'\sim\overline{\mathcal{O}}_\theta(\cdot|\bar{s}')} \overline{V}^{\bar{\pi}}\left(h\cdot a\cdot o'\right) - \mathop{\mathbb{E}}_{\bar{s}\sim\overline{\mathcal{T}}^*(h)} \mathop{\mathbb{E}}_{\bar{s}',o'\sim\overline{\mathbf{P}}_{\theta,\Omega}(\cdot|\bar{s},a)} \overline{V}^{\bar{\pi}}\left(h\cdot a\cdot o'\right) \right|$$
(as $\lambda \to 0$, by Lem. 5.2)

$$\leq \mathop{\mathbb{E}}_{s\sim\mathcal{T}^*(h)} K_{\overline{V}} \cdot \mathcal{W}_{\bar{d}}\left(\phi_\iota\mathbf{P}_\Omega(\cdot \mid s, a), \overline{\mathbf{P}}_\theta(\cdot \mid \phi_\iota(s, o), a)\right)$$
$$+ \left| \mathop{\mathbb{E}}_{s\sim\mathcal{T}^*(h)} \mathop{\mathbb{E}}_{\bar{s}'\sim\overline{\mathbf{P}}_\theta(\cdot|\phi_\iota(s,o),a)} \mathop{\mathbb{E}}_{o'\sim\overline{\mathcal{O}}_\theta(\cdot|\bar{s}')} \overline{V}^{\bar{\pi}}\left(h\cdot a\cdot o'\right) - \mathop{\mathbb{E}}_{\bar{s}\sim\Phi^*(h)} \mathop{\mathbb{E}}_{\bar{s}'\sim\overline{\mathbf{P}}_\theta(\cdot|\bar{s},a)} \mathop{\mathbb{E}}_{o'\sim\overline{\mathcal{O}}_\theta(\cdot|\bar{s}')} \overline{V}^{\bar{\pi}}\left(h\cdot a\cdot o'\right) \right|$$

$$+\left|\mathop{\mathbb{E}}_{\bar{s}\sim\Phi^*(h)}\mathop{\mathbb{E}}_{\bar{s}'\sim\overline{\mathbf{P}}_\theta(\cdot|\bar{s},a)}\mathop{\mathbb{E}}_{o'\sim\overline{\mathcal{O}}_\theta(\cdot|\bar{s}')}\overline{V}^{\overline{\pi}}\left(h\cdot a\cdot o'\right)-\mathop{\mathbb{E}}_{\bar{s}\sim\overline{\mathcal{T}}^*(h)}\mathop{\mathbb{E}}_{\bar{s}'\sim\overline{\mathbf{P}}_\theta(\cdot|\bar{s},a)}\mathop{\mathbb{E}}_{o'\sim\overline{\mathcal{O}}_\theta(\cdot|\bar{s}')}\overline{V}^{\overline{\pi}}\left(h\cdot a\cdot o'\right)\right|$$

(the belief encoder Φ comes into play, followed by Triangular Inequality)

$$\leq \mathop{\mathbb{E}}_{s\sim\mathcal{T}^*(h)} K_{\overline{V}}\cdot\mathcal{W}_{\bar{d}}\left(\phi_\iota\mathbf{P}_\Omega(\cdot\mid s,a),\overline{\mathbf{P}}_\theta(\cdot\mid\phi_\iota(s,o),a)\right)$$

$$+\mathop{\mathbb{E}}_{s\sim\mathcal{T}^*(h)}\mathop{\mathbb{E}}_{\bar{s}\sim\Phi^*(h)}\left|\mathop{\mathbb{E}}_{\bar{s}'\sim\overline{\mathbf{P}}_\theta(\cdot|\phi_\iota(s,o),a)}\mathop{\mathbb{E}}_{o'\sim\overline{\mathcal{O}}_\theta(\cdot|\bar{s}')}\overline{V}^{\overline{\pi}}\left(h\cdot a\cdot o'\right)-\mathop{\mathbb{E}}_{\bar{s}'\sim\overline{\mathbf{P}}_\theta(\cdot|\bar{s},a)}\mathop{\mathbb{E}}_{o'\sim\overline{\mathcal{O}}_\theta(\cdot|\bar{s}')}\overline{V}^{\overline{\pi}}\left(h\cdot a\cdot o'\right)\right|$$

$$+\left|\mathop{\mathbb{E}}_{\bar{s}\sim\Phi^*(h)}\mathop{\mathbb{E}}_{\bar{s}'\sim\overline{\mathbf{P}}_\theta(\cdot|\bar{s},a)}\mathop{\mathbb{E}}_{o'\sim\overline{\mathcal{O}}_\theta(\cdot|\bar{s}')}\overline{V}^{\overline{\pi}}\left(h\cdot a\cdot o'\right)-\mathop{\mathbb{E}}_{\bar{s}\sim\overline{\mathcal{T}}^*(h)}\mathop{\mathbb{E}}_{\bar{s}'\sim\overline{\mathbf{P}}_\theta(\cdot|\bar{s},a)}\mathop{\mathbb{E}}_{o'\sim\overline{\mathcal{O}}_\theta(\cdot|\bar{s}')}\overline{V}^{\overline{\pi}}\left(h\cdot a\cdot o'\right)\right|$$

(Jensen's Inequality)

$$\leq \mathop{\mathbb{E}}_{s\sim\mathcal{T}^*(h)} K_{\overline{V}}\cdot\mathcal{W}_{\bar{d}}\left(\phi_\iota\mathbf{P}_\Omega(\cdot\mid s,a),\overline{\mathbf{P}}_\theta(\cdot\mid\phi_\iota(s,o),a)\right)$$

$$+\mathop{\mathbb{E}}_{s\sim\mathcal{T}^*(h)}\mathop{\mathbb{E}}_{\bar{s}\sim\Phi^*(h)} K_{\overline{V}}\,\mathcal{W}_{\bar{d}}\left(\overline{\mathbf{P}}_\theta(\cdot\mid\phi_\iota(s,o),a),\overline{\mathbf{P}}_\theta(\cdot\mid\bar{s},a)\right)+K_{\overline{V}}\,\mathcal{W}_{\bar{d}}\left(\overline{\mathcal{T}}^*(h),\Phi^*(h)\right)$$

(as $\lambda\to 0$, by Lem. 5.2)

Now that we computed upper bounds for the two absolute value differences in Eq. (5.7), we can combine them to upper bound the next value founds as follows:

$$\begin{aligned}
&\mathop{\mathbb{E}}_{h\sim\mathcal{H}_{\overline{\pi}}}\mathop{\mathbb{E}}_{a\sim\overline{\pi}(\cdot|\Phi^*(h))}\Bigg|\mathop{\mathbb{E}}_{s\sim\mathcal{T}^*(h)}\mathop{\mathbb{E}}_{s'\sim\mathbf{P}(\cdot|s,a)}\mathop{\mathbb{E}}_{o'\sim\mathcal{O}(\cdot|s',a)}V^{\overline{\pi}}\left(h\cdot a\cdot o'\right)\\
&\qquad\qquad-\mathop{\mathbb{E}}_{\bar{s}\sim\overline{\mathcal{T}}^*(h)}\mathop{\mathbb{E}}_{\bar{s}'\sim\overline{\mathbf{P}}_\theta(\cdot|\bar{s},a)}\mathop{\mathbb{E}}_{o'\sim\overline{\mathcal{O}}_\theta(\cdot|\bar{s}')}\overline{V}^{\overline{\pi}}\left(h\cdot a\cdot o'\right)\Bigg|\\
&\leq \mathop{\mathbb{E}}_{h\sim\mathcal{H}_{\overline{\pi}}}\mathop{\mathbb{E}}_{a\sim\overline{\pi}(\cdot|\Phi^*(h))}\mathop{\mathbb{E}}_{s\sim\mathcal{T}^*(h)}\mathop{\mathbb{E}}_{s',o'\sim\mathbf{P}_\Omega(\cdot|s,o,a)}\left|V^{\overline{\pi}}\left(h\cdot a\cdot o'\right)-\overline{V}^{\overline{\pi}}\left(h\cdot a\cdot o'\right)\right|\\
&\quad+\mathop{\mathbb{E}}_{h\sim\mathcal{H}_{\overline{\pi}}}\mathop{\mathbb{E}}_{a\sim\overline{\pi}(\cdot|\Phi^*(h))} K_{\overline{V}}\,d_{TV}\left(\mathcal{O}\left(\cdot\mid s',a\right),\mathop{\mathbb{E}}_{o'\sim s',a}\overline{\mathcal{O}}_\theta\left(\cdot\mid\phi_\iota\left(s',o'\right)\right)\right)\\
&\quad+\mathop{\mathbb{E}}_{h\sim\mathcal{H}_{\overline{\pi}}}\mathop{\mathbb{E}}_{a\sim\overline{\pi}(\cdot|\Phi^*(h))}\mathop{\mathbb{E}}_{s\sim\mathcal{T}^*(h)} K_{\overline{V}}\cdot\mathcal{W}_{\bar{d}}\left(\phi_\iota\mathbf{P}_\Omega(\cdot\mid s,a),\overline{\mathbf{P}}_\theta(\cdot\mid\phi_\iota(s,o),a)\right)\\
&\quad+\mathop{\mathbb{E}}_{h\sim\mathcal{H}_{\overline{\pi}}}\mathop{\mathbb{E}}_{a\sim\overline{\pi}(\cdot|\Phi^*(h))}\mathop{\mathbb{E}}_{s\sim\mathcal{T}^*(h)}\mathop{\mathbb{E}}_{\bar{s}\sim\Phi^*(h)} K_{\overline{V}}\,\mathcal{W}_{\bar{d}}\left(\overline{\mathbf{P}}_\theta(\cdot\mid\phi_\iota(s,o),a),\overline{\mathbf{P}}_\theta(\cdot\mid\bar{s},a)\right)\\
&\quad+\mathop{\mathbb{E}}_{h\sim\mathcal{H}_{\overline{\pi}}}\mathop{\mathbb{E}}_{a\sim\overline{\pi}(\cdot|\Phi^*(h))} K_{\overline{V}}\,\mathcal{W}_{\bar{d}}\left(\overline{\mathcal{T}}^*(h),\Phi^*(h)\right)\\
&\leq \mathop{\mathbb{E}}_{h\sim\mathcal{H}_{\overline{\pi}}}\mathop{\mathbb{E}}_{a\sim\overline{\pi}(\cdot|\Phi^*(h))}\mathop{\mathbb{E}}_{s\sim\mathcal{T}^*(h)}\mathop{\mathbb{E}}_{s',o'\sim\mathbf{P}_\Omega(\cdot|s,o,a)}\left|V^{\overline{\pi}}\left(h\cdot a\cdot o'\right)-\overline{V}^{\overline{\pi}}\left(h\cdot a\cdot o'\right)\right|\\
&\quad+K_{\overline{V}}\cdot\left(L_{\mathbf{P}}+L_{\overline{\mathbf{P}}}^{\Phi}+L_{\overline{\mathcal{T}}}+L_{\mathcal{O}}\right)\qquad\text{(by definition of } L_{\mathbf{P}},\, L_{\overline{\mathbf{P}}}^{\Phi},\, L_{\overline{\mathcal{T}}},\text{ and } L_{\mathcal{O}})
\end{aligned}$$

$$\leq \mathop{\mathbb{E}}_{h,o \sim \mathcal{H}_{\overline{\pi}}} \left| V^{\overline{\pi}}(h) - \overline{V}^{\overline{\pi}}(h) \right| + K_{\overline{V}} \cdot \left(L_{\mathbf{P}} + L_{\overline{\mathbf{P}}}^{\Phi} + L_{\overline{\mathcal{T}}} + L_{\mathcal{O}} \right)$$

(We apply the stationnary property of $\mathcal{H}_{\overline{\pi}}$)

Putting all together. To recap, by Part 1 and 2, we have:

$$\begin{aligned}
\mathop{\mathbb{E}}_{h \sim \mathcal{H}_{\overline{\pi}}} \left| V^{\overline{\pi}}(h) - \overline{V}^{\overline{\pi}}(h) \right| &\leq L_{\mathcal{R}} + L_{\mathcal{R}}^{\Phi} + \mathrm{R}_{\mathrm{MAX}} L_{\overline{\mathcal{T}}} + \gamma \cdot \mathop{\mathbb{E}}_{h \sim \mathcal{H}_{\overline{\pi}}} \left| V^{\overline{\pi}}(h) - \overline{V}^{\overline{\pi}}(h) \right| \\
&\quad + \gamma K_{\overline{V}} \cdot \left(L_{\mathbf{P}} + L_{\overline{\mathbf{P}}}^{\Phi} + L_{\overline{\mathcal{T}}} + L_{\mathcal{O}} \right) \\
\mathop{\mathbb{E}}_{h \sim \mathcal{H}_{\overline{\pi}}} \left| V^{\overline{\pi}}(h) - \overline{V}^{\overline{\pi}}(h) \right| \cdot (1 - \gamma) &\leq L_{\mathcal{R}} + L_{\mathcal{R}}^{\Phi} + \mathrm{R}_{\mathrm{MAX}} L_{\overline{\mathcal{T}}} + \gamma K_{\overline{V}} \cdot \left(L_{\mathbf{P}} + L_{\overline{\mathbf{P}}}^{\Phi} + L_{\overline{\mathcal{T}}} + L_{\mathcal{O}} \right) \\
\mathop{\mathbb{E}}_{h \sim \mathcal{H}_{\overline{\pi}}} \left| V^{\overline{\pi}}(h) - \overline{V}^{\overline{\pi}}(h) \right| &\leq \frac{L_{\mathcal{R}} + L_{\mathcal{R}}^{\Phi} + \mathrm{R}_{\mathrm{MAX}} L_{\overline{\mathcal{T}}} + \gamma K_{\overline{V}} \cdot \left(L_{\mathbf{P}} + L_{\overline{\mathbf{P}}}^{\Phi} + L_{\overline{\mathcal{T}}} + L_{\mathcal{O}} \right)}{1 - \gamma}
\end{aligned}$$

which concludes the proof. □

Theorem 5.2: Representation quality

Let $\overline{\pi}^{\star}$ be an optimal policy of $\overline{\mathcal{P}}_{\theta}$, then for any $\epsilon > 0$ and $n \geq 0$, there is a $K \geq 0$ so that for any histories h_1, h_2 of length at most n that are measurable under $\mathcal{P}$ and $\overline{\mathcal{P}}$ with $\Phi^{*}(h_1) = \overline{b}_1$ and $\Phi^{*}(h_2) = \overline{b}_2$, the representation induced by Φ yields:

$$\left| V_{\overline{\pi}^{\star}}(h_1) - V_{\overline{\pi}^{\star}}(h_2) \right| \leq K \mathcal{W}_{\overline{d}} \left(\overline{b}_1, \overline{b}_2 \right) + \epsilon + \frac{K L_{\overline{\mathcal{T}}} + \mathcal{L}}{1 - \gamma} \left(\frac{1}{\mathcal{H}_{\overline{\pi}^{\star}}(h_1)} + \frac{1}{\mathcal{H}_{\overline{\pi}^{\star}}(h_2)} \right). \tag{5.8}$$

Proof. First, observe that for any measurable history h,

$$\left| V_{\overline{\pi}^{\star}}(h) - \overline{V}_{\overline{\pi}^{\star}}(h) \right| \leq \mathcal{H}_{\overline{\pi}^{\star}}(h)^{-1} \cdot \mathop{\mathbb{E}}_{h' \sim \mathcal{H}_{\overline{\pi}^{\star}}} \left| V_{\overline{\pi}^{\star}}\left(h'\right) - \overline{V}_{\overline{\pi}^{\star}}\left(h'\right) \right| \qquad \text{(Gelada et al. 2019)}$$

Let $\epsilon > 0$, $n \geq 0$, and h_1, h_2 be two measurable histories so that $h_i = \langle a_{0:T-1}, o_{1:T} \rangle$ with $T \leq n$ and $o_t \in \operatorname{supp}\left(\overline{\mathcal{O}}(\cdot \mid \overline{s}) \right)$ for some $\overline{s} \in \overline{\mathcal{S}}$ and for all $t < T$, $i \in \{1, 2\}$ (the two histories are compliant with $\mathcal{P}$ and $\overline{\mathcal{P}}$). Then, we have:

$$\begin{aligned}
&\left| V_{\overline{\pi}^{\star}}(h_1) - V_{\overline{\pi}^{\star}}(h_2) \right| \\
=&\left| V_{\overline{\pi}^{\star}}(h_1) - \overline{V}_{\overline{\pi}^{\star}}(h_1) + \overline{V}_{\overline{\pi}^{\star}}(h_1) - \overline{V}_{\overline{\pi}^{\star}}(h_2) + \overline{V}_{\overline{\pi}^{\star}}(h_2) - V_{\overline{\pi}^{\star}}(h_2) \right| \\
\leq&\left| V_{\overline{\pi}^{\star}}(h_1) - \overline{V}_{\overline{\pi}^{\star}}(h_1) \right| + \left| \overline{V}_{\overline{\pi}^{\star}}(h_1) - \overline{V}_{\overline{\pi}^{\star}}(h_2) \right| + \left| \overline{V}_{\overline{\pi}^{\star}}(h_2) - V_{\overline{\pi}^{\star}}(h_2) \right|
\end{aligned}$$

(Triangular inequality)

$$\leq \mathcal{H}_{\overline{\pi}^{\star}}(h_1)^{-1} \mathop{\mathbb{E}}_{h \sim \mathcal{H}_{\overline{\pi}^{\star}}} \left| V_{\overline{\pi}^{\star}}(h) - \overline{V}_{\overline{\pi}^{\star}}(h) \right| + \left| \overline{V}_{\overline{\pi}^{\star}}(h_1) - \overline{V}_{\overline{\pi}^{\star}}(h_2) \right|$$

$$
\begin{aligned}
&\quad + \mathcal{H}_{\bar{\pi}^\star}(h_2)^{-1} \mathop{\mathbb{E}}_{h\sim\mathcal{H}_{\bar{\pi}^\star}} \left|V_{\bar{\pi}^\star}(h) - \overline{V}_{\bar{\pi}^\star}(h)\right| \\
&\leq \left|\overline{V}_{\bar{\pi}^\star}(h_1) - \overline{V}_{\bar{\pi}^\star}(h_2)\right| + \frac{\mathcal{L}}{1-\gamma}\left(\mathcal{H}_{\bar{\pi}^\star}(h_1)^{-1} + \mathcal{H}_{\bar{\pi}^\star}(h_2)^{-1}\right) \qquad \text{(Thm. 5.1)}
\end{aligned}
$$

Consider $h_\perp = \arg\min\left\{\mathcal{H}_{\bar{\pi}^\star}(h) : h \in \text{supp}(\mathcal{H}_{\bar{\pi}^\star}) \cap \mathbf{H}\left(\mathcal{P}, \overline{\mathcal{P}}\right) \text{ and } |h| \leq n\right\}$, where $\text{supp}(\mathcal{H}_{\bar{\pi}^\star})$ denotes the support of $\mathcal{H}_{\bar{\pi}^\star}$, $|h|$ the length of the history h, and $\mathbf{H}\left(\mathcal{P}, \overline{\mathcal{P}}\right)$ the set of histories compliant with both $\mathcal{P}$ and $\overline{\mathcal{P}}$, i.e., histories whose observations belong to $\bigcup_{\bar{s}\in\overline{S}} \text{supp}\left(\overline{\mathcal{O}}(\cdot \mid \bar{s})\right)$.
Notice that $h_\perp$ exists: we consider histories which are measurable in the two models (i.e., the original and latent POMDPs) and the observation space of the latent POMDP is finite (cf. the premise of Corollary 5.1). So, the set of histories from which we extract $h_\perp$ consists of a finite number of action-observation branchings.
Recall that the latent value function (defined over the latent belief space) is almost Lipschitz continuous (Corollary 5.1). In particular, for $\delta = \frac{\epsilon}{\left(1+2\mathcal{H}_{\bar{\pi}^\star}(h_\perp)^{-1}\right)}$, there is a $K \geq 0$ so that for any $\bar{b}, \bar{b}' \in \overline{\mathcal{B}}$, $\left|\overline{V}^\star(b) - \overline{V}^\star(b')\right| \leq K\mathcal{W}_{\bar{d}}(b, b') + \delta$. Then:

$$
\begin{aligned}
&\left|\overline{V}^\star(h_1) - \overline{V}^\star(h_2)\right| \\
&= \left|\overline{V}^\star\left(\overline{\mathcal{T}}^*(h_1)\right) - \overline{V}^\star\left(\overline{\mathcal{T}}^*(h_2)\right)\right| \\
&= \left|\overline{V}^\star\left(\overline{\mathcal{T}}^*(h_1)\right) - \overline{V}^\star(\Phi^*(h_1)) + \overline{V}^\star(\Phi^*(h_1)) - \overline{V}^\star(\Phi^*(h_2)) + \overline{V}^\star(\Phi^*(h_2)) - \overline{V}^\star\left(\overline{\mathcal{T}}^*(h_2)\right)\right| \\
&\leq \left|\overline{V}^\star\left(\overline{\mathcal{T}}^*(h_1)\right) - \overline{V}^\star(\Phi^*(h_1))\right| + \left|\overline{V}^\star(\Phi^*(h_1)) - \overline{V}^\star(\Phi^*(h_2))\right| + \left|\overline{V}^\star(\Phi^*(h_2)) - \overline{V}^\star\left(\overline{\mathcal{T}}^*(h_2)\right)\right| \\
&\qquad\qquad \text{(Triangular inequality)} \\
&\leq \mathcal{H}_{\bar{\pi}^\star}(h_1)^{-1} \mathop{\mathbb{E}}_{h\sim\mathcal{H}_{\bar{\pi}^\star}} \left|\overline{V}^\star\left(\overline{\mathcal{T}}^*(h)\right) - \overline{V}^\star(\Phi^*(h))\right| + \left|\overline{V}^\star(b_1) - \overline{V}^\star(b_2)\right| \\
&\quad + \mathcal{H}_{\bar{\pi}^\star}(h_2)^{-1} \mathop{\mathbb{E}}_{h\sim\mathcal{H}_{\bar{\pi}^\star}} \left|\overline{V}^\star\left(\overline{\mathcal{T}}^*(h)\right) - \overline{V}^\star(\Phi^*(h))\right| \\
&= \left(\mathcal{H}_{\bar{\pi}^\star}(h_1)^{-1} + \mathcal{H}_{\bar{\pi}^\star}(h_2)^{-1}\right) \mathop{\mathbb{E}}_{h\sim\mathcal{H}_{\bar{\pi}^\star}} \left|\overline{V}^\star\left(\overline{\mathcal{T}}^*(h)\right) - \overline{V}^\star(\Phi^*(h))\right| + \left|\overline{V}^\star(b_1) - \overline{V}^\star(b_2)\right| \\
&\leq \left(\mathcal{H}_{\bar{\pi}^\star}(h_1)^{-1} + \mathcal{H}_{\bar{\pi}^\star}(h_2)^{-1}\right) \mathop{\mathbb{E}}_{h\sim\mathcal{H}_{\bar{\pi}^\star}} \left[K\mathcal{W}_{\bar{d}}\left(\overline{\mathcal{T}}^*(h), \Phi^*(h)\right) + \delta\right] + K\mathcal{W}_{\bar{d}}\left(\bar{b}_1, \bar{b}_2\right) + \delta \\
&\qquad\qquad (\overline{V}^\star \text{ is almost Lipschitz}) \\
&= \left(\mathcal{H}_{\bar{\pi}^\star}(h_1)^{-1} + \mathcal{H}_{\bar{\pi}^\star}(h_2)^{-1}\right)\left(KL_{\overline{\mathcal{T}}} + \delta\right) + K\mathcal{W}_{\bar{d}}\left(\bar{b}_1, \bar{b}_2\right) + \delta \qquad \text{(by definition of } L_{\overline{\mathcal{T}}}) \\
&= \left(\mathcal{H}_{\bar{\pi}^\star}(h_1)^{-1} + \mathcal{H}_{\bar{\pi}^\star}(h_2)^{-1}\right)KL_{\overline{\mathcal{T}}} + K\mathcal{W}_{\bar{d}}\left(\bar{b}_1, \bar{b}_2\right) + \delta\left(1 + \mathcal{H}_{\bar{\pi}^\star}(h_1)^{-1} + \mathcal{H}_{\bar{\pi}^\star}(h_2)^{-1}\right) \\
&\leq \left(\mathcal{H}_{\bar{\pi}^\star}(h_1)^{-1} + \mathcal{H}_{\bar{\pi}^\star}(h_2)^{-1}\right)KL_{\overline{\mathcal{T}}} + K\mathcal{W}_{\bar{d}}\left(\bar{b}_1, \bar{b}_2\right) + \delta\left(1 + 2\mathcal{H}_{\bar{\pi}^\star}(h_\perp)^{-1}\right) \\
&= \left(\mathcal{H}_{\bar{\pi}^\star}(h_1)^{-1} + \mathcal{H}_{\bar{\pi}^\star}(h_2)^{-1}\right)KL_{\overline{\mathcal{T}}} + K\mathcal{W}_{\bar{d}}\left(\bar{b}_1, \bar{b}_2\right) + \epsilon
\end{aligned}
$$

Putting all together, we have that for any $\epsilon > 0$, and any maximum history length $n \geq 0$ for h_1 and h_2, one can find a constant $K \geq 0$ so that:

$$|V_{\bar{\pi}^*}(h_1) - V_{\bar{\pi}^*}(h_2)| \leq K\mathcal{W}_{\bar{d}}\left(\bar{b}_1, \bar{b}_2\right) + \varepsilon + \frac{KL_{\bar{\mathcal{T}}} + \mathcal{L}}{1-\gamma}\left(\mathcal{H}_{\bar{\pi}^*}(h_1)^{-1} + \mathcal{H}_{\bar{\pi}^*}(h_2)^{-1}\right).$$

□

Remark 5.1: Representation quality for latent policies

The bound of Theorem 5.2 also holds for any latent policy $\bar{\pi}$ which is executed in both models by processing histories through Φ.

While Theorem 5.1 asserts that training a WAE-MDP as a latent space model of the environment results in similar behaviors (i.e., close expected returns) compared to the original environment when they are measured under the agent policy — which justifies the usage of $\bar{\mathcal{P}}$ as model of the environment — Theorem 5.2 states that our learned update procedure yields a belief representation which is well-suited to optimize the policy: execution traces leading to close latent beliefs (via our learned updater Φ) are guaranteed to yield close expected returns as well.

5.4 Learning to Believe

We now describe how the Wasserstein Belief Updater (WBU) learns to approximate the latent belief dynamics of the POMDP. Throughout this section, we assume access to the latent dynamics learned by the WAE-MDP, namely, the stochastic transition model $\bar{\mathbf{P}}$ and the observation model $\bar{\mathcal{O}}$. The goal is to construct a belief encoder Φ that maps observation-action histories to belief distributions in latent space, enabling downstream RL agents to optimize policies in the belief MDP $\mathcal{M}_B$.

5.4.1 Belief Representation via Sub-beliefs

Our belief encoder is designed to generalize to arbitrary latent POMDPs and must therefore accommodate complex belief distributions. Rather than making parametric assumptions (e.g., unimodal Gaussians), we use a *Masked Auto-Regressive Flow* (MAF) [Papamakarios et al., 2017], a powerful class of normalizing flows capable of modeling flexible, high-dimensional densities. We adopt the relaxed formulation introduced in Delgrange et al. [2023], which is well suited for use with WAE-MDPs and enables theoretical guarantees on value approximation (Section 5.3.2).

Let β_t denote the parameters of the belief distribution at time t, referred to as the *sub-belief*. The belief itself is obtained by decoding β_t using the MAF:

$$\bar{b}_t = \mathbb{M}_\iota(\beta_t), \tag{5.9}$$

where $\mathbb{M}_\iota$ denotes the masked normalizing flow. We update the sub-belief recursively using a sub-belief encoder Φ^{sub}, implemented as a feed-forward network:

$$\beta_{t+1} = \Phi^{\text{sub}}(\beta_t, a_t, o_{t+1}). \tag{5.10}$$

The full belief update function is then defined as:

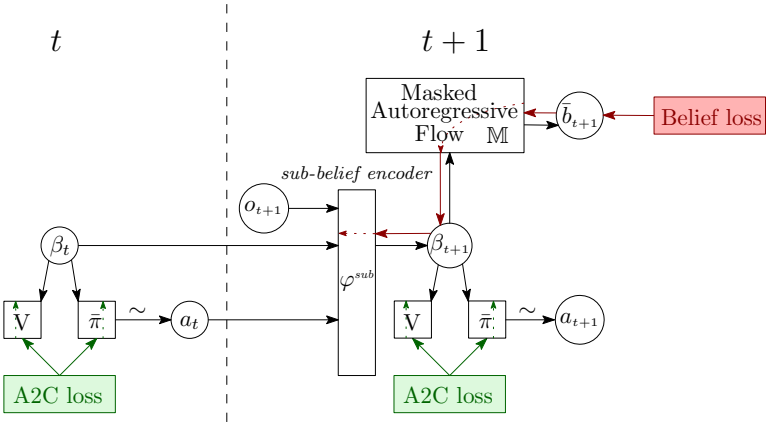
$$\Phi(\bar{b}_t, a_t, o_{t+1}) = \mathbb{M}_\iota \circ \Phi^{\text{sub}}(\beta_t, a_t, o_{t+1}). \tag{5.11}$$

This recursive structure is reminiscent of recurrent neural networks (RNNs), which also maintain a hidden state across time. However, a fundamental distinction lies in how these states are learned. In RNNs, the hidden state is trained via *back-propagation through time* (BPTT), allowing the network to use future gradients to refine earlier representations. In contrast, we *do not use BPTT* to train Φ^{sub}, a decision that stems from the conceptual difference between sub-beliefs and RNN hidden states.

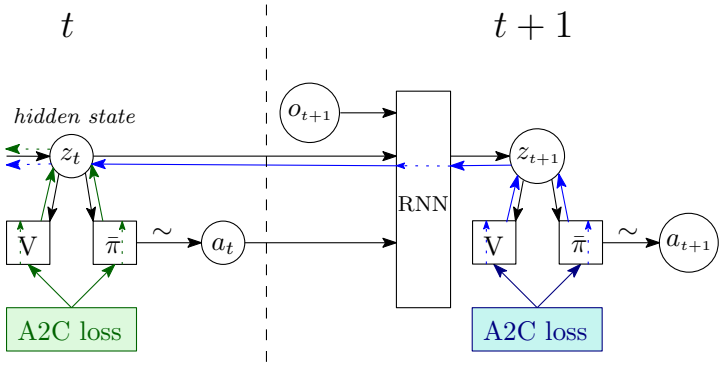
5.4.2 Why Not BPTT? Revisiting Belief Learning vs. History Compression

Figure 5.2 contrasts the WBU belief updater with a standard RNN used in R-A2C. Although both structures propagate internal states over time, they serve different purposes:

- **In RNNs (R-A2C)**, the hidden state is optimized to compress the full history into a representation useful for policy and value prediction. Since value targets are more accurate at later time-steps, BPTT propagates informative gradients backward to shape earlier hidden states. This *history compression for control* justifies BPTT.
- **In WBU**, the sub-belief is trained to follow the latent belief update rule. The belief $\bar{b}_t$ serves as a sufficient statistic for control, and is updated using the fixed latent transition and observation models learned by the WAE-MDP. Importantly, the belief update rule is a forward recurrence: it is an instance of a Markovian transition function in the belief MDP $\mathcal{M}_B$. Thus, belief learning corresponds to learning the transition dynamics of a latent MDP, for which BPTT is neither necessary nor desirable.



(a) **WBU** Beliefs are updated via a feed-forward sub-belief encoder trained only through the belief loss. The policy receives the sub-belief as input, but its gradients do not affect the belief model.



(b) **R-A2C** RNN hidden states compress the history and are trained jointly with the policy via BPTT, using future gradients to shape early representations.

Figure 5.2: Comparison of WBU and R-A2C architectures.

In fact, enabling BPTT in this setting would introduce two issues:

1. *Error accumulation and instability:* Beliefs at early time-steps are easier to infer, as they depend on shorter histories. Propagating gradients from later (and potentially noisier) beliefs can degrade rather than improve the update. Inaccurate future beliefs introduce spurious gradients that pollute earlier sub-beliefs.

2. *Violation of modularity:* Since Φ aims to replicate a Markovian update rule, its errors can be directly characterized by Theorem 5.1. Introducing BPTT entangles the belief updater with task-specific optimization and removes the clean separation between belief inference and control.

Thus, by using a feed-forward network for Φ^{sub} and avoiding BPTT, we align with the broader literature on model-based RL, where transition functions are learned with one-step recursions [Gelada et al., 2019; François-Lavet et al., 2019]. This design simplifies training, improves stability, and better supports theoretical guarantees.

5.4.3 Belief Loss: Approximating the Latent Update

The goal of training the belief updater is to learn a belief encoder $\Phi = \mathbb{M} \circ \Phi^{\text{sub}}$ that closely approximates the latent belief update $\overline{\mathcal{T}}$ of the POMDP model learned by the WAE-MDP. More precisely, we aim to minimize the divergence between the two distributions:

$$D\big(\Phi(\cdot \mid \bar{b}_t, a_t, o_{t+1}), \overline{\mathcal{T}}(\cdot \mid \bar{b}_t, a_t, o_{t+1})\big). \tag{5.12}$$

Ideally, we would use the Wasserstein distance $\mathcal{W}_{\bar{d}}$, as this aligns directly with the guarantees presented in Theorems 5.1 and 5.2. However, minimizing the Wasserstein distance is known to be challenging in practice: it often requires a dual formulation with additional networks, Lipschitz constraints, and a min-max optimization procedure [Arjovsky et al., 2017b], similar to how WAE-MDPs themselves are trained.

Moreover, computing the Wasserstein distance requires sampling from both distributions. While sampling from the belief encoder is straightforward, sampling from the latent belief update rule $\overline{\mathcal{T}}$ (Eq. 5.1) is non-trivial due to its nested expectations.

KL divergence as a surrogate. As an alternative, we minimize the Kullback-Leibler divergence:

$$D_{\text{KL}}\big(\Phi \parallel \overline{\mathcal{T}}\big).$$

This divergence is easier to optimize, and only requires sampling from the belief encoder. Furthermore, in the zero-temperature limit of the WAE-MDP, the Wasserstein distance converges to the total variation distance, which in turn is upper-bounded by

the KL divergence via Pinsker's inequality (Borwein and Lewis [2005], Equation (2.1)). This connection ensures that minimizing the KL divergence still yields guarantees on downstream value estimation.

On-policy training. We train Φ using the same on-policy samples as the policy $\bar{\pi}$, though the gradients of the belief loss do not affect the policy or vice versa. This parallel training structure helps the belief model specialize to the encountered trajectory distribution without entangling its parameters with the policy optimization.

Loss expression. The belief loss at any time-step $t \geq 0$, given the latent belief $\bar{b}_t$, the executed action a_t, and the next observation o_{t+1}, is given by:

$$D_{\mathrm{KL}}\big(\Phi\big(\bar{b}_t, a_t, o_{t+1}\big) \parallel \overline{\mathcal{T}}\big(\bar{b}_t, a_t, o_{t+1}\big)\big) = \log\Big(\mathop{\mathbb{E}}_{\bar{s}\sim\bar{b}_t}\mathop{\mathbb{E}}_{\bar{s}'\sim\overline{\mathbf{P}}_\theta(\cdot\mid\bar{s},a_t)} \overline{\mathcal{O}}_\theta\big(o_{t+1}\mid\bar{s}'\big)\Big) + \mathop{\mathbb{E}}_{\bar{s}_{t+1}\sim\Phi\left(\bar{b}_t,a_t,o_{t+1}\right)}\left[\log\Phi\big(\bar{s}_{t+1}\mid\bar{b}_t,a_t,o_{t+1}\big) - \log\mathop{\mathbb{E}}_{\bar{s}\sim\bar{b}_t}\overline{\mathbf{P}}_\theta(\bar{s}_{t+1}\mid\bar{s},a_t) - \log\overline{\mathcal{O}}_\theta(o_{t+1}\mid\bar{s}_{t+1})\right] \tag{5.13}$$

This decomposition has a clear interpretation:

- The first term is a normalization constant for the target distribution, independent of the encoder.
- The second term is the (negative) entropy of the predicted belief.
- The third term is the conformity to the latent transition model.
- The fourth term filters latent states inconsistent with the observed o_{t+1}.

Normalizing term. Given the set of parameters ι of Φ, we minimize the KL divergence D_{KL} by gradient descent on the Monte-Carlo estimate of the divergence:

$$\nabla_\iota D_{\mathrm{KL}}\big(\Phi\big(\bar{b}_t, a_t, o_{t+1};\iota\big) \parallel \overline{\mathcal{T}}\big(\bar{b}_t, a_t, o_{t+1}\big)\big) =$$

$$\nabla_\iota \mathop{\mathbb{E}}_{\bar{s}_{t+1}\sim\Phi\left(\bar{b}_t,a_t,o_{t+1};\iota\right)}\left[\log\Phi\big(\bar{s}_{t+1}\mid\bar{b}_t,a_t,o_{t+1};\iota\big) - \log\mathop{\mathbb{E}}_{\bar{s}\sim\bar{b}_t}\overline{\mathbf{P}}_\theta(\bar{s}_{t+1}\mid\bar{s},a_t) - \log\overline{\mathcal{O}}_\theta(o_{t+1}\mid\bar{s}_{t+1})\right]$$

Notice that the first term of the divergence (the belief normalization term) of Eq. 5.13 does not depend on Φ and thus yields zero gradient. Nevertheless, we observed during our experiments that adding the normalizing term allows to stabilize and reduce the variance of the belief loss.

Remark 5.2: Smoothing the observation model

The WAE-MDP defines a deterministic decoder $\overline{\mathcal{O}}_\mu$, implying that $\overline{\mathcal{O}}$ is a Dirac distribution. However, using Dirac likelihoods in Eq. 5.13 leads to vanishing gradients and inefficient training, as most sampled states have negligible likelihood.

To address this, we model $\overline{\mathcal{O}}$ as a normal distribution:

$$\overline{\mathcal{O}}_\theta(\cdot \mid \bar{s}) = \mathcal{N}(\overline{\mathcal{O}}_\mu(\bar{s}), \sigma^2 I), \tag{5.14}$$

where σ^2 is fixed or annealed. This Gaussian relaxation improves learning stability and approximates the true Dirac in the low-variance limit.

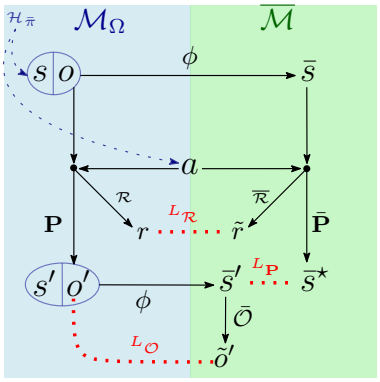
Optimizing $L^\Phi_{\overline{\mathbf{P}}}$ To optimize the Wasserstein term of $L^\Phi_{\overline{\mathbf{P}}}$ Eq. (5.3), we follow the same learning procedure than [Delgrange et al., 2023, Appenix A.5]: we introduce neural networks $\mathcal{F}_\spadesuit$ (for $\spadesuit \in \left\{\overline{\mathbf{P}}, \Omega\right\}$) that are trained to attain the supremum of the dual formulation of the Wasserstein distance (Section 2.1.3). To do so, we need enforce the Lipschitzness of $\mathcal{F}_\spadesuit$ and, as in WAE-MDPs [Delgrange et al., 2023], we do so via the gradient penalty approach of Gulrajani et al. [2017], leveraging that any differentiable function is 1-Lipschitz if and only if it has gradients with norm at most 1 everywhere. Finally, notice that we do not directly optimize the total variation distance of $L_\mathcal{O}$, but rather the Wasserstein; we take the usual Euclidean distance as metric over Ω which is proven to be Lipschitz equivalent to a distance converging to the discrete metric as the temperature of the WAE-MDP goes to zero [Delgrange et al., 2023, Appendix A.6] to finally recover d_{TV}.

5.4.4 Learning Procedure and Optimization Flows

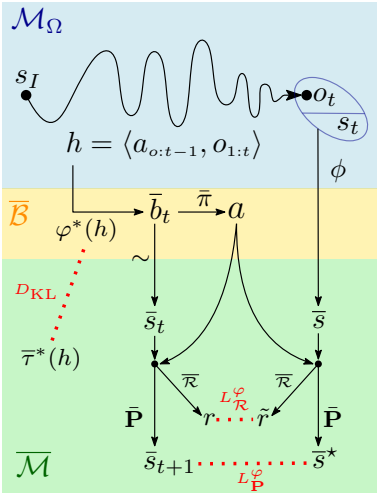
The guarantees established in Theorems 5.1 and 5.2 rely on minimizing a collection of loss functions that capture discrepancies between the learned model and the true POMDP dynamics. These losses fall into two categories: *local losses*, which concern the latent reward, transition, and observation models, and *belief losses*, which govern the belief update mechanism and its compatibility with the dynamics.

Figure 5.3 illustrates the optimization flows used to compute these losses. It highlights the relationship between the original state-observation space (blue), the learned latent space (green), and the belief space (yellow), as well as the key discrepancies (red) that drive learning.

Local losses. The flow corresponding to the local losses is shown in Fig. 5.3a. At each training step, a sample (s, o, a) is drawn from the on-policy distribution induced by $\overline{\pi}$. The pair (s, o) is mapped into the latent space via the WAE-MDP encoder $\phi(s, o) = \bar{s}$.



(a) Optimization of the latent space model parameters (i.e., $\overline{\mathcal{R}}, \overline{\mathbf{P}}$, and $\overline{\mathcal{O}}$) by minimizing local losses.



(b) Optimization of the belief encoder Φ by minimizing the proxy belief loss, as well as transition and reward regularizers.

Figure 5.3: Latent flows used to compute the local and belief losses. Arrows represent (stochastic) mappings. The original state-observation space is shown in blue, the latent state space in green, and the belief space in yellow. Discrepancies minimized during training are depicted in red. The blue region corresponds to the state-observation space $\mathcal{S}_\Omega$, which is accessible during training.

The action a is then executed in both the original environment and the latent model, producing the true outcome (s', o', r) and the model's prediction $(\tilde{s}', \tilde{o}', \tilde{r})$. These are compared to define three local losses: the reward loss $L_{\mathcal{R}}$, the transition loss L_{P} over latent states, and the observation loss $L_{\mathcal{O}}$ between o' and samples from $\overline{\mathcal{O}}(\cdot \mid \tilde{s}')$. In practice, $\overline{\mathcal{O}}$ may be stochastic (e.g., Gaussian), as discussed in Remark 5.2.

Belief losses. Figure 5.3b illustrates the computation of belief losses. Training proceeds on-policy: at each time step t, the agent receives an observation o_t of the state s_t, with full state-observation access during training.

First, the current history h_t is encoded into a latent belief $\bar{b}_t = \Phi(h_t)$. This belief is compared to the Bayesian posterior $\bar{b}' = \overline{\mathcal{T}}(h_t)$ to compute a divergence-based belief loss $L_{\overline{\mathcal{T}}}$.

Next, transition and reward regularizers are computed by comparing the outcome of executing an action $a \sim \bar{\pi}\big(\cdot \mid \bar{b}_t\big)$ on samples from $\bar{b}_t$ to that of embedding the actual state-observation pair (s_t, o_t) and transitioning under the latent model. This alignment enforces consistency between the belief and the model predictions.

Together, these optimization flows ensure that the learned latent representation accurately reflects the dynamics and uncertainty of the original environment, while producing a belief space suitable for control. This procedure directly implements the sufficient conditions required for the theoretical guarantees presented earlier, and forms the backbone of the WBU training process.

5.4.5 Policy Learning with Sub-beliefs

The policy $\bar{\pi}$ receives the sub-belief β_t as input and is optimized independently via an on-policy algorithm. Crucially, we block gradients from the RL objective from flowing through Φ, ensuring a clean separation between belief modeling and control. While A2C is used in our experiments, any on-policy method can be employed. The policy thus conditions on a sufficient statistic of the history for control.

5.5 Experiments

We evaluate the Wasserstein Belief Updater on a range of environments designed to probe different aspects of partial observability. Our goal is to assess the effectiveness of WBU in learning belief representations that are useful for control and exhibit robustness across POMDPs requiring either memory, inference, or resilience to noise.

Algorithm 6: WASSERSTEIN BELIEF UPDATER

Input: Batch sizes B_{WBU}, B_{WAE}, B_{NEXT}; global learning steps N; no. of model updates per iteration N_{MODEL}; your favorite collect strategy π_{init}; replay buffer (RB) $\mathcal{D}$; Lipschitz networks $\mathcal{F}_{\overline{\mathrm{P}}} : \overline{\mathcal{S}} \to \mathbb{R}$, $\mathcal{F}_{\Omega} : \Omega \to \mathbb{R}$; observation variance network $\overline{\mathcal{O}}_{\sigma}$; and loss weights $w_{\overline{\mathcal{R}}}$, $w_{\overline{\mathrm{P}}}$

collect $\theta = \left\{ s_i, o_i, a_i, r_i, s'_i, o'_i \right\}_{i=1}^{N_{\text{init}}}$ by executing π_{init} for N_{init} steps; **store** θ in $\mathcal{D}$

▷ *Use the exploration policy π_{init} to collect transitions and initialize the RB*

repeat N **times**

  Update WAE-MDP model with Algorithm 7

  for $i \leftarrow 1$ *to* B_{WBU}

    $s_0 \leftarrow s_I$; $\bar{s}_0 \leftarrow \bar{s}_I$; $\overline{b}_0 \leftarrow \delta_{\bar{s}_0}$; $\beta_0 \leftarrow \beta_I$ ▷ *β_I is arbitrary, e.g., zeroes*

    for $t \leftarrow 0$ *to* T

      $a_t \sim \overline{\pi}(\cdot \mid \beta_t)$ ▷ *Produce the action a_t according to the sub-belief β_t*

      $\langle s_{t+1}, o_{t+1} \rangle, r_t \leftarrow \text{env.step}(a_t)$ ▷ *Execute the action in the environment*

      store the transition $\langle s_t, o_t, a_t, r_t, s_{t+1}, o_{t+1} \rangle$ into $\mathcal{D}$

      $\beta_{t+1} \leftarrow \Phi^{sub}(\text{sg}(\beta_t), a_t, o_{t+1})$ ▷ *Update the sub-belief with stop gradients*

      $\overline{b}_{t+1} \leftarrow \mathbb{M}_i(\beta_{t+1})$ ▷ *Retrieve the belief distribution b_{t+1} via the MAF $\mathbb{M}$*

      $\bar{s}_{t+1} \sim \overline{b}_{t+1}$ ▷ ***Believe** the next latent state*

      ▷ *Marginalize the next latent state distribution w.r.t. the current belief*

      for $j \leftarrow 1$ *to* B_{NEXT}

        $\bar{s} \sim \overline{b}_t$; $\mathcal{L}^j_{\log \overline{\mathrm{P}}} \leftarrow \left[\log \overline{\mathrm{P}}(\bar{s}_{t+1} \mid \bar{s}, a_t) - \log B_{\text{NEXT}} \right]$

      $\mathcal{L}^{i,t}_{\text{KL}} \leftarrow \log \overline{b}_{t+1}(\bar{s}_{t+1}) - \text{LSE}(\left\{ \mathcal{L}^j_{\log \overline{\mathrm{P}}} \right\}_{j=1}^{B_{\text{NEXT}}}) - \log \overline{\mathcal{O}}(o_{t+1} \mid \bar{s}_{t+1})$

      ▷ *Pointwise decomposition of Eq. 5.13: divergence with the belief update rule*

      $\mathcal{L}^{i,t}_{\overline{\mathcal{R}}} \leftarrow \left| \overline{\mathcal{R}}(\phi(s_t, o_t), a_t) - \overline{\mathcal{R}}(\bar{s}_t, a_t) \right|$ ▷ *Latent reward regularizer*

      $\bar{s}' \sim \overline{\mathrm{P}}(\cdot \mid \bar{s}_t, a_t)$

      ▷ *Transition to the next latent state from the current believed latent state*

      $\mathcal{L}^{i,t}_{\overline{\mathrm{P}}} \leftarrow [\mathcal{F}_{\overline{\mathrm{P}}}(\phi(s_{t+1}, o_{t+1})) - \mathcal{F}_{\overline{\mathrm{P}}}(\bar{s}')]$ ▷ *Latent transition regularizer*

  Update $\mathcal{F}_{\overline{\mathrm{P}}}$ by maximizing $\sum_{i=1}^{B_{\text{WBU}}} \sum_{t=0}^{T-1} \mathcal{L}^{i,t}_{\overline{\mathrm{P}}}$ **and** enforcing its 1-Lipschitzness w.r.t. latent metric $\overline{d}$

  Update Φ^{sub} and $\mathbb{M}$ by minimizing

  $1/(B_{\text{WBU}}+T) \sum_{i=1}^{B_{\text{WBU}}} \sum_{t=0}^{T-1} (\mathcal{L}^{i,t}_{\text{KL}} + w_{\overline{\mathcal{R}}} \cdot \mathcal{L}^{i,t}_{\overline{\mathcal{R}}} + w_{\overline{\mathrm{P}}} \cdot \mathcal{L}^{i,t}_{\overline{\mathrm{P}}})$

  Update $\overline{\pi}$ by minimizing the A2C loss on the batch $\left\{ \beta^i_{0:T}, a^i_{0:T-1}, r^i_{0:T-1} \right\}_{i=1}^{B_{\text{WBU}}}$

Algorithm 7: WAE-MDP Model Update

Input: Batch sizes B_{WAE}; no. of model updates per iteration N_{Model}; replay buffer (RB) $\mathcal{D}$; Lipschitz networks $\mathcal{F}_{\overline{\mathbb{P}}} : \overline{S} \to \mathbb{R}$, $\mathcal{F}_{\Omega} : \Omega \to \mathbb{R}$; observation variance network $\overline{\mathcal{O}}_{\sigma}$; and loss weights $w_{\overline{R}}$, $w_{\overline{\mathbb{P}}}$

repeat N_{Model} **times**

▷ *Update the WAE-MDP model for N_{Model} consecutive training steps*

for $i \leftarrow 1$ *to* B_{WAE}

$\langle s_i, o_i, a_i, r_i, s_i', o_i' \rangle \sim \mathcal{D}$ ▷ *Sample a transition from the RB*

$\bar{s}' \leftarrow \phi_{\iota}\left(s_i', o_i'\right)$ ▷ *Embed $\langle s_i', o_i' \rangle$ to the latent space*

$\tilde{o}_i' \sim \overline{\mathcal{O}}_{\theta}(\cdot \mid \bar{s}')$ ▷ *Observe the resulting latent state via $\overline{\mathcal{O}}$*

$\mathcal{L}_{\text{WAE}} \leftarrow$ **compute the WAE-MDP loss** on transition batch $\left\{ s_i, o_i, a_i, r_i, s_i', o_i' \right\}_{i=1}^{B_{\text{WAE}}}$

Update the WAE-MDP components $\mathcal{L}_{\mathcal{O}} \leftarrow {}^{1}/_{B_{\text{WAE}}} \cdot \sum_{i=1}^{B_{\text{WAE}}} \left[\mathcal{F}_{\Omega}\left(o_i'\right) - \mathcal{F}_{\Omega}\left(\tilde{o}_i'\right) \right]$

▷ *Observation loss*

Update $\mathcal{F}_{\Omega}$ by maximizing $\mathcal{L}_{\mathcal{O}}$ **and** enforcing its 1-Lipschizness w.r.t. metric d_{Ω}

Update $\overline{\mathcal{O}}_{\sigma}$ by minimizing $\mathcal{L}_{\mathcal{O}}$

We compare WBU to two baselines:

- **R-A2C** (Section 2.3.3), the recurrent variant of A2C using a GRU as history encoder, trained via back-propagation through time (BPTT),
- **DVRL** [Igl et al., 2018], which combines R-A2C with a variational auto-encoder and particle filtering to approximate belief distributions.

We refer to Section 2.3.3 for details on these baselines. WBU is the only method with theoretical guarantees on representation quality (Theorem 5.2) and separates the learning of beliefs from policy optimization. Algorithm 6 provides the pseudocode of WBU.

5.5.1 Environment Design and Evaluation Criteria

POMDPs vary widely in the nature of their partial observability. To provide a principled evaluation, we identify three distinct categories of POMDPs:

1. **Long-term memory** tasks, where optimal behavior depends on recalling information far in the past.
2. **Hidden state features**, where the observation lacks sufficient instantaneous information to determine the state.

3. **Noisy observations**, where sensory noise must be filtered over time to recover meaningful signals.

We use environments from the POPGym benchmark [Morad et al., 2023] as well as a modified version of MinAtar [Young and Tian, 2019]. We emphasize that, unlike modified Atari benchmarks where stacking four frames is often sufficient to reconstruct the state, these environments are explicitly designed to test generalization across different types of partial observability.

5.5.2 Results and Analysis

Figure 5.4 presents learning curves for each algorithm, including the evolution of the belief loss in WBU. All experiments are run over 5 seeds. The environments and results are grouped according to the type of partial observability they target.

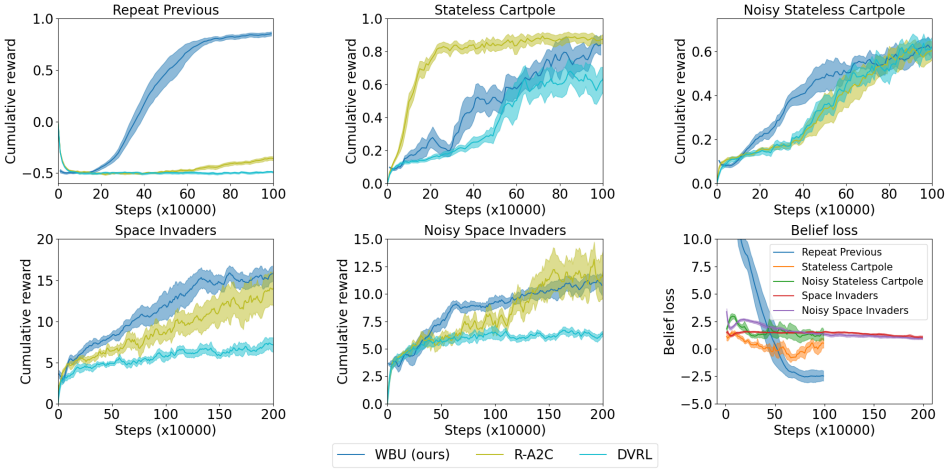


Figure 5.4: Evolution of (i) undiscounted cumulative return for WBU, R-A2C, and DVRL, and (ii) the estimated belief loss during training for WBU. Mean and standard error are reported across 5 seeds.

Long-term memory. In the RepeatPrevious environment, the agent observes one card per step and must predict the suit of the card shown 8 time steps earlier. WBU clearly outperforms all baselines, reaching positive returns rapidly and approaching perfect memorization by the end of training. R-A2C exhibits delayed and limited improvement, while DVRL fails to learn.

This environment demonstrates WBU's ability to maintain sufficient statistics of the past. While DVRL attempts to learn a belief distribution, it lacks the theoretical structure that allows WBU to maintain stable, informative beliefs over long time horizons.

Hidden features. In StatelessCartPole, velocity components are hidden. R-A2C performs best early on, leveraging its short-term memory to infer velocity from positional changes. However, WBU gradually closes the gap and matches its final performance, outperforming DVRL throughout training.

We also evaluate on SpaceInvaders, where the direction of alien motion and distinction between friendly and enemy fire are hidden. In this more complex setting, WBU consistently outperforms both baselines, showing that its learned latent beliefs are expressive and robust to structured ambiguity in the observations.

Noisy observations. To test noise resilience, we introduce two types of observation corruption:

- **Gaussian noise** on the positions in StatelessCartPole.
- **Binary masking noise** in SpaceInvaders, where each alien's position may be obscured independently at each step.

In both cases, WBU proves robust. It learns faster than DVRL and R-A2C, and converges to better policies. Unlike R-A2C, which suffers from accumulating noise in its hidden state, WBU's latent belief provides a denoised summary of the state. While DVRL's particles help mitigate noise to some extent, its performance remains below WBU and is less stable.

5.5.3 Representation Quality and Belief Dynamics

A key contribution of WBU is that it learns a belief representation aligned with value estimation. Theorem 5.2 establishes that the learned beliefs preserve value structure. To illustrate this, we project latent beliefs obtained from SpaceInvaders onto a 2D plane using t-SNE [Maaten and Hinton, 2008]. Figure 5.5 shows that nearby beliefs in this space indeed correspond to similar values, confirming the alignment between representation and control.

We also monitor the belief loss during training (Fig. 5.4). While belief estimation and model learning are conducted in parallel, the consistent decrease in the belief loss reflects the improvement of the learned latent model. Unlike baselines that entangle policy and representation updates, WBU optimizes its belief encoder independently from the policy, ensuring that improvements in belief quality directly support planning and learning.

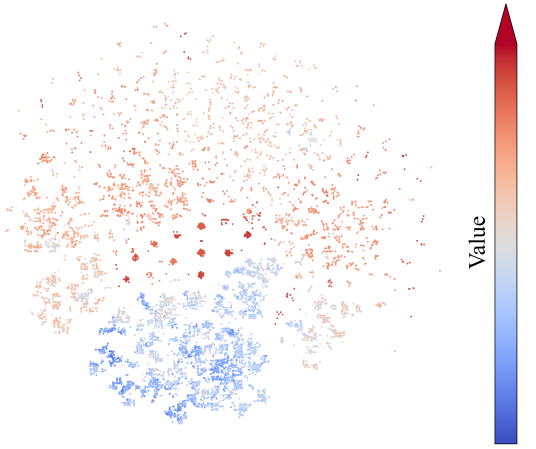


Figure 5.5: 2D t-SNE projection of latent beliefs learned by WBU in SpaceInvaders at the end of training. Beliefs with similar values are clustered.

Summary

These experiments confirm the flexibility and generalization capability of WBU. By learning a value-aligned, task-independent belief representation:

- WBU effectively solves tasks requiring memory or inference,
- remains robust to observation noise,
- and separates the learning of representation from policy optimization.

Taken together, these results validate WBU as a principled and effective approach to representation learning in partially observable settings.

5.6 Conclusion

This chapter introduced the Wasserstein Belief Updater (WBU), a model-based reinforcement learning algorithm that learns a belief representation and belief update rule by leveraging access to the state during training. Unlike methods that rely on recurrent architectures trained end-to-end through the RL objective to implicitly extract sufficient information from the observation history, WBU explicitly targets the belief dynamics of the underlying POMDP.

By minimizing the divergence between the learned update and the true Bayesian belief update of the latent POMDP, WBU constructs a latent representation that supports

control and satisfies provable guarantees. In particular, we showed that the learned belief representation is sufficient to condition the optimal value function.

This chapter contributes to the broader goal of the thesis: demonstrating how access to the ground-truth state during training can be used to learn structured, value-aligned representations that improve generalization and robustness in partially observable settings. WBU exemplifies this approach by disentangling the representation learning process from policy optimization, and by providing both theoretical and empirical support for its effectiveness.

5.6.1 Limitations and Scope

The Wasserstein Belief Updater is designed as a principled approach to belief learning under partial observability, but this emphasis also constrains the settings in which it is most effective.

First, belief representations are not necessarily the most efficient representations for control. Although beliefs form a sufficient statistic in POMDPs, they may encode substantially more information than is required to solve a given task. By explicitly targeting belief learning, WBU prioritizes representational sufficiency and generality over minimality. This can slow down learning compared to approaches that directly optimize compact latent states tailored to the control objective.

Second, WBU relies on multiple interacting learned components, including the latent dynamics model, the belief updater, and the value function. These components are trained in parallel and depend on one another for stable learning signals. In practice, this increases sensitivity to hyperparameters and training schedules, and can lead to instability, particularly in early training phases when model inaccuracies propagate across components.

Finally, the empirical evaluation presented in this chapter could be extended. A more extensive comparison against more recent POMDP algorithms and belief based or latent state baselines, as well as evaluations across a broader range of partially observable environments, would provide a more complete picture of the practical tradeoffs involved. At the same time, the primary contribution of WBU lies not in exhaustive empirical dominance, but in the principled formulation of belief learning with explicit state access and the associated theoretical guarantees. Further empirical exploration is therefore complementary to, rather than a prerequisite for, the validity of the approach.

6

Conclusion and Future Work

Revisiting the Problem and Motivation

Sequential decision-making under partial observability lies at the heart of many real-world challenges, from autonomous robots navigating uncertain environments to multiagent systems coordinating under limited communication. In these settings, agents rarely have direct access to the full environment state, and instead must act based on incomplete and noisy observations. Although the underlying dynamics remain Markovian, the agent's observation stream breaks the Markov property, making inference, memory, representation learning, and careful exploration essential for effective decision-making.

This thesis was motivated by the observation that in many practical and simulated domains, privileged access to the environment state is available in some form. Full state information is often accessible during training in simulators, and in some cases, agents can actively query more accurate information at runtime at a cost through additional sensors, communication, or computation. Rather than treating partial observability as a fixed limitation, this perspective reframes *state access as a resource* that can be leveraged strategically to stabilize learning, guide internal representation, and improve planning efficiency.

The central question explored in this thesis is: *How and when can agents strategically use access to the true state of the environment, whether during training or at execution*

time under a cost, to improve performance in partially observable sequential decision making problems? We investigated this question across three complementary settings: (i) multiagent reinforcement learning with centralized training, (ii) single-agent planning with explicit decisions over costly state access, and (iii) model-based reinforcement learning that uses state access during training to learn approximate belief updates. Together, these contributions establish a cohesive research agenda for designing agents that reason explicitly about uncertainty and strategically leverage access to the true state as a resource to overcome the challenges of partial observability.

Summary of Contributions

Local Advantage Networks

Chapter 3 introduced *Local Advantage Networks* (LAN), a scalable value-based algorithm for cooperative multi-agent reinforcement learning in Dec-POMDPs. LAN leverages access to the full environment state during training to construct a centralized value function, which is combined with local advantage functions defined over individual observation-action histories. This decomposition yields a proxy Q-value for each agent that stabilizes learning and mitigates non-stationarity, while preserving decentralized execution.

Unlike value factorization approaches such as QMIX and QPLEX, which focus on designing expressive decompositions of the joint value function, LAN takes a different route: it uses the centralized value to provide shared training targets across agents. This leads to more stable optimization, effective credit assignment, and scalability, as the centralized component's parameterization is independent of the number of agents. LAN achieves strong empirical performance, outperforming or matching state-of-the-art value based baselines on challenging SMAC scenarios, including those requiring temporally extended asymmetric cooperation, and demonstrating robust generalization on a modified MPE benchmark.

LAN highlights the benefits of strategically using state access during training to simplify optimization rather than complicate factorization. However, like other deep Q-learning approaches, it inherits sensitivity to hyperparameters and replay buffer design. These trade-offs frame LAN as a principled yet practical alternative for scaling multi-agent learning under partial observability.

AEMS-SR

Chapter 4 addressed decision-making in environments where agents can request the true state at a cost during execution. We formalized this setting as a *POMDP with State*

Requests (POMDP-SR) and introduced *AEMS-SR*, a planning algorithm that efficiently searches this augmented decision space. Classical tree-based POMDP solvers such as AEMS and POMCP become intractable in this setting due to redundant belief nodes arising from repeated state queries. AEMS-SR overcomes this limitation by replacing tree-based search with a cyclic belief graph, enabling efficient reuse of beliefs and avoiding exponential blowup.

We proved that AEMS-SR is complete and ϵ-optimal, and demonstrated its superiority over existing algorithms on the Tag domain and RobotDelivery, a novel benchmark explicitly designed to stress test POMDP-SR planning. This work contributes a novel graph-based search algorithm tailored to POMDP-SR, addressing computational bottlenecks in classical tree-based solvers.

While AEMS-SR scales well to moderate problem sizes, it remains computationally expensive for very large domains and relies on exact belief updates. These limitations motivate future extensions that integrate learning-based approximations or hierarchical planning to make POMDP-SR tractable in real-world systems.

Wasserstein Belief Updater

Chapter 5 introduced the *Wasserstein Belief Updater* (WBU), a model-based reinforcement learning framework for learning compact, value-aligned belief representations. Unlike methods that rely on recurrent architectures trained end-to-end through the RL objective, WBU explicitly learns a latent representation of the environment's belief dynamics by minimizing the divergence between the learned belief update and the true Bayesian update in a latent POMDP. By accessing the ground-truth state during training, WBU grounds its representations in the underlying dynamics rather than observation sequences alone.

We proved that the learned belief representation is sufficient to condition the optimal value function, establishing a theoretical connection between approximate belief modeling and control performance. Empirically, WBU outperforms recurrent baselines across multiple partially observable environments, demonstrating improved stability and generalization.

WBU exemplifies the central theme of this thesis: using access to the state during training not only to improve performance but also to derive theoretical guarantees. While this approach assumes access to the environment state and relies on reconstruction-based objectives that may distort bisimulation, it represents a step toward principled, interpretable representations for decision-making under uncertainty.

Synthesis

Each of these three contributions advance a research agenda for reinforcement learning and planning under partial observability. LAN demonstrates that access to the state during training can simplify optimization and stabilize multi-agent learning; AEMS-SR shows that reasoning explicitly about when to acquire state information can lead to principled and efficient planning algorithms; and WBU provides a theoretical foundation for learning value-aligned representations by supervising belief modeling with state access. Across these settings this thesis reframes state access as a powerful resource rather than an unrealistic assumption, offering a path toward scalable, robust decision-making in complex environments.

Future Work

The contributions of this thesis open several avenues for future research. Below, we outline several promising directions, both as extensions of the proposed methods and as broader developments of the themes explored in this work.

Improving world models

In Chapter 5, we relied on Wasserstein Auto-Encoded MDPs (WAE-MDPs) to learn a latent model of the environment that supports belief approximation. While this approach makes belief learning tractable, it imposes a reconstruction objective that forces the model to decode latent states back to the original observation space. This constraint can be detrimental when the goal is to learn compact or abstract representations that preserve functional properties such as bisimulation. As illustrated in Figure 6.1, reconstruction-based losses can collapse bisimilar states into distorted representations that fail to reflect their values.

To address this, we investigate a more flexible alternative based on Optimal Transport (OT), introduced in our recent workshop paper [Röpke et al., 2025]. The key idea is to eliminate reconstruction entirely and instead define a cost function that aligns real and latent transitions under criteria tailored to decision making. These cost functions can encode structure-preserving notions such as value equivalence or transition similarity, guiding the model to focus on task-relevant abstractions. This formulation generalizes the WAE-MDP framework while removing its architectural and training constraints. Although the approach is still preliminary, we believe it offers a more principled foundation for learning latent world models.

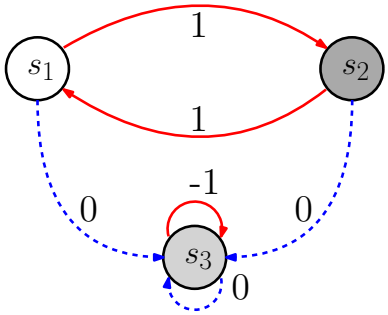


Figure 6.1: Illustration of the failure of reconstruction-based losses to preserve bisimulation. In this MDP, states s_1 and s_2 are bisimilar, but a decoder trained to minimize reconstruction error may map them both to an ambiguous mean state s_3 that differs in value. This leads to a representation that is compact but not useful for decision-making.

In future work, we aim to adapt these OT-based world models to partially observable settings, replacing the WAE-based models in Chapter 5 and enabling more scalable and accurate belief approximation.

Beyond pure representation learning, this framework creates opportunities for integrating planning into the learning process. Rather than using the model solely for embedding histories or supporting belief updates, planning can be used to guide and regularize learning directly. For example, rollouts from the model can provide planning-based value targets, improving sample efficiency and acting as an additional source of structured supervision.

Learning when to request the state

In Chapter 4, we studied the problem of agents that can request access to the true state at a cost during execution. This setting is not only theoretically interesting, but also closely mirrors real-world scenarios where different sensors offer varying levels of reliability, latency, or energy consumption.

Our planner, AEMS-SR, provides an efficient solution to this problem in small domains by explicitly modeling the cost of accessing the state during online planning. However, like most heuristic search planners, it does not scale to large environments. This opens the door to a learning-based alternative: training agents to learn when to request the state based on experience.

One promising direction is to combine this setting with the latent belief modeling approach developed in Chapter 5. By extending WBU to incorporate a state-access decision at each step, one could train an agent to maintain a latent belief while also

learning to request the state only when needed. Planning, for instance via AEMS-SR in latent space, could be used during training to guide the learning process by generating cost-aware state-access strategies.

Such a hybrid method would make it possible to scale the ideas developed in Chapter 4 to more complex environments while preserving a principled treatment of partial observability and uncertainty reduction.

Multiagent belief modeling

In Chapter 5, we showed that it is possible to learn a latent approximation of the belief update function for single-agent partially observable environments. Extending this idea to the multiagent setting is a natural yet challenging direction. As discussed in Chapter 3, one of the central difficulties of multiagent reinforcement learning (MARL) is the inherent non-stationarity introduced by the presence of other learning agents. This makes the learning problem more unstable and complicates the interpretation of observations, since the dynamics of the environment now include the evolving behaviors of other agents.

In a partially observable multiagent setting, each agent's belief must capture not only uncertainty over the environment's state, but also over the policies, observations, or internal histories of the other agents. As formalized in Definition 3.1, the induced local POMDP for each agent includes both environmental dynamics and the behaviors of others. This creates a form of nested uncertainty that is difficult to handle.

A promising research direction is to develop methods for learning latent belief models in MARL, where each agent maintains a compact belief over its own induced POMDP. This could build on the techniques introduced in Chapter 5, where access to the state during training was used to learn a belief update function. In the multiagent case, state access could include not only the environment's state but also the full joint configuration of other agents under the Centralized Training Decentralized Execution paradigm, enabling principled training of decentralized belief models. Such models could provide a foundation for more stable learning, better credit assignment, and improved coordination in complex multiagent tasks under partial observability, especially when theoretical guarantees are required to deploy such models into the real-world [Delgrange, 2024].

Multiagent communication

In many real-world scenarios, it is natural for collaborative agents to communicate in order to improve coordination and performance. When each agent has only partial observability, sharing information becomes a powerful way to reduce uncertainty and make better collective decisions.

Most existing methods rely on always-on communication, where agents constantly exchange fixed-size messages. This is often impractical in settings with bandwidth, energy, or attention constraints. A more realistic and efficient alternative is to allow agents to communicate only when needed, sending or requesting information selectively based on its utility.

For instance, our prior work on Dynamic Size Message Scheduling (DSMS) [Sun et al., 2024] shows that even simple scheduling mechanisms that adapt message sizes can lead to improved performance in bandwidth-limited settings. While DSMS does not learn when to communicate, it illustrates the benefit of moving away from rigid always-on protocols.

A promising direction is to view communication through the same lens as costly state access, as explored in Chapter 4. Instead of querying an external sensor, an agent could choose to request specific information from another agent: effectively treating peer communication as a form of selective sensing similar to Melo and Veloso [2009]. Combining this with the latent belief modeling framework of Chapter 5 could enable agents to communicate belief updates or request information that helps refine their internal models, leading to more informed and coordinated behavior under partial observability.

Final Remarks

In summary, this thesis demonstrates that strategically leveraging access to the true state is a powerful approach for tackling partial observability, enabling algorithms that are both principled and empirically effective. By treating state access as a resource, this work reframes partial observability as a continuum to exploit, not a rigid constraint, advancing our understanding of decision-making under uncertainty.

Bibliography

Arjovsky, M., S. Chintala, and L. Bottou
2017a. Wasserstein generative adversarial networks. In *Proceedings of the 34th International Conference on Machine Learning*, D. Precup and Y. W. Teh, eds., volume 70 of *Proceedings of Machine Learning Research*, Pp. 214–223. PMLR. 25

Arjovsky, M., S. Chintala, and L. Bottou
2017b. Wasserstein Generative Adversarial Networks. In *Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017*, D. Precup and Y. W. Teh, eds., volume 70 of *Proceedings of Machine Learning Research*, Pp. 214–223. PMLR. 142

Auer, P., N. Cesa-Bianchi, and P. Fischer
2002. Finite-time analysis of the multiarmed bandit problem. *Machine learning*, 47:235–256. 31

Avalos, R., E. Bargiacchi, A. Nowe, D. Roijers, and F. A. Oliehoek
2025. Online planning in POMDPs with state-requests. *Reinforcement Learning Journal*, 1:108–129. 85

Avalos, R., F. Delgrange, A. Nowe, G. Perez, and D. M. Roijers
2024. The Wasserstein Believer: Learning Belief Updates for Partially Observable Environments through Reliable Latent Space Models. 119

Avalos, R., M. Reymond, A. Nowe, and D. M. Roijers
2023. Local Advantage Networks for Multi-Agent Reinforcement Learning in Dec-POMDPs. *Transactions on Machine Learning Research*. 45

Avalos, R., M. Reymond, A. Nowé, and D. M. Roijers
2022. Local Advantage Networks for Cooperative Multi-Agent Reinforcement Learning. In *AAMAS '22: Proceedings of the 21st International Conference on Autonomous Agents and MultiAgent Systems (Extended Abstract)*. 45

Bargiacchi, E., D. M. Roijers, and A. Nowé
2020. Ai-toolbox: A c++ library for reinforcement learning and planning (with python bindings). *Journal of Machine Learning Research*, 21(102):1–12. 113

Bellinger, C., R. Coles, M. Crowley, and I. Tamblyn
2021. Active Measure Reinforcement Learning for Observation Cost Minimization. In *Canadian Conference on AI*. 89

Bellman, R.
1957. *Dynamic Programming*. Dover Publications. 26, 28, 29

Bernstein, D. S., R. Givan, N. Immerman, and S. Zilberstein
2002. The complexity of decentralized control of markov decision processes. *Mathematics of operations research*, 27(4):819–840. 40

Bertsekas, D. P.
2018. *Approximate Dynamic Programming*, Dynamic Programming and Optimal Control, 4th edition, updated second printing edition. Athena Scientific. 30

Borwein, J. and A. Lewis
2005. *Convex Analysis and Nonlinear Optimization: Theory and Examples*, CMS Books in Mathematics. Springer New York. 143

Cassandra, A., M. L. Littman, and N. L. Zhang
1997. Incremental pruning: a simple, fast, exact method for partially observable Markov decision processes. In *Proceedings of the Thirteenth conference on Uncertainty in artificial intelligence*, Pp. 54–61. 109

Castro, P. S., T. Kastner, P. Panangaden, and M. Rowland
2021. MICo: Improved representations via sampling-based state similarity for Markov decision processes. In *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, December 6-14, 2021, virtual*, M. Ranzato, A. Beygelzimer, Y. N. Dauphin, P. Liang, and J. W. Vaughan, eds., Pp. 30113–30126. 43

Çatal, O., S. Wauthier, C. De Boom, T. Verbelen, and B. Dhoedt
2020. Learning generative state space models for active inference. *Frontiers in Computational Neuroscience*, 14:574372. 124

Chatterjee, K., M. Chmelik, R. Gupta, and A. Kanodia
2016. Optimal cost almost-sure reachability in POMDPs. *Artif. Intell.*, 234:26–48. 126

Chen, X., Y. M. Mu, P. Luo, S. Li, and J. Chen
2022. Flow-based recurrent belief state learning for pomdps. In *International Conference on Machine Learning*, Pp. 3444–3468. PMLR. 124

Childs, B. E., J. H. Brodeur, and L. Kocsis
2008. Transpositions and move groups in Monte Carlo tree search. In *2008 IEEE Symposium On Computational Intelligence and Games*, Pp. 389–395. IEEE. 95

Cho, K., B. van Merrienboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio
2014. Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation. *EMNLP 2014 - 2014 Conference on Empirical Methods in Natural Language Processing, Proceedings of the Conference*, Pp. 1724–1734. Publisher: Association for Computational Linguistics (ACL). 38

Claus, C. and C. Boutilier
1998. Dynamics of reinforcement learning in cooperative multiagent systems. In *Proceedings of the National Conference on Artificial Intelligence*. 51

Coulom, R.
2007. Efficient selectivity and backup operators in Monte-Carlo tree search. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, volume 4630 LNCS. ISSN: 16113349. 29, 30

Csiszár, I. and J. Körner
2011. *Information theory: coding theorems for discrete memoryless systems*. Cambridge University Press. 24

de Witt, C. S., T. Gupta, D. Makoviichuk, V. Makoviychuk, P. H. S. Torr, M. Sun, and S. Whiteson
2020. Is Independent Learning All You Need in the StarCraft Multi-Agent Challenge? 52

Delgrange, F.
2024. *Activating formal verification of deep reinforcement learning policies by model checking bisimilar latent space models*. PhD thesis, Vrije Universiteit Brussel, Universiteit Antwerpen. 22, 158

Delgrange, F., A. Nowe, and G. Perez
2023. Wasserstein Auto-encoded MDPs: Formal Verification of Efficiently Distilled RL Policies with Many-sided Guarantees. In *International Conference on Learning Representations*. 43, 121, 139, 144

Delgrange, F., A. Nowé, and G. A. Pérez
2022. Distillation of RL Policies with Formal Guarantees via Variational Abstraction of Markov Decision Processes. *Proceedings of the AAAI Conference on Artificial Intelligence*, 36(6):6497–6505. 124

Desharnais, J., V. Gupta, R. Jagadeesan, and P. Panangaden
2004. Metrics for labelled Markov processes. *Theor. Comput. Sci.*, 318(3):323–354. 43

Dudley, R. M.
2018. *Real analysis and probability*. Chapman and Hall/CRC. 22, 24, 128

Elman, J. L.
1990. Finding structure in time. *Cognitive science*, 14(2):179–211. 38

Ferns, N., P. Panangaden, and D. Precup
2004. Metrics for finite markov decision processes. In *UAI*, volume 4, Pp. 162–169. 43

Ferns, N., P. Panangaden, and D. Precup
2011. Bisimulation Metrics for Continuous Markov Decision Processes. *SIAM J. Comput.*, 40(6):1662–1714. 43

Foerster, J., N. Nardell, G. Farquhar, T. Afouras, P. H. Torr, P. Kohli, and S. Whiteson
2017. Stabilising experience replay for deep multi-agent reinforcement learning. In *34th International Conference on Machine Learning, ICML 2017*, volume 3, Pp. 1879–1888. International Machine Learning Society (IMLS). 50, 53

Foerster, J. N., G. Farquhar, T. Afouras, N. Nardelli, and S. Whiteson
2018. Counterfactual multi-agent policy gradients. In *32nd AAAI Conference on Artificial Intelligence, AAAI 2018*, Pp. 2974–2982. 16, 40, 50, 64

François-Lavet, V., Y. Bengio, D. Precup, and J. Pineau
2019. Combined reinforcement learning via abstract representations. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, Pp. 3582–3589. Issue: 01. 142

Friston, K.
2010. The free-energy principle: a unified brain theory? *Nature reviews neuroscience*, 11(2):127–138. 124

Friston, K., J. Kilner, and L. Harrison
2006. A free energy principle for the brain. *Journal of physiology-Paris*, 100(1-3):70–87. 124

Gelada, C., S. Kumar, J. Buckman, O. Nachum, and M. G. Bellemare
2019. DeepMDP: Learning Continuous Latent Space Models for Representation Learning. In *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*, K. Chaudhuri and R. Salakhutdinov, eds., volume 97 of *Proceedings of Machine Learning Research*, Pp. 2170–2179. PMLR. 42, 43, 124, 128, 137, 142

Givan, R., T. L. Dean, and M. Greig
2003. Equivalence notions and model minimization in Markov decision processes. *Artif. Intell.*, 147(1-2):163–223. 42

Gulrajani, I., F. Ahmed, M. Arjovsky, V. Dumoulin, and A. C. Courville
2017. Improved Training of Wasserstein GANs. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, I. Guyon, U. v. Luxburg, S. Bengio, H. M. Wallach, R. Fergus, S. V. N. Vishwanathan, and R. Garnett, eds., Pp. 5767–5777. 144

Ha, D. and J. Schmidhuber
2018. World models. *arXiv preprint arXiv:1803.10122*. 41

Hafner, D., T. L. Deepmind, J. Ba, M. Norouzi, and G. Brain
2019a. Dream to Control: Learning Behaviors by Latent Imagination . 42

Hafner, D., T. Lillicrap, J. Ba, and M. Norouzi
2019b. Dream to control: Learning behaviors by latent imagination. *arXiv preprint arXiv:1912.01603*. 122

Hafner, D., T. Lillicrap, I. Fischer, R. Villegas, D. Ha, H. Lee, and J. Davidson
2019c. Learning latent dynamics for planning from pixels. In *International Conference on Machine Learning*, Pp. 2555–2565. 122

Hafner, D., T. P. Lillicrap, M. Norouzi, and J. Ba
2021. Mastering Atari with Discrete World Models. In *9th International Conference on Learning Representations , ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net. 42

Hansen, E. A. and S. Zilberstein
2001. LAO: A heuristic search algorithm that finds solutions with loops. *Artificial Intelligence*, 129(1-2):35–62. Publisher: Elsevier. 108

Hasselt, H. v., Y. Doron, F. Strub, M. Hessel, N. Sonnerat, and J. Modayil
2018. Deep Reinforcement Learning and the Deadly Triad. _eprint: 1812.02648. 32

Hausknecht, M. and P. Stone
2015. Deep recurrent q-learning for partially observable MDPs. In *AAAI Fall Symposium - Technical Report*, volume FS-15-06, Pp. 29–37. AI Access Foundation. 38, 39, 55

Hauskrecht, M.
1997. Incremental methods for computing bounds in partially observable Markov decision processes. In *AAAI/IAAI*, Pp. 734–739. 109

Hauskrecht, M.
2000. Value-Function Approximations for Partially Observable Markov Decision Processes. *J. Artif. Intell. Res.*, 13:33–94. 109, 130, 131

Heess, N., J. J. Hunt, T. P. Lillicrap, and D. Silver
2015. Memory-based control with recurrent neural networks. *arXiv preprint arXiv:1512.04455*. 39

Hochreiter, S. and J. Schmidhuber
1997. Long Short-Term Memory. *Neural Computation*, 9(8):1735–1780. Publisher: MIT Press. 38

Howard, R. A.
1960. Dynamic programming and markov processes. 29

Huang, B.
2020. Steady State Analysis of Episodic Reinforcement Learning. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, H. Larochelle, M. Ranzato, R. Hadsell, M.-F. Balcan, and H.-T. Lin, eds. 29

Igl, M., L. Zintgraf, T. A. Le, F. Wood, and S. Whiteson
2018. Deep variational reinforcement learning for POMDPs. In *35th International Conference on Machine Learning, ICML 2018*, volume 5. 39, 90, 122, 148

Jensen, J. L. W. V.
1906. Sur les fonctions convexes et les inégalités entre les valeurs moyennes. *Acta Mathematica*, 30(none):175 – 193. 24

Kaelbling, L. P., M. L. Littman, and A. R. Cassandra
1998. Planning and acting in partially observable stochastic domains. *Artificial intelligence*, 101(1-2):99–134. Publisher: Elsevier. 16, 18, 23, 26, 34, 35

Kantorovich, L. V.
1942. On the translocation of masses. In *Dokl. Akad. Nauk. USSR (NS)*, volume 37, Pp. 199–201. 25

Kingma, D. P., M. Welling, et al.
2013. Auto-encoding variational bayes. 39, 41

Klenke, A.
2008. *Probability theory: a comprehensive course*. Springer. 22

Klima, R., D. Bloembergen, R. Savani, K. Tuyls, A. Wittig, A. Sapera, and D. Izzo
2018. Space debris removal: Learning to cooperate and the price of anarchy. *Frontiers Robotics AI*, 5(JUN). Publisher: Frontiers Media S.A. 39, 47

Kocsis, L. and C. Szepesvári
2006. Bandit based monte-carlo planning. In *European conference on machine learning*, Pp. 282–293. Springer. 31

Konda, V. R. and J. N. Tsitsiklis
2000. Actor-critic algorithms. In *Advances in Neural Information Processing Systems*. ISSN: 10495258. 33

Krale, M., T. D. Simao, and N. Jansen
2023. Act-Then-Measure: Reinforcement Learning for Partially Observable Environments with Active Measuring. *Proceedings of the International Conference on Automated Planning and Scheduling*, 33(1):212–220. 90

Kullback, S. and R. A. Leibler
1951. On information and sufficiency. *The annals of mathematical statistics*, 22(1):79–86. 22, 24

Kurniawati, H., D. Hsu, and W. S. Lee
2009. SARSOP: Efficient Point-Based POMDP Planning by Approximating Optimally Reachable Belief Spaces. *Robotics: Science and systems*. Publisher: MIT Press. 37

Lambrechts, G., A. Bolland, and D. Ernst
2024. Informed pomdp: Leveraging additional information in model-based rl. In *Reinforcement Learning Conference*. 124

Larsen, K. G. and A. Skou
1991. Bisimulation through Probabilistic Testing. *Inf. Comput.*, 94(1):1–28. 42

Lee, A. X., A. Nagabandi, P. Abbeel, and S. Levine
2020. Stochastic latent actor-critic: Deep reinforcement learning with a latent variable model. *Advances in Neural Information Processing Systems*, 33:741–752. 124

Lee, J., A. Agarwal, C. Dann, and T. Zhang
2023. Learning in pomdps is sample-efficient with hindsight observability. In *International Conference on Machine Learning*, Pp. 18733–18773. PMLR. 124

Lin, L.-J.
1992. Self-improving reactive agents based on reinforcement learning, planning and teaching. *Machine Learning*, 8(3–4):293–321. 32

Littman, M. and R. S. Sutton
2001. Predictive representations of state. In *Advances in Neural Information Processing Systems*, T. Dietterich, S. Becker, and Z. Ghahramani, eds., volume 14. MIT Press. 122

Lowe, R., Y. Wu, A. Tamar, J. Harb, P. Abbeel, and I. Mordatch
2017. Multi-agent actor-critic for mixed cooperative-competitive environments. In *Advances in Neural Information Processing Systems*, volume 2017-Decem, Pp. 6380–6391. ISSN: 10495258. 16, 40, 50

Lyu, X., Y. Xiao, B. Daley, and C. Amato
2021. Contrasting Centralized and Decentralized Critics in Multi- Agent Reinforcement Learning. *Proc. of the 20th International Conference on AutonomousAgents and multi-agent Systems (AAMAS 2021)*. 41

Ma, X., P. Karkus, D. Hsu, W. S. Lee, and N. Ye
2020. Discriminative particle filter reinforcement learning for complex partial observations. In *International Conference on Learning Representations*. 122

Maaten, L. v. d. and G. Hinton
2008. Visualizing Data using t-SNE. *Journal of Machine Learning Research*, 9(86):2579–2605. 150

Madani, O., S. Hanks, and A. Condon
1999. On the undecidability of probabilistic planning and infinite-horizon partially observable Markov decision problems. *Aaai/iaai*, 10(315149.315395). 36

Mahajan, A., T. Rashid, M. Samvelyan, and S. Whiteson
2019. MAVEN: Multi-Agent Variational Exploration. *Advances in Neural Information Processing Systems*, 32. Publisher: Neural information processing systems foundation. 52

Melo, F. S. and M. Veloso
2009. Learning of coordination: Exploiting sparse interactions in multiagent systems. In *Proceedings of The 8th International Conference on Autonomous Agents and Multiagent Systems-Volume 2*, Pp. 773–780. 159

Mihaylov, M., K. Tuyls, and A. Nowé
2010. Decentralized Learning in Wireless Sensor Networks. In *Adaptive and Learning Agents*, M. Taylor and K. Tuyls, eds., Pp. 60–73, Berlin, Heidelberg. Springer Berlin Heidelberg. 39, 47

Mnih, V., A. P. Badia, M. Mirza, A. Graves, T. P. Lillicrap, T. Harley, D. Silver, and K. Kavukcuoglu
2016. Asynchronous Methods for Deep Reinforcement Learning. *33rd International*

Conference on Machine Learning, ICML 2016, 4:2850–2869. Publisher: International Machine Learning Society (IMLS). 33, 39

Mnih, V., K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, S. Petersen, C. Beattie, A. Sadik, I. Antonoglou, H. King, D. Kumaran, D. Wierstra, S. Legg, and D. Hassabis
2015. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533. Publisher: Nature Publishing Group. 31, 53

Morad, S., R. Kortvelesy, M. Bettini, S. Liwicki, and A. Prorok
2023. POPGym: Benchmarking Partially Observable Reinforcement Learning. In *The Eleventh International Conference on Learning Representations*. 149

Mordatch, I. and P. Abbeel
2017. Emergence of Grounded Compositional Language in Multi- Agent Populations. *32nd AAAI Conference on Artificial Intelligence, AAAI 2018*, Pp. 1495–1502. Publisher: AAAI press. 49, 67, 80

Müller, A.
1997. Integral Probability Metrics and Their Generating Classes of Functions. *Advances in Applied Probability*, 29(2):429–443. Publisher: Applied Probability Trust. 129

Nam, H. A., S. Fleming, and E. Brunskill
2021. Reinforcement Learning with State Observation Costs in Action-Contingent Noiselessly Observable Markov Decision Processes. In *Advances in Neural Information Processing Systems*, M. Ranzato, A. Beygelzimer, Y. Dauphin, P. S. Liang, and J. W. Vaughan, eds., volume 34, Pp. 15650–15666. Curran Associates, Inc. 89

Nilsson, N. J.
1968. Searching problem-solving and game-playing trees for minimal cost solutions. In *Proceedings IFIP Congress, 1968*, volume 2, Pp. 1556–1562. 89

Nilsson, N. J.
1982. *Principles of artificial intelligence*. Springer Science & Business Media. 88

Nishiyama, Y., A. Boularias, A. Gretton, and K. Fukumizu
2012. Hilbert Space Embeddings of POMDPs. In *Proceedings of the Twenty-Eighth Conference on Uncertainty in Artificial Intelligence*, UAI'12, Pp. 644–653, Arlington, Virginia, USA. AUAI Press. event-place: Catalina Island, CA. 124

Nowé, A., P. Vrancx, and Y. M. De Hauwere
2012. Game theory and multi-agent reinforcement learning. *Adaptation, Learning, and Optimization*, 12:441–470. Publisher: Springer Verlag. 50

Oliehoek, F. A. and C. Amato
2016. *A Concise Introduction to Decentralized POMDPs*. Publication Title: Springer Briefs in Intelligent Systems. 16, 40, 49, 54

Oliehoek, F. A., M. T. Spaan, and N. Vlassis
2008. Optimal and approximate Q-value functions for decentralized POMDPs. *Journal of Artificial Intelligence Research*, 32:289–353. 40, 49

Papadimitriou, C. H. and J. N. Tsitsiklis
1987. The complexity of Markov decision processes. *Mathematics of operations research*, 12(3):441–450. Publisher: INFORMS. 36

Papamakarios, G., T. Pavlakou, and I. Murray
2017. Masked Autoregressive Flow for Density Estimation. In *Advances in Neural Information Processing Systems*, I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, eds., volume 30. Curran Associates, Inc. 139

Pineau, J., G. Gordon, and S. Thrun
2003. Point-based value iteration: An anytime algorithm for POMDPs. In *Proceedings of 18th International Joint Conference on Artificial Intelligence (IJCAI '03)*, Pp. 1025 – 1032. 37

Pineau, J., G. Gordon, and S. Thrun
2006. Anytime point-based approximations for large POMDPs. *Journal of Artificial Intelligence Research*, 27:335–380. 111, 112

Pinto, L., M. Andrychowicz, P. Welinder, W. Zaremba, and P. Abbeel
2017. Asymmetric actor critic for image-based robot learning. *arXiv preprint arXiv:1710.06542*. 16

Puterman, M. L.
1990. Markov decision processes. *Handbooks in operations research and management science*, 2:331–434. Publisher: Elsevier. 26, 29

Puterman, M. L.
1994. *Markov Decision Processes: Discrete Stochastic Dynamic Programming*, Wiley Series in Probability and Statistics. Wiley. 130

Rashid, T., M. Samvelyan, C. S. de Witt, G. Farquhar, J. Foerster, and S. Whiteson
2018. QMIX: Monotonic Value Function Factorisation for Deep Multi-Agent Reinforcement Learning. *Proceedings of the35thInternational Conference on MachineLearning , Stockholm, Sweden, PMLR 80, 2018.* 47, 51

Röpke, W., R. Avalos, R. Rădulescu, A. Nowe, D. M. Roijers, and F. Delgrange
2025. Integrating RL and planning through optimal transport world models. In *The Seventeenth Workshop on Adaptive and Learning Agents*. 156

Ross, S., B. Chaib-Draa, and others
2007. AEMS: An anytime online search algorithm for approximate policy refinement in large POMDPs. In *IJCAI*, Pp. 2592–2598. 37, 87, 88, 99, 100

Ross, S., J. Pineau, S. Paquet, and B. Chaib-Draa
2008. Online planning algorithms for POMDPs. *Journal of Artificial Intelligence Research*, 32:663–704. 37, 87, 89

Samvelyan, M., T. Rashid, C. S. de Witt, G. Farquhar, N. Nardelli, T. G. J. Rudner, C.-M. Hung, P. H. S. Torr, J. Foerster, and S. Whiteson
2019. The StarCraft Multi-Agent Challenge. *Proceedings of the International Joint Conference on Autonomous Agents and Multiagent Systems, AAMAS*, 4:2186–2188. Publisher: International Foundation for Autonomous Agents and Multiagent Systems (IFAAMAS). 48, 67, 68, 70, 75

Satia, J. and R. Lave
1973. Markovian decision processes with probabilistic observation of states. *Management Science*, 20(1):1–13. Publisher: INFORMS. 89

Scherrer, B., V. Gabillon, M. Ghavamzadeh, and M. Geist
2012. Approximate modified policy iteration. *arXiv preprint arXiv:1205.3054*. 30

Schulman, J., P. Moritz, S. Levine, M. I. Jordan, and P. Abbeel
2016. High-dimensional continuous control using generalized advantage estimation. In *4th International Conference on Learning Representations , ICLR 2016 - Conference Track Proceedings*. International Conference on Learning Representations, ICLR. 33

Schulman, J., F. Wolski, P. Dhariwal, A. Radford, and O. Klimov
2017. Proximal Policy Optimization Algorithms. 33

SeyedSalehi, E., N. Akbarzadeh, A. Sinha, and A. Mahajan
2023. Approximate information state based convergence analysis of recurrent Q-learning. In *Sixteenth European Workshop on Reinforcement Learning*. 125

Silver, D., A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. v. d. Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot, S. Dieleman, D. Grewe, J. Nham, N. Kalchbrenner, I. Sutskever, T. Lillicrap, M. Leach, K. Kavukcuoglu, T. Graepel, and D. Hassabis
2016. Mastering the game of Go with deep neural networks and tree search. 31

Silver, D. and J. Veness
2010. Monte-Carlo Planning in Large POMDPs. In *Advances in Neural Information Processing Systems*, J. Lafferty, C. Williams, J. Shawe-Taylor, R. Zemel, and A. Culotta, eds., volume 23. Curran Associates, Inc. 37, 87, 89

Smallwood, R. D. and E. J. Sondik
1973. The Optimal Control of Partially Observable Markov Processes over a Finite Horizon. *Oper. Res.*, 21(5):1071–1088. 131

Smith, T. and R. Simmons
2004. Heuristic search value iteration for POMDPs. In *Proceedings of the 20th conference on Uncertainty in artificial intelligence*, Pp. 520–527. 37, 111

Somani, A., N. Ye, D. Hsu, and W. S. Lee
2013. DESPOT: Online POMDP planning with regularization. *Advances in neural information processing systems*, 26. 38

Sondik, E. J.
1971. *The optimal control of partially observable Markov processes.* Stanford University. 23, 35, 36

Sondik, E. J.
1978. The Optimal Control of Partially Observable Markov Processes over the Infinite Horizon: Discounted Costs. *Oper. Res.*, 26(2):282–304. 36, 131

Spaan, M. T. and N. Vlassis
2005. Perseus: Randomized point-based value iteration for POMDPs. *Journal of artificial intelligence research*, 24:195–220. 37

Sriperumbudur, B. K., K. Fukumizu, A. Gretton, B. Schölkopf, and G. R. Lanckriet
2009. On integral probability metrics,\ backslashphi-divergences and binary classification. *arXiv preprint arXiv:0901.2698*. 129

Sun, Q., D. Steckelmacher, Y. Yao, A. Nowe, and R. Avalos
2024. Dynamic size message scheduling for multi-agent communication under limited bandwidth. *IEEE Transactions on Mobile Computing*. 159

Sunehag, P., G. Lever, A. Gruslys, W. M. Czarnecki, V. Zambaldi, M. Jaderberg, M. Lanctot, N. Sonnerat, J. Z. Leibo, K. Tuyls, and T. Graepel
2018. Value-decomposition networks for cooperative multi-agent learning based on team reward. In *Proceedings of the International Joint Conference on Autonomous Agents and Multiagent Systems, AAMAS*, volume 3, Pp. 2085–2087. ISSN: 15582914. 47, 51, 67, 76

Sutton, R. and A. Barto
1998. Reinforcement Learning: An Introduction. *IEEE Transactions on Neural Networks*. 17, 130

Sutton, R. S., D. McAllester, S. Singh, and Y. Mansour
2000. Policy gradient methods for reinforcement learning with function approximation. In *Advances in Neural Information Processing Systems*. ISSN: 10495258. 33

Szepesvári, C. and R. Munos
2005. Finite-time bounds for sampling-based fitted value iteration. In *Proceedings of the 22nd International Conference on Machine Learning (ICML)*, Pp. 880–887. 30

Tampuu, A., T. Matiisen, D. Kodelja, I. Kuzovkin, K. Korjus, J. Aru, J. Aru, and R. Vicente
2015. Multiagent Cooperation and Competition with Deep Reinforcement Learning. *PLoS ONE*, 12(4). Publisher: Public Library of Science. 48, 54

Tampuu, A., T. Matiisen, D. Kodelja, I. Kuzovkin, K. Korjus, J. Aru, J. Aru, and R. Vicente
2017. Multiagent cooperation and competition with deep reinforcement learning. *PloS one*, 12(4):e0172395. 50, 51

Tan, M.
1993. Multi-Agent Reinforcement Learning: Independent vs. Cooperative Agents. In *Machine Learning Proceedings 1993*. 48, 50, 51, 54

Tolstikhin, I., O. Bousquet, S. Gelly, and B. Schoelkopf
2018. Wasserstein auto-encoders. In *International Conference on Learning Representations*. 25

Tsitsiklis, J. and B. Van Roy
1996. Analysis of temporal-diffference learning with function approximation. *Advances in neural information processing systems*, 9. 31

Tuyls, K. and G. Weiss
2012. Multiagent learning: Basics, challenges, and prospects. In *AI Magazine*. ISSN: 07384602. 39, 47

van Hasselt, H.
2010. Double Q-learning. In *Advances in Neural Information Processing Systems*, J. Lafferty, C. Williams, J. Shawe-Taylor, R. Zemel, and A. Culotta, eds., volume 23. Curran Associates, Inc. 32

van Hasselt, H., A. Guez, and D. Silver
2015. Deep Reinforcement Learning with Double Q-learning. *30th AAAI Conference on Artificial Intelligence, AAAI 2016*, Pp. 2094–2100. Publisher: AAAI press. 32

Vaswani, A., N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin
2017. Attention is all you need. In *Advances in Neural Information Processing Systems*, volume 2017-Decem, Pp. 5999–6009. ISSN: 10495258. 51

Villani, C.
2009. *Optimal Transport: Old and New*. Berlin, Heidelberg: Springer Berlin Heidelberg. 22, 25, 44

Wang, J., Z. Ren, T. Liu, Y. Yu, and C. Zhang
2021. QPLEX: Duplex Dueling Multi-Agent Q-Learning. *International Conference on Learning Representations*. 47, 51

Wang, T., H. Dong, V. Lesser, and C. Zhang
2020. ROMA: Multi-Agent Reinforcement Learning with Emergent Roles. *37th International Conference on Machine Learning, ICML 2020*, PartF168147-13:9818–9828. Publisher: International Machine Learning Society (IMLS). 52

Wang, Y. and X. Tan
2021. Deep recurrent belief propagation network for POMDPs. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, Pp. 10236–10244. Issue: 11. 124

Wang, Z., T. Schaul, M. Hessel, H. Van Hasselt, M. Lanctot, and N. De Frcitas
2016. Dueling Network Architectures for Deep Reinforcement Learning. In *33rd International Conference on Machine Learning, ICML 2016*, volume 4, Pp. 2939–2947. 32, 64, 66

Washington, R.
1997. BI-POMDP: Bounded, incremental partially-observable Markov -model planning. In *European Conference on Planning*, Pp. 440–451. Springer. 89

Watkins, C. J. C. H. and P. Dayan
1992. Q-learning. *Machine Learning*. 31, 89

Williams, R. J.
1992. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8:229–256. 33

Xu, L., S. Gholami, S. M. Carthy, B. Dilkina, A. Plumptre, M. Tambe, R. Singh, M. Nsubuga, J. Mabonga, M. Driciru, F. Wanyama, A. Rwetsiba, T. Okello, and E. Enyel
2020. Stay ahead of poachers: Illegal wildlife poaching prediction and patrol planning under uncertainty with field test evaluations (Short Version). *Proceedings - International Conference on Data Engineering*, 2020-April:1898–1901. Publisher: IEEE Computer Society. 39, 47

Yang, Y., J. Hao, B. Liao, K. Shao, G. Chen, W. Liu, and H. Tang
2020. Qatten: A General Framework for Cooperative Multiagent Reinforcement Learning. 51

Young, K. and T. Tian
2019. MinAtar: An Atari-Inspired Testbed for Thorough and Reproducible Reinforcement Learning Experiments. *arXiv preprint arXiv:1903.03176*. 149

Yu, C., A. Velu, E. Vinitsky, Y. Wang, A. Bayen, and Y. Wu
2021. The Surprising Effectiveness of PPO in Cooperative, Multi -Agent Games. 52

Zahavy, T., M. Haroush, N. Merlis, D. J. Mankowitz, and S. Mannor
2018. Learn what not to learn: Action elimination with deep reinforcement learning. In *Advances in Neural Information Processing Systems*, volume 2018-Decem, Pp. 3562–3573. ISSN: 10495258. 72

Zhang, A., R. McAllister, R. Calandra, Y. Gal, and S. Levine
2020. Learning invariant representations for reinforcement learning without reconstruction. *arXiv preprint arXiv:2006.10742*. 42, 43

Åström, K. J.
1965. Optimal control of Markov processes with incomplete state information. *Journal of Mathematical Analysis and Applications*, 10(1):174–205. 16, 34, 35

Publication Previews of the papers included in this thesis

Local Advantage Networks

for Multi-Agent Reinforcement Learning in Dec-POMDPs

Raphael Avalos Mathieu Reymond
Ann Nowe Diederik M Roijers

Abstract

Many recent successful off-policy multi-agent reinforcement learning (MARL) algorithms for cooperative partially observable environments focus on finding factorized value functions, leading to convoluted network structures. Building on the structure of independent Q-learners, our LAN algorithm takes a radically different approach, leveraging a dueling architecture to learn for each agent a decentralized best-response policies via individual advantage functions. The learning is stabilized by a centralized critic whose primary objective is to reduce the moving target problem of the individual advantages. The critic, whose network's size is independent of the number of agents, is cast aside after learning. Evaluation on the StarCraft II multi-agent challenge benchmark shows that LAN reaches state-of-the-art performance and is highly scalable with respect to the number of agents, opening up a promising alternative direction for MARL research.

Publication Information:

- **Journal:** Transactions on Machine Learning Research (TMLR) 2023
- **Workshop:** EWRL 2023
- **Conference:** AAMAS 2022 (Extended Abstract)

Online Planning in POMDPs with State-Requests

Raphael Avalos Eugenio Bargiacchi
Ann Nowé Diederik M Roijers Frans Oliehoek

Abstract

In key real-world problems, full state information is sometimes available but at a high cost, like activating precise yet energy-intensive sensors or consulting humans, thereby compelling the agent to operate under partial observability. For this scenario, we propose AEMS-SR (Anytime Error Minimization Search with State Requests), a principled online planning algorithm tailored for POMDPs with state requests. By representing the search space as a graph instead of a tree, AEMS-SR avoids the exponential growth of the search space originating from state requests. Theoretical analysis demonstrates AEMS-SR's -optimality, ensuring solution quality, while empirical evaluations illustrate its effectiveness compared with AEMS and POMCP, two SOTA online planning algorithms. AEMS-SR enables efficient planning in domains characterized by partial observability and costly state requests offering practical benefits across various applications.

Publication Information:

- **Journal:** Reinforcement Learning Journal (RLJ) 2024
- **Workshop:** EWRL 2024

The Wasserstein Believer

Learning Belief Updates for Partially Observable Environments through Reliable Latent Space Models

Raphael Avalos[*] Florent Delgrange[*]
Ann Nowe Guillermo Perez Diederik M Roijers

[*] *Equal contribution*

Abstract

Partially Observable Markov Decision Processes (POMDPs) are useful tools to model environments where the full state cannot be perceived by an agent. As such the agent needs to reason taking into account the past observations and actions. However, simply remembering the full history is generally intractable due to the exponential growth in the history space. Keeping a probability distribution that models the belief over what the true state is can be used as a sufficient statistic of the history, but its computation requires access to the model of the environment and is also intractable. State-of-the-art algorithms use Recurrent Neural Networks to compress the observation-action history aiming to learn a sufficient statistic, but they lack guarantees of success and can lead to sub-optimal policies. To overcome this, we propose the Wasserstein Belief Updater, an RL algorithm that learns a latent model of the POMDP and an approximation of the belief update. Our approach comes with theoretical guarantees on the quality of our approximation ensuring that our outputted beliefs allow for learning the optimal value function.

Publication Information:

- **Conference:** ICLR 2024
- **Workshop:** EWRL 2024

Publication Previews of others papers produced during the PhD

Integrating RL and Planning through Optimal Transport World Models

Willem Röpke* Raphaël Avalos*
Roxana Rădulescu Ann Nowe Diederik M Roijers Florent Delgrange

* *Equal contribution*

Abstract

We introduce Optimal Transport MDPs (OT-MDPs), a framework for learning principled latent world models via optimal transport. Our approach formulates a generic optimal transport objective that trains a generative model of the environment by minimising a customisable cost function, which quantifies the discrepancy between latent and real trajectories. Through this perspective, we highlight the limitations of reconstruction-based methods and establish conditions on the cost function that enable theoretical guarantees. The quality of the learned model allows us to integrate reinforcement learning and planning methods. In particular, we leverage model-based value expansion to refine value estimates, providing rigorous theoretical justification. Additionally, we examine the use of Monte Carlo tree search and provide a theoretical analysis of the assumptions under which its application remains sound. Empirical evaluation across four MinAtar environments demonstrates that OT-MDPs yield high-fidelity models, leading to strong performance. Moreover, our results reveal challenges associated with planning in the latent model, suggesting critical directions for future research.

Publication Information:

- **Workshop:** ALA 2025

Sample, Predict, then Proceed

Self-Verification Sampling for Tool Use of LLMs

Shangmin Guo Omar Darwiche Domingues Raphaël Avalos
Aaron Courville Florian Strub

Abstract

Tool use in stateful environments presents unique challenges for large language models (LLMs), where existing test-time compute strategies relying on repeated trials in the environment are impractical. We propose dynamics modelling (DyMo), a method that augments LLMs with a state prediction capability alongside function calling during post-training. This enables LLMs to predict the future states of their actions through an internal environment model. On the Berkeley Function Calling Leaderboard V2, DyMo improves success rates and significantly reduces hallucinations. We further integrate the internal environment model into self-verification sampling (SVS), and show that this substantially improves pass$\hat{k}$ over number of trials k, and allows the model to refuse unreliable outputs. Together, DyMo and SVS greatly enhance the effectiveness and reliability of LLMs for tool use. We believe this work charts a path towards scalable planning RL methods for LLM inference without repeatedly querying the oracle environment.

Publication Information:

- **Preprint** : arXiv:2506.02918

ShiQ: Bringing back Bellman to LLMs

Pierre Clavier Nathan Grinsztajn Raphael Avalos
Yannis Flet-Berliac Irem Ergun Omar D. Domingues Eugene Tarassov
Olivier Pietquin Pierre H. Richemond Florian Strub Matthieu Geist

Abstract

The fine-tuning of pre-trained large language models (LLMs) using reinforcement learning (RL) is generally formulated as direct policy optimization. This approach was naturally favored as it efficiently improves a pretrained LLM, seen as an initial policy. Another RL paradigm, Q-learning methods, has received far less attention in the LLM community while demonstrating major success in various non-LLM RL tasks. In particular, Q-learning effectiveness comes from its sample efficiency and ability to learn offline, which is particularly valuable given the high computational cost of sampling with LLMs. However, naively applying a Q-learning-style update to the model's logits is ineffective due to the specificity of LLMs. Our core contribution is to derive theoretically grounded loss functions from Bellman equations to adapt Q-learning methods to LLMs. To do so, we carefully adapt insights from the RL literature to account for LLM-specific characteristics, ensuring that the logits become reliable Q-value estimates. We then use this loss to build a practical algorithm, ShiQ for Shifted-Q, that supports off-policy, token-wise learning while remaining simple to implement. Finally, we evaluate ShiQ on both synthetic data and real-world benchmarks, e.g., UltraFeedback and BFCL-V3, demonstrating its effectiveness in both single-turn and multi-turn LLM settings

Publication Information:

- **Preprint** : NeurIPS 2025

Exploration and Communication for Partially Observable Collaborative Multi-Agent Reinforcement Learning

Raphaël Avalos

Abstract

Multi-agent reinforcement learning (MARL) enables us to create adaptive agents in challenging environments, even when the agents have limited observation. The cooperative multi-agent setting introduces numerous challenges compared to the single-agent setting, such as the moving target problem and the curse of dimensionality with respect to the action space. It also aggravates the credit assignment problem, as the credit is not only spread across a sequence of actions, but also multiple agents. This setting also introduces new possibilities, such as task parallelization, specialization, and communication. The Centralized Training with Decentralized Execution paradigm has emerged as a popular strategy to mitigate some of the difficulties in MARL, while still ensuring that the policy of the agents is only conditioned on their local history. However, how to fully leverage this paradigm is still an open question. During the first year of my Ph. D., I developed a novel algorithm, Local Advantage Networks (LAN), that proposes an alternative direction to value factorization, that is more scalable, not limited in its representation and state-of-the-art. The next parts of my research will focus on multi-agent exploration and learning to communicate.

Publication Information:

- **Conference:** AAMAS 2022 (Doctoral Consortium)

Dynamic Size Message Scheduling

for Multi-Agent Communication under Limited Bandwidth

Qingshuang Sun Denis Steckelmacher
Yuan Yao Ann Nowé Raphaël Avalos

Abstract

Communication plays a vital role in multi-agent systems, fostering collaboration and coordination. However, in real-world scenarios where communication is bandwidth-limited, existing multi-agent reinforcement learning (MARL) algorithms often provide agents with a binary choice: either transmitting a fixed amount of data or no information at all. This rigid communication strategy hinders the ability to effectively utilize bandwidth. To overcome this challenge, we present the Dynamic Size Message Scheduling (DSMS) method, which introduces finer-grained communication scheduling by considering the actual size of the information being exchanged. Our approach lies in adapting message sizes using Fourier transform-based compression techniques with clipping, enabling agents to tailor their messages to match the allocated bandwidth according to importance weights. This method realizes a balance between information loss and bandwidth utilization. Receiving agents reliably decompress the messages using the inverse Fourier transform. We evaluate DSMS in cooperative tasks where the agent has partial observability. Experimental results demonstrate that DSMS significantly improves performance by optimizing the utilization of bandwidth and effectively balancing information importance.

Publication Information:

- **Journal:** IEEE Transactions on Mobile Computing (TMC) 2024

Command a

An enterprise-ready large language model

Team Cohere (*including Raphaël Avalos*)

Abstract

In this report we describe the development of Command A, a powerful large language model purpose-built to excel at real-world enterprise use cases. Command A is an agent-optimised and multilingual-capable model, with support for 23 languages of global business, and a novel hybrid architecture balancing efficiency with top of the range performance. It offers best-in-class Retrieval Augmented Generation (RAG) capabilities with grounding and tool use to automate sophisticated business processes. These abilities are achieved through a decentralised training approach, including self-refinement algorithms and model merging techniques. We also include results for Command R7B which shares capability and architectural similarities to Command A. Weights for both models have been released for research purposes. This technical report details our original training pipeline and presents an extensive evaluation of our models across a suite of enterprise-relevant tasks and public benchmarks, demonstrating excellent performance and efficiency.

Publication Information:

- **Preprint** : arXiv:2504.00698

Multi-agent RMax for multi-agent multi-armed bandits

Eugenio Bargiacchi Raphaël Avalos Timothy Verstraeten
Pieter Libin Ann Nowé Diederik M Roijers

Abstract

In this paper, we provide PAC bounds for best-arm identificationin multi-agent multi-armed bandits (MAMABs), via an algorithmwe call multi-agent RMax (MARMax). In a MAMAB, the rewardstructure is expressed as a coordination graph, i.e., the total teamreward is a sum over local reward functions that are conditionedon the actions of a subset of the agents. Assuming that these localrewards functions are bounded, we derive a PAC(δ, ϵ) learningalgorithm that achieves an accuracy of ϵ relative to the maximumteam reward, with a number of required samples at most linear inthe size of the largest reward reward function with probability $1-\delta$. Furthermore, we show that in practice MARMax performs muchbetter than this theoretical upper bound on the sample complexityby an order of magnitude. We further improve on the empiricalresult by tempering the optimism used in MARMax, resulting ina new algorithm that we call MAVMax. We show empirically thatMAVMax further cuts down the number of required samples byaround 30% w.r.t. MARMax.

Publication Information:

- **Workshop:** ALA 2022

Laser Learning Environment

A new environment for coordination-critical multi-agent tasks

Yannick Molinghen Raphaël Avalos
Mark Van Achter Ann Nowé Tom Lenaerts

Abstract

We introduce the Laser Learning Environment (LLE), a collaborative multi-agent reinforcement learning environment in which coordination is central. In LLE, agents depend on each other to make progress (interdependence), must jointly take specific sequences of actions to succeed (perfect coordination), and accomplishing those joint actions does not yield any intermediate reward (zero-incentive dynamics). The challenge of such problems lies in the difficulty of escaping state space bottlenecks caused by interdependence steps since escaping those bottlenecks is not rewarded. We test multiple state-of-the-art value-based MARL algorithms against LLE and show that they consistently fail at the collaborative task because of their inability to escape state space bottlenecks, even though they successfully achieve perfect coordination. We show that Q-learning extensions such as prioritized experience replay and n-steps return hinder exploration in environments with zero-incentive dynamics, and find that intrinsic curiosity with random network distillation is not sufficient to escape those bottlenecks. We demonstrate the need for novel methods to solve this problem and the relevance of LLE as cooperative MARL benchmark.

Publication Information:

- **Conference:** BNAIC/BENELEARN 2023 (Best Paper Award)

Tackling Scheduling Problems with Graph Structured Reinforcement Learning

Seppe Renty Raphaël Avalos Andries Rosseau Ann Nowé

Abstract

Job scheduling is a central element of our society. While this problem has been actively researched by the field Operations Research (OR), yielding good algorithms, these solutions are usually not generalizable across different problem instances. They also tend to behave poorly when there is uncertainty during execution. In this thesis, we research whether dispatching rules that generalize to groups of similar problem instances can be learned through Reinforcement learning. To leverage the complex structure of a Job scheduling problem we designed a graph representation of the problem. We then used a combination of Graph Neural Networks (GCN) and Multi-Agent Reinforcement Learning (MARL) to improve on existing dispatching rules. We found that the technique generates shorter schedules on average than popular dispatching rules for multiple families of problems.

Publication Information:

- **Conference:** BNAIC/BENELEARN 2022

Toward evolutionary autocurricula

Emergent sociality from inclusive rewards

Andries Rosseau Raphaël Avalos Ann Nowe

Abstract

The competitive and cooperative forces of natural selection have driven the evolution of intelligence for many millions of years, eventually culminating in nature's vast biodiversity and the complexity of our human minds. In this paper, we present a novel multi-agent reinforcement learning framework, inspired by the process of evolution. We assign a genotype to each agent, and propose an inclusive reward that optimizes for the fitness of an agent's genes. Since an agent's genetic material can be present in other agents as well, our inclusive reward also takes genetically related individuals into account. We study the effect of inclusion on the resulting social dynamics in two network games with prisoner's dilemmas, and find that our results follow well-established principles from biology. Furthermore, we lay the foundation for future work in a more open-ended 3D environment, where agents have to ensure the survival of their genes in a natural world with limited resources. We hypothesize the emergence of an arms race of strategies, where each new strategy will be a gradual improvement in response to an earlier adaptation from other agents, effectively creating a multi-agent autocurriculum similar to biological evolution. Our evolutionary rewards provide a novel social dimension that features a non-stationary spectrum of cooperation due to the finite environmental resources and changing population distribution. It has the potential to create increasingly advanced strategies, where agents learn to balance cooperative and competitive incentives in a more complex and dynamic setup than previous works, where agents were often confined to predefined team setups that did not entail the social intricacies that biological evolution has. We argue this could be an important contribution towards creating advanced, general and socially intelligent agents.

Publication Information:

- **Workshop:** Cells2Societies 2022 (at ICLR)