

Exploring Learned Abstract Models For Efficient Planning and Learning

He, J.

DOI

[10.4233/uuid:73d7596e-7909-4186-85e6-845ae4cf2372](https://doi.org/10.4233/uuid:73d7596e-7909-4186-85e6-845ae4cf2372)

Publication date

2025

Document Version

Final published version

Citation (APA)

He, J. (2025). *Exploring Learned Abstract Models For Efficient Planning and Learning*. [Dissertation (TU Delft), Delft University of Technology]. <https://doi.org/10.4233/uuid:73d7596e-7909-4186-85e6-845ae4cf2372>

Important note

To cite this publication, please use the final published version (if applicable).
Please check the document version above.

Copyright

Other than for strictly personal use, it is not permitted to download, forward or distribute the text or part of it, without the consent of the author(s) and/or copyright holder(s), unless the work is under an open content license such as Creative Commons.

Takedown policy

Please contact us and provide details if you believe this document breaches copyrights.
We will remove access to the work immediately and investigate your claim.

Exploring Learned Abstract Models for Efficient Planning and Learning



Jinke He

EXPLORING LEARNED ABSTRACT MODELS FOR EFFICIENT PLANNING AND LEARNING

EXPLORING LEARNED ABSTRACT MODELS FOR EFFICIENT PLANNING AND LEARNING

Dissertation

for the purpose of obtaining the degree of doctor at
Delft University of Technology

by the authority of the Rector Magnificus
Prof.dr.ir. T.H.J.J. van der Hagen,
chair of the Board for Doctorates

to be defended publicly on
Wednesday 4 June 2025 at 15:00 hours

by

Jinke HE

Master of Science in Computer Science,
University of Oxford, United Kingdom,
born in Changzhou, China.

This dissertation has been approved by the promoters.

Composition of the doctoral committee:

Rector Magnificus,	chairperson
Dr. F.A. Oliehoek,	Delft University of Technology, promoter
Prof.dr. C.M. Jonker,	Delft University of Technology, promoter

Independent members:

Prof.dr. M.T.J. Spaan,	Delft University of Technology
Dr. H.J.S. Baier,	Eindhoven University of Technology
Dr. J. Alonso-Mora,	Delft University of Technology
Prof.dr. L.P. Kaelbling,	Massachusetts Institute of Technology
Prof.dr. M.M. de Weerd,	Delft University of Technology, reserve member

This project had received funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (grant agreement No. 758824 –INFLUENCE).

SIKS Dissertation Series No. 2025-28.

The research reported in this thesis has been carried out under the auspices of SIKS, the Dutch Research School for Information and Knowledge Systems.



Keywords: Reinforcement Learning, Online Planning, Abstraction

Printed by: Proefschriftmaken

Cover: Jinke He

Style: TU Delft House Style, with modifications by Moritz Beller
<https://github.com/Inventitech/phd-thesis-template>

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

To my dreams of youth

致我年少时的梦

CONTENTS

Summary	ix
1 Introduction	1
1.1 Sequential decision-making	2
1.1.1 Models	2
1.1.2 Solution Methods	2
1.2 Abstractions in Sequential Decision-Making	4
1.2.1 Action and Temporal Abstractions	4
1.2.2 State abstractions	5
1.3 Contributions of this thesis.	7
1.3.1 Learned Abstract Models for Efficient Online Planning.	7
1.3.2 Learned Abstract Models in Model-Based Reinforcement Learning.	9
1.4 Additional Research	10
1.4.1 Scaling up Deep RL with IALS	10
1.4.2 The 2022 Pathways to Net Zero RL Challenge	10
1.4.3 The Second Neural MMO Challenge	11
1.4.4 Other Collaborations.	12
2 Background	13
2.1 Sequential Decision-Making Problems	14
2.1.1 Markov Decision Processes.	14
2.1.2 Partially Observable MDPs.	16
2.1.3 Factored POMDPs	17
2.2 Influence-Based Abstraction	18
2.2.1 Categories of State Variables	18
2.2.2 The Local Transition, Observation and Reward functions.	20
2.2.3 Influence-Augmented Local Model	21
2.2.4 Remarks	24
2.3 Sample-Based Online Planning	25
2.3.1 Monte-Carlo Tree Search.	25
2.3.2 Monte-Carlo Planning in Large POMDPs.	26
2.4 Model-Based Reinforcement Learning	26
2.4.1 AlphaZero and Expert Iteration	27
2.4.2 The Value-Equivalence Principle	28
2.4.3 MuZero	29

3	Learning and Planning with Influence-augmented Local Simulators	31
3.1	Introduction	32
3.2	Influence-augmented Local Simulators	33
3.3	Empirical Analysis	35
3.4	Related Work.	39
3.5	Conclusion.	40
3.6	Appendix	41
4	Learning and Planning with Self-improving Simulators	45
4.1	Introduction	46
4.2	Self-improving Simulators	47
4.3	Empirical Analysis	53
4.4	Related Work.	57
4.5	Conclusion.	57
4.6	Appendix	58
5	What Model Does MuZero Learn?	69
5.1	Introduction	70
5.2	Policy Evaluation Experiments	72
5.3	Policy Improvement Experiments	78
5.4	Discussion	80
5.5	Conclusion.	81
5.6	Appendix	82
6	Conclusion	85
6.1	The Big Picture: The Evolution of Human Decision-Making	86
6.2	Contributions and Insights	92
6.3	Limitations and Future Work.	93
	Acknowledgments	113
	Curriculum Vitæ	117
	List of Publications	119
	List of Supervised Student Theses	121
	SIKS Dissertations	123

SUMMARY

This thesis investigates the role of learned abstract models in online planning and model-based reinforcement learning (MBRL). We explore how abstract models can accelerate search in online planning and evaluate their effectiveness in supporting policy evaluation and improvement in MBRL.

In online planning, we focus on reducing the high computational cost of simulating large, factored, partially observable environments. In Chapter 3, we introduce the influence-augmented local simulator (IALS), which approximates external influences while preserving local agent interactions. By replacing the full simulator with IALS, we enable faster planning while maintaining decision quality. We propose a two-phase approach where the influence model is trained offline and later integrated into planning, allowing significantly more simulations within a fixed computational budget. However, this approach has limitations, including potential distribution shifts and the risk of poor generalization.

To address these issues, Chapter 4 introduces the self-improving simulator, which eliminates offline training by learning the abstract model online during planning. A simulator selection mechanism dynamically balances the use of the learned and original simulators, improving computational efficiency over time while ensuring planning accuracy. Our results show that this approach avoids distribution shift issues, prevents premature reliance on inaccurate models, and removes the delay associated with offline training.

In MBRL, we examine the effectiveness of MuZero’s learned model in supporting policy evaluation and improvement. In Chapter 5, we analyze how well MuZero’s model generalizes beyond its training distribution and find that it struggles to support planning "outside the box" due to accumulated model inaccuracies. However, we show that MuZero’s learned policy prior mitigates these errors by guiding the search toward regions where the model is more reliable. This insight highlights the dual role of the policy prior—not only improving search efficiency but also compensating for model imperfections, contributing to MuZero’s strong empirical performance.

Overall, this thesis advances the understanding of learned abstract models in sequential decision-making, demonstrating their potential to improve computational efficiency while identifying key limitations in their ability to support planning. We hope these findings encourage further research into abstraction-driven approaches for adaptive, scalable decision-making in complex environments.

SAMENVATTING

Dit proefschrift onderzoekt de rol van geleerde abstracte modellen in online planning en model-gebaseerde reinforcement learning (MBRL). We verkennen hoe abstracte modellen het zoeken in online planning kunnen versnellen en evalueren hun effectiviteit in het ondersteunen van beleidsevaluatie en -verbetering binnen MBRL.

Bij online planning richten we ons op het verminderen van de hoge rekentijd die nodig is om grote, gefactoreerde en partieel observeerbare omgevingen te simuleren. In Hoofdstuk 3 introduceren we de influence-augmented local simulator (IALS), die externe invloeden benadert terwijl lokale interacties tussen agenten behouden blijven. Door de volledige simulator te vervangen door IALS, maken we snellere planning mogelijk zonder in te boeten aan beslissingskwaliteit. We stellen een tweefasenaanpak voor waarbij het invloedsmodel offline wordt getraind en later in de planning wordt geïntegreerd, waardoor aanzienlijk meer simulaties binnen een vast rekencapaciteitsbudget kunnen worden uitgevoerd. Deze aanpak heeft echter beperkingen, zoals mogelijke distributieveverschuivingen en het risico op slechte generalisatie.

Om deze problemen aan te pakken, introduceert Hoofdstuk 4 de self-improving simulator, die offline training overbodig maakt door het abstracte model online tijdens de planning te leren. Een simulatiekeuzemechanisme balanceert dynamisch het gebruik van de geleerde en originele simulatoren, waardoor de computationele efficiëntie in de loop van de tijd verbetert terwijl de nauwkeurigheid van de planning gewaarborgd blijft. Onze resultaten tonen aan dat deze aanpak distributieveverschuivingen vermijdt, voortijdig vertrouwen in onnauwkeurige modellen tegengaat en de vertraging van offline training wegneemt.

Binnen MBRL onderzoeken we de effectiviteit van het geleerde model van MuZero in het ondersteunen van beleidsevaluatie en -verbetering. In Hoofdstuk 5 analyseren we hoe goed het model van MuZero generaliseert buiten de trainingsdistributie en ontdekken we dat het moeite heeft met planning “buiten de gebaande paden” vanwege opgebouwde modelonnauwkeurigheden. We laten echter zien dat de door MuZero geleerde beleidsprior deze fouten verzacht door de zoekrichting te sturen naar gebieden waar het model betrouwbaarder is. Dit inzicht benadrukt de dubbele rol van de beleidsprior: niet alleen het verbeteren van de zoek efficiëntie, maar ook het compenseren van modelimperfecties, wat bijdraagt aan MuZero’s sterke empirische prestaties.

Samengevat draagt dit proefschrift bij aan het begrip van geleerde abstracte modellen bij sequentiële besluitvorming. Het toont hun potentieel om de computationele efficiëntie te verbeteren en identificeert tegelijkertijd belangrijke beperkingen in hun vermogen om planning te ondersteunen. We hopen dat deze bevindingen verder onderzoek stimuleren naar op abstractie gebaseerde benaderingen voor adaptieve, schaalbare besluitvorming in complexe omgevingen.

1

INTRODUCTION

"A journey of a thousand miles begins with a single step."

Laozi

Concepts such as space, time, and causality constitute the physical world that we inhabit. Within this world, diverse systems, from humans and animals to coffee machines and chairs, constantly interact with each other. The behavior of these systems varies in complexity. Some, like coffee machines and chairs, follow predictable patterns and are relatively easy to model. Others, in contrast, exhibit complex behavior by observing and reasoning about other systems and act accordingly.

This evolving landscape of complex systems and interactions fundamentally shapes one of the central questions in artificial intelligence (AI): how can we build autonomous systems, or agents, that are capable of making decisions in environments full of other complex interacting systems? This question not only challenges our understanding of intelligence but also has many real-world applications. Indeed, recent advances in AI have led to a surge of success in creating intelligent agents capable of tackling a wide range of real-world tasks previously considered too difficult for machines. These tasks range from mastering games such as Go [Silver et al., 2016, 2017b, 2018], Starcraft [Vinyals et al., 2019] and Dota 2 [OpenAI et al., 2019b], solving combinatorial optimization problems, such as chip design [Mirhoseini et al., 2021], neural network architecture search [Zoph and Le, 2016] and causal discovery [Zhu et al., 2019], to controlling complex systems, such as cooling systems of buildings [Luo et al., 2022], robotic systems [Kober et al., 2013] and nuclear fusion plasma [Degraeve et al., 2022].

1.1 SEQUENTIAL DECISION-MAKING

The first step towards building intelligent agents capable of solving these tasks is often formulating them as sequential decision-making problems. In this framework, an agent interacts with an environment in discrete time steps. At each time step, the agent observes the environment and chooses an action, causing the environment to transition to a new state. The agent's goal is then to maximize the rewards accumulated over time, which are numerical signals that indicate how well the agent is doing.

1.1.1 MODELS

Researchers have developed a rich set of mathematical models and solution methods to tackle these problems, each tailored to specific assumptions and settings. Markov decision processes (MDPs) [Bellman, 1957b] form a foundational model for sequential decision-making. In MDPs, it is assumed that an agent receives observations that constitute a Markovian signal of the underlying environment state, enabling optimal decision-making based solely on the current observation. Partially observable MDPs (POMDPs) [Kaelbling et al., 1998] relax this assumption by considering situations where the environment state is only partially observable. Consequently, acting optimally in POMDPs requires agents to reason about the hidden state by integrating their past actions and observations, rather than relying solely on the current observation. Extending sequential decision-making problems to multi-agent contexts, partially observable stochastic games (POSGs) [Hansen et al., 2004, Shapley, 1953] model interactions among multiple agents as well as their environment. A notable special case of POSGs is the class of decentralized POMDPs (Dec-POMDPs) [Oliehoek and Amato, 2016], in which agents collaborate to maximize a shared reward. Formulating sequential decision-making problems using these mathematical frameworks allows for a formal definition of the challenges addressed in this thesis and facilitates the principled development and application of solution methods.

1.1.2 SOLUTION METHODS

A diverse range of solution methods have been proposed for sequential decision-making problems, each suited to different assumptions and paradigms. Although the overarching goal is similar—to develop autonomous agents capable of effective decision-making within specific environments—these methods differ considerably. For comprehensive overviews, readers can consult textbooks such as Wiering and Van Otterlo [2012], Sutton and Barto [2018], and Kochenderfer et al. [2022]. For our purposes, we will focus on a key distinction: *what knowledge about the environment is available when constructing the agents?* In practice, this translates to the question of what type of simulator or model we assume is accessible when developing decision-making agents. This distinction serves as our primary criterion for classifying sequential decision-making methods.

Environment models used for decision-making can generally be categorized into three types: *full models*, *generative models*, and *trajectory models*. A full model provides a complete mathematical description of the environment, specifying precise probability distributions for state transitions and rewards. Given this comprehensive representation, optimal policies

can be computed through dynamic programming methods such as value iteration [Bellman, 1957a] and policy iteration [Howard, 1960].

In contrast, generative and trajectory models do not provide explicit probability distributions of future states and rewards, making exact computations infeasible. Instead, they require the use of sampling-based methods. Generative models, for example, allow simulation of possible future states and rewards from *any* arbitrary state-action pair. This capability enables the use of online planning approaches, where agents select actions by simulating future outcomes. Monte-Carlo Tree Search (MCTS) [Browne et al., 2012, Coulom, 2006, Kocsis and Szepesvári, 2006] exemplifies this class, incrementally building a search tree through simulations to identify promising actions.

Unlike generative models, trajectory models permit the simulation of only a single next state from the current state-action pair. This closely resembles interactions with real environments, where arbitrary state resets are impossible; indeed, the real environment itself can be viewed as a trajectory model. Consequently, methods designed for trajectory models are inherently suited to learning in real environments. Reinforcement learning (RL) [Sutton and Barto, 2018], which learns policies through trial-and-error interactions, is the predominant method within this category. A key advantage of RL is its ability to operate without explicit environment models or simulators. However, practical constraints such as safety risks and poor sample efficiency often limit direct learning in real environments. Consequently, RL methods are typically employed within simulators or models prior to deployment—a paradigm known as simulated reinforcement learning.

Importantly, full models can often be trivially transformed into generative models, and generative models into trajectory models, establishing a natural hierarchy from most expressive (full models) to least expressive (trajectory models). Consequently, trajectory-model-based methods like RL can, in principle, also be applied within full or generative model settings. In practice, RL methods often hold advantages over dynamic programming and online planning approaches (when all of them are applicable), particularly in scenarios involving continuous or high-dimensional state and action spaces, and when generalization, scalability, or rapid deployment-time decisions are required. The integration of neural network-based function approximation has recently led to the emergence of deep reinforcement learning (deep RL) [François-Lavet et al., 2018] as a highly flexible framework capable of effectively handling complex, high-dimensional decision-making problems.

Nevertheless, RL methods are not universally superior. Deep RL training typically suffers from issues such as sample inefficiency, computational expense, training instability, and sensitivity to hyperparameters and implementation nuances [Henderson et al., 2018, Ipan, 2018, Islam et al., 2017, Mannor and Tamar, 2023]. Online planning approaches, which avoid explicit training phases, circumvent these limitations by focusing solely on selecting effective actions for the current state. This makes them particularly suitable for large state-space problems and environments subject to dynamic changes, conditions under which learned RL policies often struggle. Finally, as explored in Section 2.4.1, RL and online planning can be effectively combined under the expert iteration framework [Anthony et al., 2017] to develop more efficient algorithms for sequential decision-making.

1.2 ABSTRACTIONS IN SEQUENTIAL DECISION-MAKING

Despite the significant progress in sequential decision-making problems, many challenges remain. For example, in environments characterized by intricate interacting systems, the vastness of the state space poses a significant challenge for decision-making agents operating with limited resources. This challenge is particularly acute in online planning scenarios, where agents must predict future states to make informed decisions. The computational demands of simulating the entire environment can severely constrain the agent's ability to act both efficiently and effectively. Similarly, in RL, the task of "representing" a good policy (i.e., remembering what to do in each state) in a complex environment can already be challenging due to the curse of dimensionality, let alone the sample complexity required to find a good policy. Furthermore, in simulated RL, where agents are trained in simulations before deployment in the real environment, the challenge of computational complexity in simulation is also present.

Moreover, the complexity of sequential decision-making is influenced not only by the environment but also by the agent's action space and the task horizon. A larger action space—i.e., having many possible actions—increases the computational effort required to identify optimal decisions. Tasks with long horizons similarly amplify complexity, as agents must reason further into the future, significantly complicating the search for optimal decisions. Thus, efficiently solving long-horizon tasks in complex environments remains an important open challenge, especially relevant since real-world applications often exhibit precisely these complexities.

A promising approach to addressing these challenges is the use of abstractions, which simplify complex problems by deliberately ignoring certain details. Specifically, abstractions in sequential decision-making can simplify the environment, reduce the complexity of the action space, or shorten the effective horizon of the task.

1.2.1 ACTION AND TEMPORAL ABSTRACTIONS

Two key forms of abstraction in sequential decision-making are action abstraction, which simplifies the set of available actions, and temporal abstraction, which simplifies decisions over longer time horizons.

Action abstractions address large action spaces by reducing the number of candidate actions an agent must consider, typically eliminating actions that are irrelevant, infeasible, or consistently inferior. A simple yet effective method is action elimination, where unnecessary actions are removed manually using domain knowledge [Kanervisto et al., 2020] or automatically via auxiliary environmental signals [Zahavy et al., 2018]. A practical example is invalid action masking [Huang and Ontañón, 2022], crucial in environments with numerous invalid actions.

Beyond "hard" elimination (completely removing actions), "soft" elimination methods reduce the influence of non-promising actions by assigning them lower priority during exploration and decision-making. Soft elimination often uses a prior policy to focus exploration on promising actions [Anthony et al., 2017, Rosin, 2011]. From a Bayesian viewpoint, this approach imposes a prior over policies [Doshi-velez et al., 2010, Levine, 2018, Wingate et al., 2011], effectively regularizing policy optimization and facilitating faster

convergence [Geist et al., 2019, Grill et al., 2020]. Thus, with a suitable prior, agents achieve more efficient planning and learning, requiring fewer samples and resources, leading to improved performance.

Temporal abstractions simplify the policy space by enabling agents to reason over temporally extended sequences of primitive actions rather than individual steps. This is particularly useful in long-horizon tasks, where step-by-step planning in large action spaces is impractical. For example, a robot navigating to a distant location can select high-level waypoints instead of controlling low-level motor commands at each step.

In the literature, temporally extended actions are called options [Sutton et al., 1999], low-level controllers [Heess et al., 2016, Nachum et al., 2018], workers [Vezhnevets et al., 2017], or skills [Eysenbach et al., 2018], reflecting different modeling frameworks. A prominent formalism, the options framework [Sutton et al., 1999], defines options as low-level policies executed over multiple timesteps with explicit termination conditions, enabling hierarchical reasoning. These options can be handcrafted by experts or learned from data [Achiam et al., 2018, Bagaria and Konidaris, 2019, Fox et al., 2017, Jinnai et al., 2019, Machado et al., 2017, Ramesh et al., 2019], an ongoing open challenge. Recent work in hierarchical RL demonstrates that temporal abstractions provide a strong inductive bias, enhancing end-to-end deep RL efficiency in long-horizon tasks [Bacon et al., 2017, Harb et al., 2018]. Moreover, temporal abstractions can serve as reusable knowledge in transfer learning, facilitating effective exploration and accelerated learning of new tasks [Eysenbach et al., 2018, Heess et al., 2016, Igl et al., 2020, Wulfmeier et al., 2020].

1.2.2 STATE ABSTRACTIONS

To address the challenge of large state spaces, researchers have explored state abstractions, which simplify the environment by filtering out irrelevant information from the state space. State abstractions can be implemented in various ways: by removing state variables in factored domains, creating partitions of the state space in tabular domains, or learning state representations in high-dimensional domains. Despite differences in implementation, all state abstractions share the core of grouping "similar" states to simplify the policy space.

A fundamental question is: which states should be grouped together? To address this, Li et al. [2006] proposed a unified framework of state abstractions for MDPs, defining a hierarchy of abstractions based on the properties they preserve:

- Model-irrelevance abstraction: States are grouped together if they lead to the same rewards and next abstract states for all actions.
- Q^Π -irrelevance abstraction: States are grouped together if they share the same Q value for all actions and policies to follow afterwards.
- Q^* -irrelevance abstraction: States are grouped together if they share the same optimal Q value for all actions.
- a^* -irrelevance abstraction: States are grouped together if they share an optimal action and the optimal Q value for this action.
- π^* -irrelevance abstraction: States are grouped together if they share an optimal action while their optimal Q values for this action not necessarily being the same.

Here, the Q-value (or action-value) of a state-action pair represents the long-term rewards an agent receives when taking an action in a state and then following a policy thereafter.

This abstraction hierarchy moves from the most restrictive (model-irrelevance) to the most permissive (π^* -irrelevance). As we go down the hierarchy, the abstractions rely less on full knowledge of the environment's dynamics (i.e., transition and reward functions) and more on value-based information, which is often harder to obtain or estimate accurately. This leads to a trade-off: more abstract models can improve learning and planning efficiency, but they may require access to difficult-to-compute information, such as the optimal policy or value function—what we later refer to as the "curse" of abstraction. Moreover, abstractions can be extended beyond states to include state-action pairs, as in the framework of MDP homomorphisms, which capture deeper symmetries and structural regularities in the environment [Ravindran and Barto, 2001, van der Pol et al., 2020a,b].

According to [Starre et al., 2023b], state abstractions are applied in three distinct settings, depending on the available information. First, when both the abstraction and the corresponding abstract model (including transitions and rewards) are available, planning and learning can occur entirely within the abstract model. This typically leads to significant computational and sample efficiency gains, particularly valuable for online planning and simulated RL [Buesing et al., 2018, Chitnis and Lozano-Pérez, 2020]. This approach directly relates to Chapters 3 and 4 of this thesis.

Second, when only a state abstraction (but no abstract model) is available, we can still enhance sample efficiency by applying the abstraction directly to data sampled from the original environment/model. The resulting "empirical abstract model" reduces the complexity of policy and value function learning. However, approximate abstractions may introduce non-Markovian dependencies in the data [Starre et al., 2023a]. This approach has proven particularly effective in Monte Carlo Tree Search (MCTS), reducing search-tree complexity and enhancing planning efficiency [Anand et al., 2015b, Bai et al., 2016, Hostetler et al., 2014, Jiang et al., 2014, Xu et al., 2023], and in abstracted RL, where agents learn policies from abstract observations [Abel et al., 2016].

Third, when neither the abstraction nor the abstract model is initially known, abstraction frameworks still guide representation learning in model-based RL (MBRL)—a hybrid planning and learning approach explored in Chapter 5 of this thesis. In MBRL, agents learn abstract models directly from environment interactions, offering greater sample efficiency than traditional model-free methods. However, accurately identifying abstract states without full access to the true environment model is challenging, creating a "chicken-and-egg" problem: defining abstractions typically requires prior knowledge (like optimal policies or Q-values), which itself must be learned from data.

Thus, most research on learning abstract models in MBRL has focused on the model-irrelevance abstraction, avoiding the complexity of predicting full states or observations—especially beneficial in high-dimensional, visual domains [Gelada et al., 2019, van der Pol et al., 2020a]. Recently, researchers have also explored value-equivalent models [Grimm et al., 2020, 2021, 2022], corresponding to Q^π -irrelevance abstraction. While theoretically more abstract and potentially more compact, these methods require accurate estimation of value functions from data. Notable successes in this direction include MuZero [Schrittwieser et al., 2020] and EfficientZero [Ye et al., 2021], demonstrating significant improvements in both performance and sample efficiency compared to model-free RL.

1.3 CONTRIBUTIONS OF THIS THESIS

The research in this thesis explores learned abstract models in the contexts of online planning and model-based reinforcement learning (MBRL). Specifically, we investigate how learned abstract models can accelerate search in online planning and examine to what extent they support effective planning and policy improvement in MBRL.

Main Research Question: How can learned abstract models improve the efficiency and effectiveness of online planning and model-based reinforcement learning?

Key Contributions:

- A novel two-phase method for online planning in large Factored POMDPs that combines exact local dynamics with a learned influence model, significantly improving simulation efficiency (Chapter 3).
- A self-improving simulator framework that learns abstract models online and dynamically balances learned and ground-truth dynamics for efficient planning (Chapter 4).
- A systematic analysis of MuZero’s learned model, showing its limitations for policy evaluation and highlighting the central role of the learned policy prior in supporting effective planning (Chapter 5).

1.3.1 LEARNED ABSTRACT MODELS FOR EFFICIENT ONLINE PLANNING

Chapters 3 and 4 address the problem of online planning in large, factored, partially observable environments. A key challenge in this setting is the high computational cost of simulating the full environment, which can limit practical applicability.

We leverage structural properties common in real-world domains (e.g., traffic networks, warehouses), where the agent interacts with a local environment but is influenced by a broader system. To exploit this, we apply Influence-Based Abstraction (IBA) [Oliehoek et al., 2021] to build an efficient abstract simulator — the influence-augmented local simulator (IALS). This simulator combines exact local dynamics with a learned model that approximates external influences.

Research Question 1 (Chapter 3): How can we learn and integrate an influence-augmented local simulator (IALS) to accelerate online planning?

- We propose a two-phase approach: an offline phase, where we train a deep recurrent neural network as an influence model from simulation data; and an online planning phase, where this influence model is combined with an exact local simulator to form the IALS.

- We demonstrate seamless integration of the IALS into the POMCP planning algorithm, improving computational efficiency by caching recurrent neural network states rather than reprocessing complete histories.

Research Question 2 (Chapter 3): How does planning with the learned IALS compare to planning with the original simulator?

- We empirically show that planning with the IALS significantly reduces computational costs. Under fixed simulation budgets, planning with the IALS outperforms planning with the original simulator, demonstrating greater efficiency despite approximations.

Research Question 3 (Chapter 3): How does environmental influence strength affect the performance of the learned IALS?

- We find that environments with weaker influence coupling result in smaller performance gaps between planning with the IALS and the true simulator. Stronger coupling environments demand more accurate influence modeling for high planning performance.

Despite its advantages, the two-phase approach has key drawbacks: a costly offline training phase, sensitivity to distribution shifts during planning, and risk associated with fully replacing the true simulator.

To overcome these issues, we introduce a self-improving simulator paradigm that learns the abstract model online and dynamically chooses between the abstract and original simulators during planning.

Research Question 4 (Chapter 4): How can we address the limitations of the two-phase approach through a new simulation paradigm?

- We introduce the self-improving simulator, which incrementally learns the abstract model during planning, eliminating the need for an offline training phase.
- We propose a simulator-selection mechanism that uses a UCB1-based strategy to dynamically switch between the abstract and ground-truth models based on estimated prediction accuracy.
- We demonstrate that this mechanism mitigates distribution shift, improves prediction accuracy, and reduces the risk of planning failures due to inaccurate modeling.

1.3.2 LEARNED ABSTRACT MODELS IN MODEL-BASED REINFORCEMENT LEARNING

1

Chapter 5 explores learned abstract models in the context of MuZero [Schrittwieser et al., 2020], which uses a value-equivalent model for sample-efficient reinforcement learning.

While MuZero has shown impressive empirical results, its learned model’s contribution to effective planning and policy improvement is not well understood.

Research Question 5 (Chapter 5): How does MuZero’s learned value-equivalent model support effective planning and policy improvement?

- We show that MuZero’s learned model struggles with accurate policy evaluation, particularly for policies outside its training distribution, limiting direct policy improvement through planning.
- By isolating the learned model from the learned value function, we find that effective planning in MuZero critically depends on the learned policy prior, which guides the search toward areas where the model is accurate, reducing accumulated prediction errors. This insight partially explains MuZero’s effectiveness despite inaccuracies in the learned model itself.

1.4 ADDITIONAL RESEARCH

In this section, we describe our research that extends or complements the core contributions of this thesis. As an overview, there are two main lines of work: one on applying influence-augmented local simulators to reinforcement learning settings and the other on applying sequential decision-making methods to AI competitions inspired by real-world applications.

1.4.1 SCALING UP DEEP RL WITH IALS

We begin by describing our work on applying influence-augmented local simulators to reinforcement learning (RL) settings, which is closely related to the research presented in Chapters 3 and 4. This line of investigation was led by our close colleague, Miguel Suau.

As discussed earlier in Section 1.2, complex systems pose significant computational challenges for both online planning and simulated RL, where policies are learned in a simulated environment before being deployed in the real world. These challenges are particularly pronounced in deep RL methods, which require large amounts of data to learn effective policies. To improve the feasibility of deep RL in cases where environment simulations are costly, we explore the use of influence-augmented local simulators as a surrogate for data generation and policy learning. Our role in this line of work primarily involved contributing to experimental design.

We investigated this approach in single-agent RL settings [Suau et al., 2022c], with experiments conducted in two weakly coupled environments: warehouse commissioning and traffic light control. Results demonstrate that using influence-augmented local simulators can significantly reduce the overall runtime needed to learn an effective policy compared to training with a global simulator.

Building on this work, we extended the approach to multi-agent systems [Suau et al., 2022a], where we train multiple agents in a networked environment. In this case, each agent operates with its own influence-augmented local simulator, running independently and in parallel. This decentralized approach avoids the need for centralized simulations traditionally required in multi-agent RL, thereby improving the scalability of deep RL methods. We highlight the advantages of this method by demonstrating that 100 agents can be trained to control a traffic network in under six hours, as opposed to more than ten days, when using a global simulator.

Additionally, inspired by the influence-based abstraction framework, we propose a new neural network architecture, influence-aware memory, for deep RL in partially observable Markov decision processes (POMDPs). This architecture shows improved runtime efficiency and final performance compared to standard recurrent neural networks [Suau et al., 2022b].

1.4.2 THE 2022 PATHWAYS TO NET ZERO RL CHALLENGE

The second line of work is applying sequential decision-making methods to AI competitions. This led to two studies [Chen et al., 2023, Zobernig et al., 2022] that summarized the overall results of the competitions, where our solutions were discussed and analyzed.

In the 2022 Pathways to Net Zero RL Challenge, we, alongside our colleague Aleksander Czechowski, were tasked with developing a policy for investing in three types of renewable

energy. The objective was to maximize long-term rewards, considering factors such as revenue, carbon emissions, and job creation over the period from 2031 to 2050.

Unlike other teams that primarily applied deep RL methods, we developed a lightweight sequential optimization algorithm called sequential golden section search. In our approach, we modeled the problem as an open-loop planning problem, where the agent selects a sequence of actions for the entire planning horizon without receiving feedback from the environment. The algorithm iteratively refines the action sequence step by step, using the golden section search method [Kiefer, 1953] to optimize each action while keeping the others fixed. We found that a single iteration of the search, performed in the backward direction, was sufficient to achieve strong performance.

Due to the simplicity and scalability of our approach compared to deep RL methods, we were able to achieve the highest score in the competition, using only a dozen CPUs and a few hours of search. Our method earned us joint first place in the competition and was summarized and analyzed in [Zobernig et al., 2022], which we coauthored.

1.4.3 THE SECOND NEURAL MMO CHALLENGE

In the second Neural MMO Challenge at IJCAI 2022, where we placed third, our team implemented a planning-based solution for controlling eight agents in a 128x128 multi-agent grid environment. This environment required agents to make movement and attack decisions to optimize a team score based on exploration, foraging, combat, and equipment. Unlike other teams using computationally expensive deep RL methods, we adopted a decentralized planning approach that allowed each agent to make independent decisions within an autoregressive framework, where prior decisions influenced subsequent ones. To manage the agents' coordination, a manually designed master policy dynamically assigned tasks based on current team needs, such as exploring the map or gathering resources.

Our solution included three key components: a simulation model, a preference model, and a planning algorithm. The simulation model predicted outcomes of possible actions using a simplified representation of the environment, with adversarial modeling for other player agents. The preference model evaluated trajectories based on criteria like resource collection and proximity to threats, using the concept of satisfying to ensure balanced decision-making in the presence of multiple objectives. Finally, our planning algorithm, primarily based on Dijkstra's algorithm [Dijkstra, 1959], reduced computational complexity by focusing on minimal paths, enhancing efficiency within time constraints. To address partial observability, we compiled a global map from past agent observations, allowing agents to extend their planning horizon and reducing issues from limited sight.

While our approach was successful, we identified limitations, particularly in the level of agent coordination and the manual effort required to refine the preference model. Future work could involve integrating learning with planning, allowing a reinforcement learning-based agent to set high-level goals while the planner handles low-level decisions, which could provide a more adaptive, end-to-end learning framework. This combination could potentially improve both the performance and scalability of the approach.

Our method, analyzed in [Chen et al., 2023], demonstrated a balance between computation and coordination, yet also highlighted opportunities to enhance automation and flexibility in multi-agent planning tasks.

1.4.4 OTHER COLLABORATIONS

In addition to our previously mentioned work, we collaborated with several bachelor's and master's students on their theses, as listed in the Appendix. A notable collaboration was with Daniele Foffano, a former master's student, on robust model-based reinforcement learning (MBRL) [Foffano et al., 2022]. In this work, we proposed Robust Ensemble Adversarial MBRL, a method in which both an ensemble of dynamics models and an adversarial model selector are learned. The policy is then trained in the worst-case model from the ensemble, as determined by the adversary. This approach encourages robustness by exposing the policy to the worst model dynamics. We served as daily supervisor for this project.

We also contributed to a project led by Maximilian Igl, which introduced a hierarchical multitask framework called Multitask Soft Option Learning (MSOL). MSOL is built upon the control as inference framework [Levine, 2018] and extends it to hierarchical reinforcement learning [Sutton et al., 1999]. The key idea is to distill shareable policy priors, called soft options, from a set of training tasks while allowing the agent to learn task-specific policy posteriors for each task. This approach stabilizes training in multi-task settings and accelerates the learning of new tasks by using the learned soft options as priors. Our role in the project primarily involved evaluating the effectiveness of transferring policy priors learned by MSOL from training environments to new, unseen environments.

2

BACKGROUND

"Scientia potentia est."

Francis Bacon

In this chapter, we provide the necessary background for the rest of the thesis. In Section 2.1, we introduce the mathematical models of sequential decision-making problems, including Markov Decision Processes (MDPs), Partially Observable Markov Decision Processes (POMDPs), and Factored POMDPs. In Section 2.2, we introduce the Influence-based Abstraction (IBA), an effective state abstraction method for factored POMDPs that we use in our research (Chapter 3 and Chapter 4). Then, in Section 2.3, we introduce online planning, a class of solution methods for sequential decision-making problems. Finally, in Section 2.4, we introduce Model-Based Reinforcement Learning (MBRL), a class of reinforcement learning methods that (learn and) use a model of the environment to improve the efficiency of learning policies.

2.1 SEQUENTIAL DECISION-MAKING PROBLEMS

In this section, we start by introducing Markov Decision Processes (MDPs), the fundamental mathematical model for sequential decision-making problems. We will simultaneously refer to an MDP as (a model of) a problem, a task, and an environment. Then, we introduce Partially Observable Markov Decision Processes (POMDPs), an important generalization of MDPs that account for agents' partial observability of environment states. Next, we introduce factored POMDPs, often used to model sequential decision-making problems in structured environments.

2.1.1 MARKOV DECISION PROCESSES

THE MODEL

A Markov decision process (MDP) [Bellman, 1957b] is a model that describes the interaction between a decision-making agent and an environment. Formally, a discounted infinite-horizon MDP is a 6-tuple $\mathcal{M} = (\mathcal{S}, \mathcal{A}, \mathcal{T}, \mathcal{R}, \mu, \gamma)$ where:

- \mathcal{S} represents the set of possible states of the environment,
- \mathcal{A} represents the set of actions that the agent can take in the environment,
- $\mathcal{T} : \mathcal{S} \times \mathcal{A} \rightarrow \Delta(\mathcal{S})$ represents the transition function,
- $\mathcal{R} : \mathcal{S} \times \mathcal{A} \rightarrow \mathbb{R}$ represents the reward function,
- $\mu \in \Delta(\mathcal{S})$ represents the initial state distribution,
- $\gamma \in [0, 1)$ represents the discount factor.

The agent interacts with the environment as follows. At the initial time step $t = 0$, an initial state of the environment is drawn from the initial state distribution, $S_0 \in \mu$. At each subsequent time step t , the agent observes the environment state S_t and selects an action A_t . This action causes the environment to transition to a new state $S_{t+1} \sim \mathcal{T}(\cdot | S_t, A_t)$, and the environment returns $R_t = \mathcal{R}(S_t, A_t)$.

The discount factor γ indicates the agent or user's preference for immediate versus future rewards. While γ is often considered as part of the problem description and it is necessary for bounding the sum of rewards in infinite-horizon tasks, it can be treated as a hyperparameter. As a hyperparameter, γ can be used to control the variance of values in planning [Jiang et al., 2015] and learning [Schulman et al., 2015], trading off bias and variance to improve performance with limited samples.

POLICIES AND VALUE FUNCTIONS

Let Π be the set of all policies, including those that are non-stationary and randomized. Given a policy $\pi \in \Pi$, the state-value function V measures the expected discounted sum of future rewards, or discounted return in short, from a state s by following π afterwards:

$$V^\pi(s) = \mathbb{E} \left[\sum_{k=0}^{\infty} \gamma^k R_{t+k} | S_t = s, \pi \right] \quad (2.1)$$

The action-value function Q measures the expected discounted return from a state s by taking an action a and following π afterwards:

$$Q^\pi(s, a) = \mathbb{E} \left[\sum_{t=0}^{\infty} \gamma^t R_{t+k} | S_t = s, A_t = a, \pi \right] \quad (2.2)$$

OPTIMALITY CRITERIA

In the discounted infinite-horizon setting, the goal of sequential decision making is to learn or plan a policy π that maximizes the expected discounted return,

$$\pi^* = \arg \max_{\pi \in \Pi} \mathbb{E}_{S_0 \sim \mu} [V^\pi(S_0)] \quad (2.3)$$

We refer interested readers to [Puterman, 1994] for other optimality criteria in this and other settings. In pursuit of the optimal policy π^* , the concepts of optimal value functions become relevant, which are defined for every state $s \in \mathcal{S}$ and action $a \in \mathcal{A}$ as below:

$$V^*(s) = \max_{\pi \in \Pi} V^\pi(s) \quad (2.4)$$

$$Q^*(s, a) = \max_{\pi \in \Pi} Q^\pi(s, a) \quad (2.5)$$

In the discounted infinite-horizon setting, there always exists a stationary and deterministic policy π that satisfies $V^\pi(s) = V^*(s)$ and $Q^\pi(s, a) = Q^*(s, a)$ for all states $s \in \mathcal{S}$ and actions $a \in \mathcal{A}$ [Puterman, 1994]. Consequently, when searching for optimal policies, it is sufficient to restrict ourselves to the space of deterministic stationary policies.

BELLMAN EQUATIONS

The Bellman consistency and optimality equations (terminologies from [Agarwal et al., 2019]) are fundamental properties of MDPs that are crucial for the evaluation of value functions and the search for optimal policies. Given a stationary policy $\pi : \mathcal{S} \rightarrow \Delta(\mathcal{A})$, where $\Delta(\mathcal{A})$ denotes the space of distributions over actions, the Bellman consistency equations are necessary conditions for the value functions of the policy π , which state that for all states $s \in \mathcal{S}$ and actions $a \in \mathcal{A}$:

$$V^\pi(s) = \mathbb{E}_{A \sim \pi(\cdot|s)} [Q^\pi(s, A)] \quad (2.6)$$

$$Q^\pi(s, a) = \mathcal{R}(s, a) + \gamma \mathbb{E}_{S' \sim \mathcal{T}(\cdot|s, a)} [V^\pi(S')] \quad (2.7)$$

This recursive form of value functions gives rise to the temporal difference learning methods [Sutton, 1988], a commonly used class of methods for estimating the value function of a policy. The Bellman optimality equations are necessary conditions for the optimal value functions, which state that for all states $s \in \mathcal{S}$ and actions $a \in \mathcal{A}$:

$$V^*(s) = \max_{a \in \mathcal{A}} Q^*(s, a) \quad (2.8)$$

$$Q^*(s, a) = \mathcal{R}(s, a) + \gamma \mathbb{E}_{S' \sim \mathcal{T}(\cdot|s, a)} \left[\max_{a' \in \mathcal{A}} Q^*(S', a') \right] \quad (2.9)$$

DETERMINISTIC MDPs

In deterministic MDPs, the next state s_{t+1} is solely determined by the current state s_t and action a_t with probability 1. In this setting, we will overload the notation \mathcal{T} for deterministic transitions, $s_{t+1} = \mathcal{T}(s_t, a_t)$. The initial state distribution can still be stochastic.

FINITE-HORIZON MDPs

Different from discounted infinite-horizon MDPs, finite-horizon MDPs do not require a discount factor in the model specification to bound the sum of rewards. Instead, it introduces an integer \mathcal{H} that defines the horizon of the task. In the most general case, the reward and transition functions in finite-horizon MDPs can both be dependent on time steps. Consequently, the Markovian state of finite-horizon MDPs does not only include the environment state but also the current time step. In this setting, the optimality criterion of sequential decision making is often the expected sum of rewards. Accordingly, to make optimal decisions, the agent needs to take into account the current time step, resulting in time-dependent policies, $\pi : S \times \{0, \dots, \mathcal{H} - 1\} \rightarrow \Delta(\mathcal{A})$, and time-dependent value functions:

$$V_t^\pi(s) = \mathbb{E} \left[\sum_{k=0}^{\mathcal{H}-t-1} \gamma^k R_{t+k} | S_t = s, \pi \right] \quad (2.10)$$

$$Q_t^\pi(s, a) = \mathbb{E} \left[\sum_{k=0}^{\mathcal{H}-t-1} \gamma^k R_{t+k} | S_t = s, A_t = a, \pi \right] \quad (2.11)$$

2.1.2 PARTIALLY OBSERVABLE MDPs

When the agent does not fully observe the environment state, the decision-making tasks are often formulated as partially observable MDPs (POMDPs) [Kaelbling et al., 1998]. Formally, a discounted infinite-horizon POMDP is a 8-tuple $\mathcal{M} = (S, \mathcal{A}, \Omega, \mathcal{T}, \mathcal{R}, \mathcal{O}, \mu, \gamma)$, which consists of an underlying MDP and an observation model. The underlying MDP describes the same agent-environment interaction as in MDPs where:

- S represents the set of possible states of the environment,
- \mathcal{A} represents the set of actions that the agent can take in the environment,
- $\mathcal{T} : S \times \mathcal{A} \rightarrow \Delta(S)$ represents the transition function,
- $\mathcal{R} : S \times \mathcal{A} \rightarrow \mathbb{R}$ represents the reward function,
- $\mu \in \Delta(S)$ represents the initial state distribution,
- $\gamma \in [0, 1)$ represents the discount factor.

The observation model describes how the agent perceives the environment where:

- Ω represents the set of possible observations,
- $\mathcal{O} : \mathcal{A} \times S \rightarrow \Delta(\Omega)$ represents the observation function.

In the general case of POMDPs, the agent does not directly observe the environment state but receives a noisy or partial observation of the environment after a transition, $O_{t+1} \sim \mathcal{O}(\cdot|A_t, S_{t+1})$. When the agent does observe the environment state, the POMDP simplifies to an MDP, making MDPs special cases of POMDPs.

In POMDPs, the agent's current observation O_t is not a sufficient statistic for the next environment state S_{t+1} and reward R_t . As such, in the most general case, the agent has to take into account the entire history of observations and actions $H_t = \{A_0, O_1, \dots, A_{t-1}, O_t\}$ to make optimal decisions. This motivates the consideration of history-dependent policies, which map a history h_t to a distribution over actions, and history-dependent value functions:

$$V^\pi(h_t) = \mathbb{E} \left[\sum_{k=0}^{\infty} \gamma^k R_{t+k} | H_t = h_t, \pi \right] \quad (2.12)$$

$$Q^\pi(h_t, a) = \mathbb{E} \left[\sum_{k=0}^{\infty} \gamma^k R_{t+k} | H_t = h_t, A_t = a, \pi \right] \quad (2.13)$$

2.1.3 FACTORED POMDPs

In many real-world environments, the state space is structured and can be decomposed into a finite set of state variables, or factors, that interact with each other in a modular way. Factored POMDPs [Hansen and Feng, 2000] exploit this structure for more concise representations of large POMDPs. They represent the environment state as a joint random variable over a set of state variables, $S = \{S^1, \dots, S^N\}$, such that each environment state corresponds to an assignment of these state variables. Within this framework, the transition, reward, and observation functions can be modeled by a Two-stage dynamic Bayesian network (2DBN) [Boutilier et al., 1999], which also integrates the agent's action, observation, and reward as random variables. This model capitalizes on the conditional independence among random variables to maintain the compactness of representations. As a result, they can model very large environments with thousands of state variables. However, the state space can still be too large for efficient simulation-based planning and learning, necessitating state abstraction methods in factored POMDPs.

2.2 INFLUENCE-BASED ABSTRACTION

The complex state space of the ground-truth model motivates the use of abstract models. Adhering to the principle that a model’s construction should be informed by its final use, one intuitive approach to identifying such abstract models is to ensure they behave identically to the original model within the context where they are used. In the context of a POMDP model, if we can identify another model that, for any action-observation history and action, produces the same expected rewards and distribution for the next observations, then these models can be used interchangeably in solution methods for POMDPs that focus on estimating values and computing optimal policies for histories. Crucially, these abstract models can employ state spaces that are different and potentially simpler than that of the original model while guaranteeing the optimality of planning and learning with them.

Influence-based abstraction (IBA) [Oliehoek et al., 2021] is one such approach for performing lossless state abstraction [Li et al., 2006] in factored POMDPs. Given a factored POMDP model, referred to here as the *global model*, IBA defines a so-called *influence-augmented local model* (IALM) that abstracts away state variables that do not directly affect the agent’s observations and rewards. In this thesis, we will demonstrate how to construct and use this model for efficient simulation in online planning (Chapters 3 and 4; He et al. [2020, 2022]). Additionally, our research beyond this thesis demonstrates how IBA can significantly scale up simulation-based reinforcement learning as well in both single [Suau et al., 2022c] and multi-agent [Suau et al., 2022a] settings.

In the following sections, we will formally define the influence-augmented local model and demonstrate its property as a lossless abstraction of the ground-truth model. We begin by categorizing the state variables of a factored POMDP into those that directly affect the agent’s observation and reward (local state variables) and those that do not (Section 2.2.1). Then, we define the local model that captures the agent’s observation and reward with only the local state variables (Section 2.2.2). Finally, we describe how to capture the influence of the remaining state variables on the local model with only local information, using a so-called *influence predictor* (Section 2.2.3). The resulting *influence-augmented local model*, which integrates the local model with an influence predictor, induces the same history MDP as the original model and is, as such, a lossless abstraction of it.

2.2.1 CATEGORIES OF STATE VARIABLES

Notations For ease of notation, we will override the notation, for example, S , to represent both a set of state variables, i.e., $S = \{S^1, S^2, \dots, S^N\}$, and the joint random variable over them, i.e., the global state. Moreover, we will use $S = \times_{S^i \in S} \text{dom}(S^i)$ to denote the space of this joint random variable. Finally, we will use superscript to denote the index of a state variable and subscript to denote the time step.

Assumptions Employing 2DBNs [Boutilier et al., 1999] to represent factored POMDPs implies a stationary structure among the state variables, which remains consistent across all time steps. In this thesis, we further assume that there is no intra-stage dependency among the state variables in the 2DBN representing the global model. However, techniques introduced in this thesis can be readily extended to the setting where intra-stage dependen-

cies are present at the expense of more complex notations and models. We refer interested readers to Oliehhoek et al. [2021] for how to deal with them in the IBA.

Assumption 1. *There is no intra-stage dependency among the state variables - a state variable at time step $t + 1$ can only depend on state variables at time step t and the action.*

Local state variables We start by defining the *local state variables* S^{local} as state variables that directly affect the agent's observation and reward ¹. Given that R_t depends on S_t and A_t , and O_{t+1} depends on A_t and S_{t+1} , a state variable is a local state variable if it directly affects R_t as part of S_t or if it directly affects O_{t+1} as part of S_{t+1} .

Definition 1. *Given a factored POMDP represented by a 2DBN with a finite set of state variables $S = \{S^1, S^2, \dots, S^N\}$, the local state variables are those that are direct parents of the observation and reward variables:*

$$S^{\text{local}} \stackrel{\text{def}}{=} \{S^i \in S \mid S_t^i \in \text{PA}(R_t)\} \cup \{S^i \in S \mid S_{t+1}^i \in \text{PA}(O_{t+1})\} \quad (2.14)$$

By their definition, the local state variables can be used to model the next observation and reward without other state variables. Consequently, if we have a transition function over the local states, we will have an alternative model to the original global model.

Influence source state variables However, the transitions of local state variables S^{local} cannot be solely determined by themselves as they may be affected by the rest of the state variables in the system, i.e., the non-local state variables $S^{\neg \text{local}}$.

Definition 2. *Given a factored POMDP represented by a 2DBN with a finite set of state variables $S = \{S^1, S^2, \dots, S^N\}$, the non-local state variables are:*

$$S^{\neg \text{local}} \stackrel{\text{def}}{=} S \setminus S^{\text{local}} \quad (2.15)$$

While it seems we need to keep track of the non-local state variables $S^{\neg \text{local}}$ to model the local state variables S^{local} , ending up modeling the entire system, the structure of the environment can be exploited again here. We split the set of non-local state variables into two disjoint subsets: the *influence source state variables* and the *auxiliary state variables*.

To capture the influence from the rest of the system on the local state variables, we define the *influence source state variables* as the non-local state variables that directly affect the local state variables. The name comes from the fact that the rest of the system can only influence the local state variables and, as such, the agent's observation and reward through the influence source state variables.

Definition 3. *Given a factored POMDP represented by a 2DBN with a finite set of state variables $S = \{S^1, S^2, \dots, S^N\}$, the influence source state variables are non-local state variables that are direct parents of the local state variables:*

$$S^{\text{src}} \stackrel{\text{def}}{=} \{S^i \in S^{\neg \text{local}} \mid \exists S^j \in S^{\text{local}} \ S_t^i \in \text{PA}(S_{t+1}^j)\} \quad (2.16)$$

¹Technically, the IBA only requires that all state variables directly affecting the agent's observation and reward are modeled, meaning that other state variables can also be local state variables. The definition here is for simplicity.

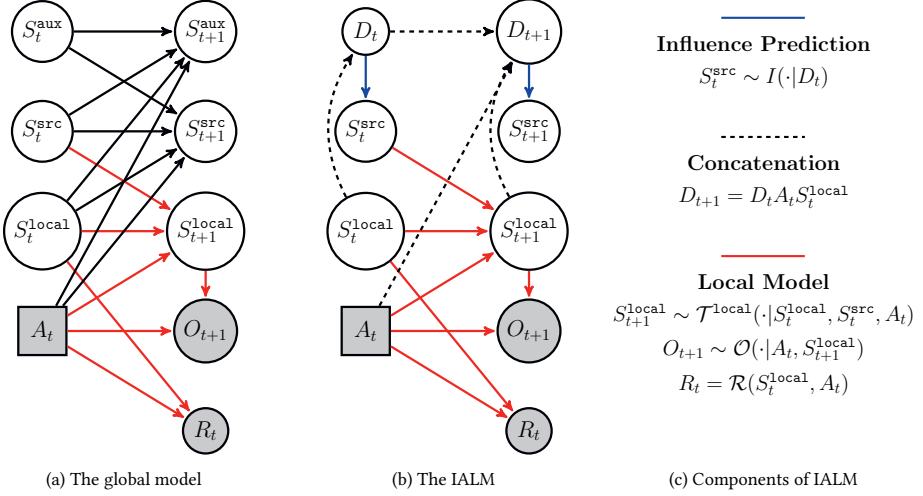


Figure 2.1: Two-stage dynamic Bayesian networks for the global model (a) and the influence-augmented local model (b). The global model and the IALM share the same local model that captures the transitions of local states and the agent’s observation and reward. However, the IALM abstracts away the auxiliary state variables and models the influence sources state differently from the global model.

The auxiliary state variables Finally, the *auxiliary state variables* are non-local state variables that do not directly affect the local state variables.

Definition 4. Given a factored POMDP represented by a 2DBN with a finite set of state variables $S = \{S^1, S^2, \dots, S^N\}$, the auxiliary state variables are non-local state variables that are not direct parents of the local state variables:

$$S^{\text{aux}} \stackrel{\text{def}}{=} \{S^i \in S^{\neg \text{local}} \mid \forall_{S^j \in S^{\text{local}}} S_t^i \notin \text{PA}(S_{t+1}^j)\} \quad (2.17)$$

We have now categorized the state variables into three subsets, $S = (S^{\text{local}}, S^{\neg \text{local}}) = (S^{\text{local}}, S^{\text{src}}, S^{\text{aux}})$, as pictured in Figure 2.1a. The name “auxiliary” comes from the fact that to model the agent’s observation and reward, one only needs to model the local and influence source state variables. Of course, to model the influence source state variables, one needs to model the auxiliary state variables. However, the IBA shows that the influence source state variables can also be modeled in a way that does not involve auxiliary state variables without losing the optimality of planning and, therefore, the name “auxiliary”.

Next, we will introduce the local model that captures the agent’s observation and reward with only the local state variables.

2.2.2 THE LOCAL TRANSITION, OBSERVATION AND REWARD FUNCTIONS

According to the definition of local state variables, we can model the agent’s observation and reward with only the local states and actions. This allows us to define a local observation function that conditions only on the local states and actions:

Definition 5. Given a factored POMDP represented by a 2DBN with a finite set of state variables $S = \{S^1, S^2, \dots, S^N\}$, the local observation function $\mathcal{O}^{\text{local}} : \mathcal{A} \times S^{\text{local}} \rightarrow \Delta(\Omega)$ maps an action A_t and a next local state S_{t+1}^{local} to a distribution over observations O_{t+1} , where $S^{\text{local}} = \times_{S^i \in S^{\text{local}}} \text{dom}(S^i)$ denotes the space of local states.

and similarly a local reward function:

Definition 6. Given a factored POMDP represented by a 2DBN with a finite set of state variables $S = \{S^1, S^2, \dots, S^N\}$, the local reward function $\mathcal{R}^{\text{local}} : S^{\text{local}} \times \mathcal{A} \rightarrow \mathbb{R}$ maps a local state S_t^{local} and an action A_t to a numerical reward R_t .

Then, according to the definition of influence source state variables, we can define a local transition function that predicts the next local states given the current local states, actions, and influence source states:

Definition 7. Given a factored POMDP represented by a 2DBN with a finite set of state variables $S = \{S^1, S^2, \dots, S^N\}$, the local transition function $\mathcal{T}^{\text{local}} : S^{\text{local}} \times S^{\text{src}} \times \mathcal{A} \rightarrow \Delta(S^{\text{local}})$ maps a local state S_t^{local} , an influence source state S_t^{src} and an action A_t to a distribution over next local states S_{t+1}^{local} .

These functions constitute the *local model* $\mathcal{M}_{\text{local}} = (\mathcal{T}^{\text{local}}, \mathcal{O}^{\text{local}}, \mathcal{R}^{\text{local}})$, a partial model of the local states and the agent's observation and reward with an external dependency on the influence source states.

2.2.3 INFLUENCE-AUGMENTED LOCAL MODEL

To construct a POMDP model that abstracts away state variables not directly affecting the agent's observations or rewards, we aim to track the influence source states using only local states and actions. We introduce the *influence-augmented local model* (IALM) — a reformulation of the original global model that preserves decision-relevant dynamics.

Every POMDP can be viewed as a history MDP, where the Markovian state at time step t is the action-observation history h_t . Two POMDPs with identical action and observation spaces induce the same history MDP if they agree on the expected reward $\mathcal{R}(h_t, a_t)$ and the distribution over next observations $\mathcal{T}(O_{t+1}|h_t, a_t)$ for all histories h_t and actions a_t .

We first show that $\mathcal{T}(O_{t+1}|h_t, a_t)$ can be expressed without auxiliary state variables via marginalization. Building on this, we define the IALM — a POMDP model that induces the same history MDP as the original model.

Given the global model $\mathcal{M}_{\text{global}} = (S, \mathcal{A}, \Omega, \mathcal{T}, \mathcal{R}, \mathcal{O}, \mu, \gamma)$, for every history h_t , action a_t , and next observation o_{t+1} :

$$\Pr(o_{t+1}|h_t, a_t; \mathcal{M}_{\text{global}}) \quad (2.18)$$

$$= \frac{\Pr(o_{t+1}, o_{0:t}|a_{0:t-1}, a_t; \mathcal{M}_{\text{global}})}{Z} \quad (2.19)$$

(conditional probability; where $Z = \Pr(o_{0:t}|a_{0:t}) = \Pr(o_{0:t}|a_{0:t-1})$ is the normalization constant)

$$= \frac{\sum_{s_{0:t+1}^{\text{local}}} \Pr(s_{0:t+1}^{\text{local}}, o_{0:t+1}|a_{0:t}; \mathcal{M}_{\text{global}})}{Z} \quad (2.20)$$

(law of total probability)

$$= \frac{\sum_{s_{0:t+1}^{\text{local}}} \Pr(s_{0:t+1}^{\text{local}}|a_{0:t}; \mathcal{M}_{\text{global}}) \left(\prod_{i=0}^t \mathcal{O}^{\text{local}}(o_{i+1}|a_i, s_{i+1}^{\text{local}}) \right)}{Z} \quad (2.21)$$

(observation is conditionally independent of other variables given the action and local state)

$$= \frac{\sum_{s_{0:t+1}^{\text{local}}} \Pr(s_0^{\text{local}}) \left(\prod_{i=0}^t \Pr(s_{i+1}^{\text{local}} | s_{0:i}^{\text{local}}, a_{0:i}; \mathcal{M}_{\text{global}}) \right) \left(\prod_{i=0}^t \mathcal{O}^{\text{local}}(o_{i+1} | a_i, s_{i+1}^{\text{local}}) \right)}{Z} \quad (2.22)$$

(chain rule)

$$= \frac{\sum_{s_{0:t+1}^{\text{local}}} \Pr(s_0^{\text{local}}) \left(\prod_{i=0}^t \Pr(s_{i+1}^{\text{local}} | s_{0:i}^{\text{local}}, a_{0:i}; \mathcal{M}_{\text{global}}) \right) \left(\prod_{i=0}^t \mathcal{O}^{\text{local}}(o_{i+1} | a_i, s_{i+1}^{\text{local}}) \right)}{Z} \quad (2.23)$$

(future actions cannot affect past local states)

$$= \frac{\sum_{s_{0:t+1}^{\text{local}}} \Pr(s_0^{\text{local}}) \left(\prod_{i=0}^t \Pr(s_{i+1}^{\text{local}} | s_{0:i}^{\text{local}}, a_{0:i-1}, a_i; \mathcal{M}_{\text{global}}) \mathcal{O}^{\text{local}}(o_{i+1} | a_i, s_{i+1}^{\text{local}}) \right)}{Z} \quad (2.24)$$

(reorganization)

$$= \frac{\sum_{s_{0:t+1}^{\text{local}}} \Pr(s_0^{\text{local}}) \left(\prod_{i=0}^t \sum_{s_i^{\text{src}}} \Pr(s_{i+1}^{\text{local}}, s_i^{\text{src}} | s_{0:i}^{\text{local}}, a_{0:i-1}, a_i; \mathcal{M}_{\text{global}}) \mathcal{O}^{\text{local}}(o_{i+1} | a_i, s_{i+1}^{\text{local}}) \right)}{Z} \quad (2.25)$$

(law of total probability)

$$= \frac{\sum_{s_{0:t+1}^{\text{local}}} \Pr(s_0^{\text{local}}) \left(\prod_{i=0}^t \sum_{s_i^{\text{src}}} \mathcal{T}^{\text{local}}(s_{i+1}^{\text{local}} | s_i^{\text{local}}, s_i^{\text{src}}, a_i) \Pr(s_i^{\text{src}} | s_{0:i}^{\text{local}}, a_{0:i-1}, a_i; \mathcal{M}_{\text{global}}) \mathcal{O}^{\text{local}}(o_{i+1} | a_i, s_{i+1}^{\text{local}}) \right)}{Z} \quad (2.26)$$

(conditional probability)

$$= \frac{\sum_{s_{0:t+1}^{\text{local}}} \Pr(s_0^{\text{local}}) \left(\prod_{i=0}^t \sum_{s_i^{\text{src}}} \mathcal{T}^{\text{local}}(s_{i+1}^{\text{local}} | s_i^{\text{local}}, s_i^{\text{src}}, a_i) \Pr(s_i^{\text{src}} | s_{0:i}^{\text{local}}, a_{0:i-1}; \mathcal{M}_{\text{global}}) \mathcal{O}^{\text{local}}(o_{i+1} | a_i, s_{i+1}^{\text{local}}) \right)}{Z} \quad (2.27)$$

(future actions cannot affect past local states)

$$= \frac{\sum_{s_{0:t+1}^{\text{local}}} \Pr(s_0^{\text{local}}) \left(\prod_{i=0}^t \mathbb{E}_{s_i^{\text{src}} \sim \Pr(\cdot | s_{0:i}^{\text{local}}, a_{0:i-1}; \mathcal{M}_{\text{global}})} [\mathcal{T}^{\text{local}}(s_{i+1}^{\text{local}} | s_i^{\text{local}}, s_i^{\text{src}}, a_i)] \mathcal{O}^{\text{local}}(o_{i+1} | a_i, s_{i+1}^{\text{local}}) \right)}{Z} \quad (2.28)$$

(rewrite into expectation)

where $\Pr(s_0^{\text{local}})$ is the initial distribution over the local state from the 2DBN.

Local history In the global model, the influence source state at time t is conditioned on the full previous state: $\Pr(s_t^{\text{src}} | s_{t-1}^{\text{local}}, s_{t-1}^{\text{src}}, s_{t-1}^{\text{aux}}, A_{t-1})$. After marginalizing out the auxiliary variables (as in Equation (2.28)), the resulting distribution depends on the full history of local states and actions: $\Pr(s_t^{\text{src}} | s_{t-1}^{\text{local}}, s_{t-1}^{\text{src}}, s_{t-1}^{\text{aux}}, A_{t-1})$. We refer to this sequence as the *local history*, denoted by $D_t = (s_{0:t}^{\text{local}}, A_{0:t-1})$, which excludes non-local state variables.

Influence predictor Below, we define an *influence predictor* that captures the conditional probability distribution in Equation (2.28) for time steps $t = 0, 1, \dots$ and show that it can be computed exactly from the global model.

Definition 8. Given a factored POMDP $\mathcal{M}_{\text{global}} = (S, \mathcal{A}, \Omega, \mathcal{T}, \mathcal{R}, \mathcal{O}, \mu, \gamma)$ represented by a 2DBN with a finite set of state variables $S = \{S^1, S^2, \dots, S^N\}$, the influence predictor is a sequence of conditional probability distributions over the influence source states, given the history of local states and actions:

$$I = (\Pr(s_t^{\text{src}} | D_t = (s_{0:t}^{\text{local}}, A_{0:t-1})); \mathcal{M}_{\text{global}})_{t \in \mathbb{N}} \quad (2.29)$$

where for every $t \in \mathbb{N}$, history of local states and actions $d_t = (s_0^{\text{local}}, a_0, s_1^{\text{local}}, \dots, s_t^{\text{local}})$, and influence source state s_t^{src} :

$$\Pr(s_t^{\text{src}} | d_t = (s_{0:t}^{\text{local}}, a_{0:t-1}); \mathcal{M}_{\text{global}}) \quad (2.30)$$

$$= \frac{\sum_{s_{0:t-1}^{\text{src}}, s_{0:t}^{\text{aux}}} \Pr(s_{0:t}^{\text{local}}, s_{0:t}^{\text{src}}, s_{0:t}^{\text{aux}} | a_{0:t-1}; \mathcal{M}_{\text{global}})}{Z} \quad (2.31)$$

$$= \frac{\sum_{s_{0:t-1}^{\text{src}}, s_{0:t}^{\text{aux}}} \mu(s_0^{\text{local}}, s_0^{\text{src}}, s_0^{\text{aux}}) \prod_{i=0}^{t-1} \mathcal{T}(s_{i+1}^{\text{local}}, s_{i+1}^{\text{src}}, s_{i+1}^{\text{aux}} | s_i^{\text{local}}, s_i^{\text{src}}, s_i^{\text{aux}}, a_i)}{Z} \quad (2.32)$$

where the denominator Z is a constant that makes $\Pr(s_t^{\text{src}} | d_t)$ a valid probability distribution.

Note that while in the general case the entire history of local states and actions may be needed for influence prediction, in many cases a sufficient statistic of this history, known as the D-set, exists and enables optimal prediction [Oliehoek et al., 2021]. This motivates the use of sequence models for this prediction.

The influence-augmented local model We have now shown the distribution over the next observation can be rewritten in a recursive form without the auxiliary state variables. Below, we will formally introduce the influence-augmented local model that integrates the local model with the influence predictor, as pictured in Figure 2.1b. The IALM employs the local model to capture the transition of local states and the agent’s observation and reward and the influence predictor to model the influence source states, making it a complete POMDP model.

Definition 9. Given a factored POMDP $\mathcal{M}_{\text{global}} = (S, \mathcal{A}, \Omega, \mathcal{T}, \mathcal{R}, \mathcal{O}, \mu, \gamma)$ represented by a 2DBN with a finite set of state variables $S = \{S^1, S^2, \dots, S^N\}$, the influence-augmented local model is a POMDP model that consists of a local model and an influence predictor $\mathcal{M}_{\text{IALM}} = (S^{\text{IALM}}, \mathcal{A}, \Omega, \mathcal{T}^{\text{IALM}}, \mathcal{R}^{\text{IALM}}, \mathcal{O}^{\text{IALM}}, \mu^{\text{IALM}}, \gamma)$ where:

1. \mathcal{A} and Ω are the action space and observation space.
2. S^{IALM} is the state space of the IALM. It contains not only the local state variables but also the past local states and actions as they are needed for the transition function. Technically, D_t contains S_t^{local} but we write that $S_t^{\text{IALM}} = (S_t^{\text{local}}, D_t)$ for ease of notation, where $D_t = D_{t-1} A_{t-1} S_t^{\text{local}} = S_{0:t}^{\text{local}} A_{0:t-1}$ is the history of local states and actions.
3. $\mathcal{R}^{\text{IALM}}$ is the reward function, where $\mathcal{R}^{\text{IALM}}(S_t^{\text{IALM}} = (S_t^{\text{local}}, D_t), A_t) = \mathcal{R}^{\text{local}}(S_t^{\text{local}}, A_t)$.
4. $\mathcal{O}^{\text{IALM}}$ is the observation function, where $\mathcal{O}^{\text{IALM}}(A_t, S_{t+1}^{\text{IALM}} = (S_{t+1}^{\text{local}}, D_{t+1})) = \mathcal{R}^{\text{local}}(A_t, S_{t+1}^{\text{local}})$.
5. $\mathcal{T}^{\text{IALM}}$ is the transition function of the IALM, made of the local transition function $\mathcal{T}^{\text{local}}$ and the influence predictor I , as described below.
6. $\mu^{\text{IALM}}(S_0^{\text{local}}) = \Pr(S_0^{\text{local}})$ is the initial distribution over the local state.

For every state of the IALM $(s_t^{\text{local}}, d_t)$, action a_t , and next state $(s_{t+1}^{\text{local}}, d_{t+1})$, we have that

$$\mathcal{T}^{\text{IALM}}(s_{t+1}^{\text{local}}, d_{t+1} | s_t^{\text{local}}, d_t, a_t) = \sum_{s_t^{\text{src}}} I(s_t^{\text{src}} | d_t) \mathcal{T}^{\text{local}}(s_{t+1}^{\text{local}} | s_t^{\text{local}}, s_t^{\text{src}}, a_t) \mathbb{1}(d_{t+1} = d_t a_t s_{t+1}^{\text{local}}) \quad (2.33)$$

2.2.4 REMARKS

2

The IALM is a lossless abstraction to the global model To demonstrate that the IALM is a lossless abstraction to the global model, we can show that they induce the same history MDP. That is, they give the same expected reward for all histories and actions and the same distribution over the next observation. This guarantees that planning with either model would lead to the same optimal policies and values. The original result, proven in a more general setting, can be found in [Oliehoek et al., 2021] (Section 6, Theorem 1).

Proposition 1. *The influence-augmented local model $\mathcal{M}_{\text{IALM}}$ constructed from the global model $\mathcal{M}_{\text{global}}$ of a factored POMDP defines the same history MDP as the global model.*

Proof Sketch. By construction, the IALM induces the same distribution over the next observation $\Pr(O_{t+1}|h_t, a_t)$ as the original model, given any history h_t and action a_t .

In a very similar way, we can prove that the IALM and the global model induce the same distribution over the current local state $\Pr(S_t^{\text{local}}|h_t)$. This allows us to show that both models induce the same expected reward as for both models, we have $\mathcal{R}(h_t, a_t) = \mathbb{E}_{S_t^{\text{local}} \sim \Pr(\cdot|h_t)} [\mathcal{R}^{\text{local}}(S_t^{\text{local}}, a_t)]$. \square

Computing the influence predictor exactly We have shown above how to compute the influence predictor exactly via marginalization. However, doing that for T steps has a time and space complexity that is exponential in T in the worst case, which makes it computationally infeasible in practice. We will discuss how to address this challenge in Chapter 3.

When is the IBA most beneficial? We can construct an influence-augmented local model from any factored POMDP represented by a 2DBN. The advantage is that we can abstract all the auxiliary state variables away from the state space and simulations. As such, the IBA is most effective when the number of auxiliary state variables far exceeds the number of local state variables, which happens in scenarios where the environment is huge, but the agent directly interacts with a smart part of it. An example would be controlling a robot in a very large warehouse to perform some local tasks.

Why not directly model the next local state? The IALM predicts the distribution over the next local state by first predicting the distribution over the current influence source state and then using the local model to obtain the distribution over the next local state. Alternatively, one could directly use a model for local states $\Pr(S_{t+1}^{\text{local}}|S_{0:t}^{\text{local}}, A_{0:t})$ or observations $\Pr(O_{t+1}|O_{0:t}, A_{0:t})$ to obtain the same history MDP. However, we choose to model the influence source state first for two reasons. First, in many structured environments, an agent’s local state variables are directly influenced by only a few influence source state variables. This makes predicting the influence source state simpler and more efficient than predicting the next local state. Second, directly modeling $\Pr(S_{t+1}^{\text{local}}|S_{0:t}^{\text{local}}, A_{0:t})$ or $\Pr(O_{t+1}|O_{0:t}, A_{0:t})$ requires building a model from scratch, involving either exact inference or learning. In contrast, modeling the influence source state allows us to leverage the exact local model that can be extracted from the 2DBN, resulting in a hybrid model where only the part of the influence predictor is approximated.

2.3 SAMPLE-BASED ONLINE PLANNING

Many real-world decision-making tasks are so complex that finding a policy that performs well in all situations is infeasible. In such cases, planning methods that use online computation to find a good action for the current state become more practical. These methods, referred to as online or decision-time planning methods, differ from background planning methods like value iteration, which aim to find an optimal action or policy for all states.

Decision-time planning methods use online computation to predict future outcomes from the current state s_t using an environment model, and then select actions based on their estimated long-term utilities. While it is theoretically possible to run a global or local dynamic programming algorithm to find the optimal action at decision time, this is usually computationally infeasible for large problems. Moreover, these exact planning algorithms cannot provide results at any time, making them unsuitable for real-time decision-making. Therefore, practical decision-time planning methods are often iterative algorithms, capable of returning a good action at any point during computation.

The iterative nature of decision-time planning methods makes them similar to search algorithms, where the objective is to find the action that maximizes the local state-action value, $a^* = \arg \max_{a \in \mathcal{A}} Q^*(s_t, a)$. Like search algorithms, decision-time planning algorithms face the challenge of efficiently navigating the search space, specifically deciding which actions to consider and evaluate during the search process. The literature has explored various approaches, including random search, heuristic search, and tree search, which are different strategies for guiding the search using previous results—akin to the exploration challenge in reinforcement learning.

Another critical challenge for efficient online planning is evaluating the long-term utilities of actions. Given the preference for anytime algorithms, decision-time planning methods often rely on sampling-based techniques to estimate these utilities. Monte-Carlo rollout methods, due to their simplicity, are frequently used to simulate trajectories from the current state to the end of the episode, thereby estimating action values.

In this thesis, we extensively use Monte-Carlo Tree Search (MCTS), a highly flexible and scalable sample-based online planning method that has been successfully applied to a wide range of decision-making problems, including games, robotics, and recommendation systems. In the following section, we will introduce MCTS and then describe POMCP, a variant of MCTS adapted for partially observable domains.

2.3.1 MONTE-CARLO TREE SEARCH

MCTS is an online planning algorithm that uses rollouts to estimate the value of states and actions. Making use of a generative simulator \mathcal{G} , MCTS incrementally builds a search tree by simulating trajectories from the root node to the leaf nodes following a tree policy. In this search tree, each node represents a state, which stores statistics such as visit count and average return, and each edge represents an action (and the corresponding transition). The tree policy is often designed to balance exploration and exploitation explicitly, for example, by using the UCB1 algorithm [Auer et al., 2002] as in upper confidence tree search (UCT) [Kocsis and Szepesvári, 2006]. At leaf nodes, the algorithm uses a rollout policy, such as a random policy, to quickly estimate the node’s value by simulating a trajectory from it

to the end of the episode. The simulation results are then back-propagated to update the statistics of the nodes in the tree, which determine the tree policy in the next iteration. At the end of each iteration, a new node is added to the tree to represent the first newly encountered state-action pair. When the search budget is up, the algorithm returns either the action that maximizes the average return (the max child) or the action that has been visited the most (the robust child).

In essence, MCTS is a sample-efficient method for approximating the local optimal state-action value function $\hat{Q}^*(s_t, \cdot)$. It achieves this by (1) focusing the online computation on the current state s_t , (2) effectively balancing exploration and exploitation when selecting actions for simulation, and (3) efficiently backing up the simulation results to the tree nodes to direct future search. Using search algorithm terminology, MCTS implements a form of best-first search, where the search is directed towards the most promising branches based on the current search results.

2.3.2 MONTE-CARLO PLANNING IN LARGE POMDPs

Classical POMDP techniques [Spaan, 2012] scale moderately as they explicitly represent the exact beliefs over states. To improve scalability, Silver and Veness [2010] adapted MCTS to POMDPs and proposed POMCP, an online planning method for large POMDPs where particle filtering [Doucet and Johansen, 2009] is applied to approximate the belief.

POMCP inherits the basic structure of MCTS but is adapted to the partially observable setting. The main difference between planning in MDPs and POMDPs is that in POMDPs, the agent needs to make decisions based on the entire history of past observations and actions h_t instead of the current observation o_t . Ideally, if we have access to a generative simulator of histories that can simulate a possible next observation and reward, given the current history and action, we can apply MCTS to POMDPs straightforwardly. However, in practice, we often do not have access to such a simulator, and the histories are too large to be stored and simulated explicitly. POMCP addresses this challenge by using a generative simulator of states and approximating the simulation of histories with the simulation of states. Specifically, POMCP uses a particle filter to approximate the belief over states, given the current history, and simulates trajectories from particles sampled from the belief, each of which represents a possible state of the environment. The use of states in simulation is implicit in POMCP, as the nodes in the search tree still represent histories and tree policy still conditions on the statistics of the history nodes. That means the way that POMCP selects actions to simulate within the tree and the way that it updates and expands the tree nodes are the same as in MCTS. POMCP has been shown effective in scaling up planning in large POMDPs due to the sample efficiency of MCTS and the scalability of particle filters in approximating the belief over states.

2.4 MODEL-BASED REINFORCEMENT LEARNING

In the RL literature, the term model-based reinforcement learning (MBRL) [Moerland et al., 2023] has been used in various contexts, often without a precise definition. In recent years, however, MBRL has come to more specifically refer to the paradigm in which an agent

learns a model of the environment and uses it to make decisions while interacting with the environment.

The promise behind MBRL in comparison to model-free RL is that by learning a model of the environment, the agent can further improve its policy or value functions by planning with the model without additional interactions with the environment, a process known as model-based credit assignment [van Hasselt et al., 2019].

Apart from the potentially improved sample efficiency, another hypothesized benefit of MBRL is more efficient exploration. Model-based exploration can leverage the learned model by using model uncertainty to guide exploration and by planning with the model to identify actions that will enable deeper exploration. This approach, often referred to as directed exploration, has been investigated in Bayes-adaptive model-based RL [Duff, 2002, Ghavamzadeh et al., 2015, Guez et al., 2012, Katt et al., 2017, Ross et al., 2007, 2011, Shyam et al., 2019] as well as in deep RL settings [Brafman and Tennenholtz, 2002, Henaff, 2019, Lowrey et al., 2018, Pathak et al., 2017, 2019, Sekar et al., 2020, Shyam et al., 2019].

Model-based RL has been an active research area since the early days of reinforcement learning [Brafman and Tennenholtz, 2002, Deisenroth and Rasmussen, 2011, Moore and Atkeson, 1993, Sutton, 1991, Sutton and Barto, 2018]. Many MBRL frameworks in the literature exist, differing in a diverse range of design choices. At a high level, these design choices concern the following aspects: (1) what model to learn, (2) how to learn the model, and (3) how to use the model for decision-making. For example, on the last aspect, once a model is learned, one can use offline planning methods, such as deep reinforcement learning, or online planning methods, such as MCTS, to find a policy that maximizes the expected return. Furthermore, when a model is parameterized by a neural network and learned from data, it is differentiable, in which case the agent can also use gradient-based optimization to directly find a good action or policy for the current state.

In the following, we will introduce MuZero [Schrittwieser et al., 2020], an MBRL algorithm that has achieved state-of-the-art performance across multiple benchmarks and a significant impact on real-world applications. We will begin by discussing AlphaZero, MuZero’s predecessor, along with the framework of expert iteration [Anthony et al., 2017], which forms the foundation of both AlphaZero and MuZero. Next, we will introduce the value-equivalence principle, a line of theoretical work that connects MuZero’s empirical success to the state abstraction theory. Finally, we will describe MuZero itself.

2.4.1 ALPHAZERO AND EXPERT ITERATION

The development of this family of algorithms started with AlphaGo [Silver et al., 2016], a computer program designed to play the board game Go. AlphaGo used deep neural networks to guide Monte-Carlo Tree Search (MCTS): a value network estimated state values, and a policy network guided action selection during simulation. Unlike classical MCTS, AlphaGo’s MCTS replaced rollouts with value function estimates at leaf nodes and incorporated the policy network (the prior policy) to bias action selection toward promising moves. These modifications served two purposes: (1) bootstrapping from a value network reduced variance and improved value estimates, and (2) using a prior policy accelerated search by focusing on high-potential actions.

AlphaGo’s networks were first trained via supervised learning on expert games, fol-

lowed by reinforcement learning through self-play, where the networks were updated using MCTS-generated targets. This approach led to AlphaGo defeating world champion Lee Sedol in 2016—one of the first major successes in deep reinforcement learning.

AlphaGo Zero [Silver et al., 2017b] improved upon AlphaGo by removing human data and learning entirely from self-play. This made the system more general and improved its performance—after just three days of training, AlphaGo Zero decisively beat AlphaGo.

AlphaZero [Silver et al., 2018] generalized this approach beyond Go to other board games like Chess and Shogi, achieving superhuman performance from scratch in each domain. Although AlphaZero was designed for two-player games, its architecture extends naturally to single-agent decision-making problems, paving the way for MuZero.

While AlphaGo and AlphaZero’s integration of deep learning and MCTS may appear ad hoc, the expert iteration framework [Anthony et al., 2017] provides a conceptual foundation. This framework aligns with dual-process theories of reasoning [Evans, 2008, Gershman, 2017, Kahneman, 2011, Stanovich, 2011], which propose two cognitive systems: a fast, intuitive one (System 1) and a slow, deliberative one (System 2). In this analogy, neural networks serve as the fast, approximate component, while MCTS provides slow, exact deliberation. Crucially, the fast system accelerates the slow one, and is in turn trained by imitating its outputs—a process called expert iteration. This framework is general: MCTS can be replaced by other planning methods (e.g., model predictive control), and neural networks by other function approximators (e.g., decision trees). Even what constitutes the “fast system” can vary—for example, it could represent high-level skills in a hierarchical agent [Young and Sutton, 2023].

Grill et al. [2020] analyzed the role of MCTS through the lens of policy improvement, showing that it approximately solves a regularized policy optimization problem at decision time. The strength of this regularization—favoring the policy prior—diminishes with more simulations. Thus, MCTS acts as a policy improvement operator: it refines the learned policy during planning, producing better actions for execution and more informative targets for learning [Grill et al., 2020, Hamrick et al., 2022]. This interaction between planning and learning accelerates the learning of the policy [Bertsekas, 2022, Hamrick et al., 2022].

2.4.2 THE VALUE-EQUIVALENCE PRINCIPLE

The value equivalence principle for model-based reinforcement learning, introduced by Grimm et al. [2020, 2021, 2022], is motivated by the consideration that the construction of a model should take into account the final use of the model. It defines the order- k value equivalence class $\mathcal{M}^k(\Pi, \mathcal{V})$ as the subset of all models that can predict the correct k -step bellman update for any $\pi \in \Pi$ and $v \in \mathcal{V}$ in a set of policies Π and functions \mathcal{V} . For $k \rightarrow \infty$, an additional proper value equivalence class $\mathcal{M}^\infty(\Pi)$ is defined, which excludes the set of functions from the specification. In essence, a model in the proper value equivalence class must have the true value function as the fixed point of the bellman operator for all policies in this set. As such, these models can be seen as the Q^Π -irrelevance abstractions for the MDP [Li et al., 2006]. Notably, the largest proper value equivalence class that guarantees optimal planning is $\mathcal{M}^\infty(\Pi^{\text{DET}})$, where Π^{DET} is the set of all deterministic policies. Moreover, Grimm et al. [2021] showed that a simplified version of MuZero’s loss upper bounds the proper value equivalence loss, making an explicit connection between

the theory of value equivalence principle and the strong empirical performance of MuZero.

2.4.3 MUZERO

MuZero [Schrittwieser et al., 2020] is a model-based reinforcement learning algorithm that inherits most of its structure from AlphaZero. The key difference between MuZero and AlphaZero is that AlphaZero has access to an exact model of the environment, while MuZero learns a model of the environment from data. MuZero gained its popularity by achieving state-of-the-art performance in Atari games and matching the superhuman performance of AlphaZero [Silver et al., 2018] in Go, chess, and shogi.

Components MuZero learns a deterministic world model that consists of a representation function h_θ and a dynamics function g_θ . The representation function encodes an environment state s_t into a latent state $z_t^0 = h_\theta(s_t)$. Here, the subscript denotes the time step in the environment at which the encoding occurs, and the superscript denotes the number of time steps that have been spent in the learned model since then. Given a latent state z_t^k and an action a_{t+k} , the dynamics function predicts the next latent state z_t^{k+1} and the reward u_t^k , $(z_t^{k+1}, u_t^k) = g_\theta(z_t^k, a_{t+k})$. Apart from the representation and dynamics functions, MuZero uses a prediction function to predict the value and policy at a latent state, $\pi_t^k, v_t^k = f_\theta(z_t^k)$. For ease of notation, we split the prediction function into a policy function $\pi_\theta(z_t^k)$ and a value function $v_\theta(z_t^k)$. To distinguish between the policy function π_θ and the MuZero policy π^{MuZero} , which runs MCTS, we will refer to the former as the policy prior and the latter as MuZero’s behavior policy.

Acting MuZero makes decisions by planning with the learned model. At each time step t , MuZero encodes the environment state s_t into a latent state z_t^0 and uses it as the root node to perform MCTS. As the result of the search, MuZero selects the action a_t by sampling from a distribution that is constructed using the visit counts at the root node and a temperature parameter $T \in (0, \infty)$:

$$\pi^{\text{MuZero}}(a|s_t) = \frac{N(z_t^0, a)^{1/T}}{\sum_b N(z_t^0, b)^{1/T}} \quad (2.34)$$

MuZero’s planning differs from traditional MCTS methods like UCT [Kocsis and Szepesvári, 2006] in two key ways. First, rather than employing random rollouts for leaf node value estimation, MuZero uses its learned value function v_θ to produce potentially more informed value estimates. Second, MuZero incorporates the policy prior into its action selection, guiding the simulation of actions at tree nodes towards more promising candidates:

$$\arg \max_a \left[Q(z, a) + c \cdot \pi_\theta(a|z) \cdot \frac{\sqrt{\sum_b N(z, b)}}{1 + N(z, a)} \right] \quad (2.35)$$

where $c = c_1 + \log(\frac{\sum_b N(z, b) + c_2 + 1}{c_2})$ with $c_1 = 1.25$ and $c_2 = 19652$. Q and N are the estimated values and visit counts of actions. MuZero inherits much of its search mechanism from AlphaZero, including the use of policy prior and value function.

Training The key difference between MuZero and many prior works in MBRL lies in their approach to learning the model. In Figure 2.2, we illustrate the loss function of MuZero. Essentially, given a segment of a real episode that starts from state s_t , MuZero

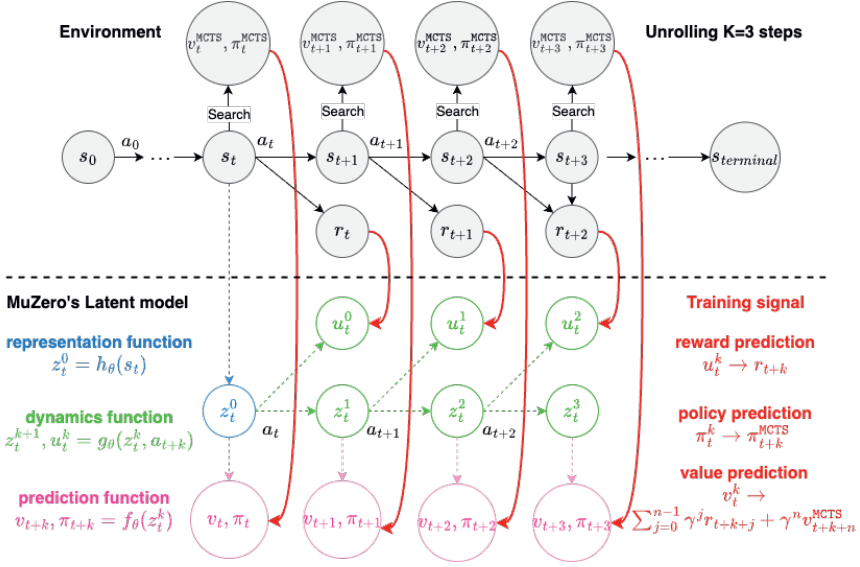


Figure 2.2: An illustration of MuZero's loss function.

unrolls its model for K simulated time steps (below the dotted line) and compares these to real experiences (above the dotted line). This comparison results in a loss consisting of three terms for each of the K steps. The first term is the per-step reward prediction loss, with the target being the real reward received. The second term is the policy prediction loss, with the target being the action visit distribution of the MCTS at the root node. The third term is the value prediction loss, with the target being the discounted sum of n -step real rewards plus a value estimation bootstrapped from MCTS n steps into the future $v_t^{\text{target}} = \sum_{k=0}^{n-1} \gamma^k r_{t+k} + \gamma^n v_t^{\text{MCTS}}$. For simplicity, we omit the links for value targets in Figure 2.2. All components of MuZero are trained jointly end-to-end by minimizing the aggregated loss. In practice, MuZero incorporates various additional techniques from the literature, such as prioritized experience replay, to improve training. It also introduces a novel algorithm called ‘reanalyse’ to generate fresh targets from old trajectories by re-running MCTS on them using the latest network.

3

3

LEARNING AND PLANNING WITH INFLUENCE-AUGMENTED LOCAL SIMULATORS

"The body is a cage for the mind, and the world is a cage for the body."

Lord of Mysteries

How can we plan efficiently in real time to control an agent in a complex environment that may involve many other agents? While existing sample-based planners have enjoyed empirical success in large POMDPs, their performance heavily relies on a fast simulator. However, real-world scenarios are complex in nature and their simulators are often computationally demanding, which severely limits the performance of online planners. In this work, we propose influence-augmented online planning, a principled method to transform a factored simulator of the entire environment into a local simulator that samples only the state variables that are most relevant to the observation and reward of the planning agent and captures the incoming influence from the rest of the environment using machine learning methods. Our main experimental results show that planning with this less accurate but much faster local simulator with POMCP leads to higher real-time planning performance than planning with the simulator that models the entire environment.

3.1 INTRODUCTION

We consider the online planning setting where we control an agent in a complex environment that is partially observable and may involve many other agents. When other agents' policies are known, the entire environment can be modeled as a Partially Observable Markov Decision Process (POMDP) [Kaelbling et al., 1998], and classical online planning approaches can be applied. While sample-based planners like POMCP [Silver and Veness, 2010] have been shown effective for large POMDPs, their performance relies heavily on a fast simulator to perform a vast number of Monte Carlo simulations in a step. However, many real-world scenarios are complex in nature, making simulators that capture the dynamics of the entire environment extremely computationally demanding and hence preventing existing planners from being useful in practice. Towards effective planning in realistic scenarios, this work is motivated by the question: can we significantly speed up a simulator by replacing the part of the environment that is less important with an approximate learned model?

We build on the multi-agent decision-making literature that tries to identify compact representations of complex environments for an agent to make optimal decisions [Becker et al., 2003, 2004, Petrik and Zilberstein, 2009, Witwicki and Durfee, 2010]. These methods exploit the fact that in many structured domains, only a small set of (state) variables, which we call *local (state) variables*, of the environment directly affects the observation and reward of the agent. The rest of the environment can only impact the agent indirectly through their influence on the local state variables. For example, Figure 3.1 shows a game called Grab A Chair, in which there are N agents that, at every time step, need to decide whether they will try to grab the chair on their left or right side. An agent can only secure a chair if that chair is not targeted by the other neighboring agent. At the end of every step, each agent only observes whether it obtains the chair without knowing the decisions of others. Additionally, there is a noise on observation, i.e., a chance that the agent gets an incorrect observation. In this game, it is clear that to the planning agent, whose goal is to obtain a chair in as many steps as possible, the decisions of neighboring agents 2 and 5 are more important than those of agents 3 and 4, as the former directly determine if the planning agent can secure a chair. In other words, only agents 2 and 5 directly *influence* agent 1's local decision making, while agents 3 and 4 may only do so indirectly.

To utilize this structure, we propose influence-augmented online planning, a principled method based on the influence-based abstraction (IBA) (Chapter 2.2; Oliehoek et al. [2021]) that transforms a factored simulator of the environment, called *global simulator*, into a faster *influence-augmented local simulator (IALS)*. The IALS simulates only the local state variables and concisely captures the influence of the other state variables by predicting only the subset of them, called *influence source state variables* that directly affect the local state variables. Using off-the-shelf supervised learning methods, the influence predictor is learned offline with data collected from the global simulator. We posit that when planning with sample-based planners, the advantage of substantially more simulations in the IALS may outweigh the simulation inaccuracy caused by approximating the incoming influence.

In this work, we investigate this hypothesis and show that this approach can indeed improve online planning performance. More specifically, our planning experiments with POMCP show that, by replacing the global simulator with an IALS that learns the incoming

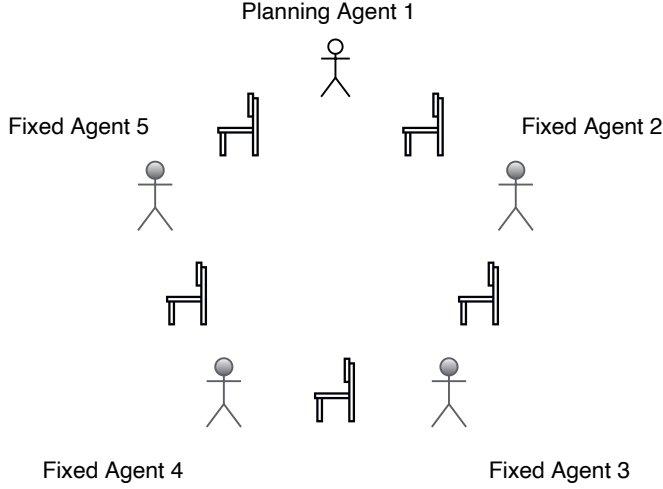


Figure 3.1: Controlling a single agent in the Grab A Chair game with 4 other agents.

influence with a recurrent neural network (RNN), we achieve matching performance while using much less time. More importantly, our real-time online planning experiments show that planning with the less accurate but much faster IALS yields better performance than planning with the global simulator in a complex environment when the planning time per step is constrained. In addition, we find that learning an accurate influence predictor is more important for good performance when the local planning problem is tightly coupled with the rest of the environment.

3.2 INFLUENCE-AUGMENTED LOCAL SIMULATORS

While the influence-based abstraction (Chapter 2.2; Oliehoek et al. [2021]) results in an IALM $\mathcal{M}_{\text{IALM}}$ that abstracts away non-local state variables S^{local} in a lossless way, it is not useful in practice because computing the distribution $I(S_t^{\text{src}}|D_t)$ exactly is in general intractable. Our approach trades off between the time spent before and during the online planning, by approximating $I(S_t^{\text{src}}|D_t)$ with a function approximator \hat{I}_θ learned offline. The learned influence predictor \hat{I}_θ will then be integrated with an accurate local simulator $\mathcal{G}_{\text{local}}$ to construct an influence-augmented local simulator (IALS) that only simulates the local state variables S^{local} but concisely captures the influence of the non-local state variables S^{local} by predicting the influence source state variables S^{src} with \hat{I}_θ . During the online planning, the integrated IALS will be used to replace the accurate but slow global simulator to speed up the simulations for the sample-based online planners. See Table 3.1 for a comparison between the global simulator and the IALS.

Our motivation is that by simulating the local transitions that directly decide the observation and reward of the agent with an accurate local simulator, the simulation inaccuracy caused by approximating the distribution $I(S_t^{\text{src}}|D_t)$ with \hat{I}_θ can be overcome

Simulator & State	Feature	Simulation
the global simulator $\mathcal{G}_{\text{global}}$ $s_t = (s_t^{\text{local}}, s_t^{\text{src}}, s_t^{\text{aux}})$	accurate & slow	$s_{t+1}, o_{t+1}, r_t \sim \mathcal{G}_{\text{global}}(s_t, a_t)$
the IALS $\mathcal{G}_{\text{IALS}}^\theta = (\mathcal{G}_{\text{local}}, \hat{I}_\theta)$ $s_t^{\text{IALS}} = (s_t^{\text{local}}, d_t)$	approximate & fast	$s_t^{\text{src}} \sim \hat{I}_\theta(\cdot d_t)$ $s_{t+1}^{\text{local}}, o_{t+1}, r_t \sim \mathcal{G}_{\text{local}}(s_t^{\text{local}}, s_t^{\text{src}}, a_t)$ $d_{t+1} = d_t a_t s_{t+1}^{\text{local}}$

Table 3.1: Comparison between the global simulator and the influence-augmented local simulator.

3

by the advantage that simulations can be performed significantly faster in the IALS, which is essential to sample-based planners like POMCP [Silver and Veness, 2010], leading to improved online planning performance in realistic scenarios with limited planning time. Our overall approach, influence-augmented online planning, is presented in Algorithm 1, followed by our method to learn an approximate influence predictor with recurrent neural networks (RNNs) [Cho et al., 2014, Hochreiter and Schmidhuber, 1997] and integrate it with a local simulator to form a plannable IALS for sample-based planners.

Algorithm 1: Influence-Augmented Online Planning

input: a real environment `env`

input: a global simulator $\mathcal{G}_{\text{global}}$ and a local simulator $\mathcal{G}_{\text{local}}$

input: an exploratory policy π_{explore}

input: a sample-based planner `planner` with a termination condition T , e.g., a fixed time limit

input: a planning horizon \mathcal{H}

Offline *Influence Learning*

Collect a dataset \mathcal{D} of input sequences $D_{\mathcal{H}-1} = (A_{i-1}, S_i^{\text{local}})_{i=1}^{\mathcal{H}-1}$ and target sequences $(S_i^{\text{src}})_{i=1}^{\mathcal{H}-1}$ by interacting with the global simulator $\mathcal{G}_{\text{global}}$ using the policy π_{explore} ;

Train an approximate influence predictor \hat{I}_θ on the dataset \mathcal{D} by minimizing the average empirical KL Divergence between $I(\cdot|D_t)$ and $\hat{I}_\theta(\cdot|D_t)$;

Online *Planning with a sample-based planner*

Integrate the local simulator $\mathcal{G}_{\text{local}}$ and the learned influence predictor \hat{I}_θ into an IALS $\mathcal{G}_{\text{IALS}}^\theta$;

for $t = 0, \dots, \mathcal{H}-1$ **do**

 plan for an action until T is met: $a_t = \text{planner.plan}(\mathcal{G}_{\text{IALS}}^\theta, T)$;

 execute the action in the real environment: $o_{t+1} = \text{env.act}(a_t)$;

 process the new observation: `planner.observe`(o_{t+1})

end

3.2.1 LEARNING APPROXIMATE INFLUENCE PREDICTOR OFFLINE WITH RECURRENT NEURAL NETWORKS

The dependency of $I(S_t^{\text{src}}|D_t)$ on the d-separation set D_t renders it infeasible to be computed exactly online or offline. In this work we learn an approximate influence predictor offline with RNNs by formalizing it as a supervised sequential classification problem.

For planning with horizon H , we need to predict the conditional distribution over the influence source state $I(S_t^{\text{src}}|D_t)$ for $t = 1$ to $H-1$. We do not need to predict $I(S_0^{\text{src}}|D_0)$ as it is the initial belief over the influence source state. As RNNs require the input size to be constant for every time step, we drop the initial local state S_0^{local} from D_t so that the input to RNNs at time step t is $\{A_{t-1}, S_t^{\text{local}}\}$ and the target is S_t^{src} . If there exists a distribution from which we can sample a dataset \mathcal{D} of input sequence D_{H-1} and target sequence $(S_1^{\text{src}}, \dots, S_{H-1}^{\text{src}})$, then this is a classic sequential classification setup that can be learned by training a RNN \hat{I}_θ to minimize the average empirical KL divergence between $I(\cdot|D_t)$ and $\hat{I}_\theta(\cdot|D_t)$ with stochastic gradient descent (SGD) [Ruder, 2016], which yields a cross-entropy loss in practice. While we leave the question on how can we collect the dataset \mathcal{D} in a way that maximizes the online planning performance for future investigation, in this paper we use a uniform random policy to sample \mathcal{D} from the global simulator $\mathcal{G}_{\text{global}}$.

3.2.2 INTEGRATING THE LOCAL SIMULATOR AND RNN INFLUENCE PREDICTOR FOR ONLINE PLANNING

To plan online in a POMDP, sample-based planners like POMCP [Silver and Veness, 2010] require a generative simulator that supports sampling the initial states and transitions. As shown in Figure 2.1b, to sample a transition in the IALS $\mathcal{G}_{\text{IALS}}^\theta$, we need to first sample an influence source state S_t^{src} and then sample the local transitions in the local simulator $\mathcal{G}_{\text{local}}$. While in the original formulation of IBA, $\hat{I}_\theta(S_t^{\text{src}}|D_t)$ conditions on the d-separation set D_t which grows with actions A_t and new local states S_{t+1}^{local} at every time step, we avoid feeding the entire D_t into RNNs for every prediction of S_t^{src} by taking the advantage of RNNs whose hidden state Z_t is a sufficient statistic of the previous inputs. As a result, we use $S_t^{\text{IALM}} = (S_t^{\text{local}}, S_t^{\text{src}}, Z_t)$ as the state of the IALS in practice. The transition $s_{t+1}^{\text{IALM}}, o_{t+1}, r_{t+1} \sim \mathcal{G}_{\text{IALS}}^\theta(s_t^{\text{IALM}}, a_t)$ can then be sampled in two steps:

1. sample the next local state, observation and reward: $s_{t+1}^{\text{local}}, o_{t+1}, r_{t+1} \sim \mathcal{G}_{\text{local}}(s_t^{\text{local}}, s_t^{\text{src}}, a_t)$
 2. sample the next RNN hidden state and influence source state: $z_{t+1}, s_{t+1}^{\text{src}} \sim \hat{I}_\theta(\cdot|z_t, a_t, s_{t+1}^{\text{local}})$
- The initial state S_0^{IALM} of the IALS can be easily sampled by first sampling a full state $s \sim \mu$ and then extracting the local state and the influence source state $(s_0^{\text{local}}, s_0^{\text{src}})$ from s .

3.3 EMPIRICAL ANALYSIS

We perform online planning experiments with the POMCP planner [Silver and Veness, 2010] to answer the following questions: when learning approximate influence predictors with RNNs,

- can planning with an IALS be faster than planning with the global simulator while achieving similar performance, *when the same number of simulations are allowed per*

planning step?

- can planning with an IALS yield better performance than planning with the global simulator, *when the same amount of planning time is allowed per planning step?*

3.3.1 EXPERIMENTAL SETUP

Our codebase was implemented in C++, including a POMCP planner and several benchmarking domains ¹. We ran each of our experiments for many times on a computer cluster with the same amount of computational resources. To report results, we plot the means of evaluation metrics with standard errors as error bars. Details of our experiments are provided in the appendix.

3.3.2 GRAB A CHAIR

The first domain we use is Grab A Chair mentioned in Section 3.1. In our setting, the other agents employ a policy that selects chairs randomly in the beginning and greedily afterwards according to the frequency of observing to obtain a chair when visiting it.

Our intuition is that the amount of speedup we can achieve by replacing $\mathcal{G}_{\text{global}}$ with $\mathcal{G}_{\text{IALS}}^\theta$ depends on how fast we can sample influence source state variables S^{src} from the approximate influence predictor \hat{I}_θ and the size of hidden state variables S^{aux} we can avoid simulating in $\mathcal{G}_{\text{IALS}}^\theta$. We perform planning with different simulators in games of $\{5, 9, 17, 33, 65, 129\}$ agents for a horizon of 10 steps, where a fixed number of 1000 Monte Carlo simulations are performed per step.

To obtain an approximate influence predictor \hat{I}_θ , we sample a dataset \mathcal{D} of 1000 episodes from the global simulator $\mathcal{G}_{\text{global}}$ with a uniform random policy and train a variant of RNN called Gated Recurrent Units (GRU) [Cho et al., 2014] on \mathcal{D} until convergence. To test if capturing the incoming influence is essential for achieving good performance when planning with $\mathcal{G}_{\text{IALS}}^\theta$, we use an IALS with a uniform random influence predictor as an additional baseline, denoted as $\mathcal{G}_{\text{IALS}}^{\text{random}}$.

Figure 3.2a shows the performance of planning with different simulators in scenarios of varying sizes. Clearly, planning with $\mathcal{G}_{\text{IALS}}^\theta$ achieves significantly better performance than planning with $\mathcal{G}_{\text{IALS}}^{\text{random}}$, emphasizing the importance of learning \hat{I}_θ to capture the influence. While planning with $\mathcal{G}_{\text{IALS}}^\theta$ can indeed achieve matching performance with $\mathcal{G}_{\text{global}}$ as shown by the small differences in their returns, the advantage of the IALS, its speed, is shown in Figure 3.2b. In contrast to $\mathcal{G}_{\text{global}}$ which slows down quickly due to the growing number of state variables to simulate, the computation time of both $\mathcal{G}_{\text{IALS}}^\theta$ and $\mathcal{G}_{\text{IALS}}^{\text{random}}$ barely increases. This is because those state variables added by more chairs and agents are abstracted away from the simulations in the IALS with their influence concisely captured by \hat{I}_θ in the distribution of the two neighboring agents' decisions. Note that $\mathcal{G}_{\text{IALS}}^\theta$ is slower than $\mathcal{G}_{\text{global}}$ in scenarios with few agents due to the overheads of feedforward passing in the GRU.

To further investigate how will influence-augmented online planning perform in environments with different *influence strengths*, by which we mean the degree to which the

¹available at <https://github.com/INFLUENCEorg/IAOP>

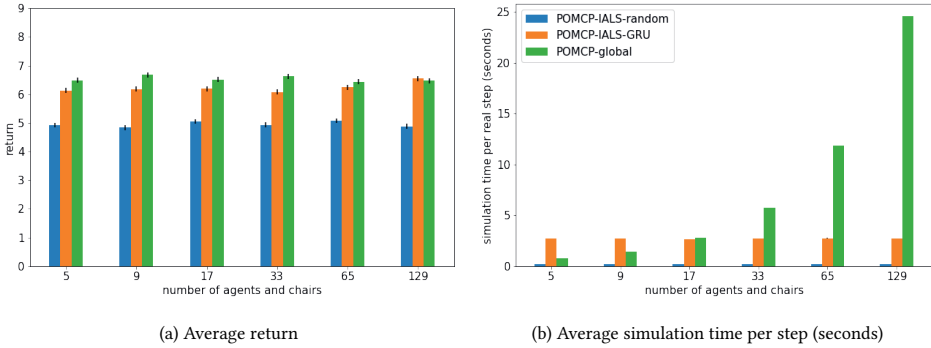


Figure 3.2: Performance of POMCP with different simulators in Grab A Chair games of various sizes. While the IALS with GRU influence predictor achieves matching returns with the global simulator, the simulation is significantly faster in scenarios with many other agents.

local states are affected by the influence source states, we repeat our experiments above in a variant of the 5-agent Grab A Chair game where the only difference is that when two agents target the same chair, both of them have the same probability $p \in [0, 1]$ to obtain the chair². The intuition is that when p is lower, the influence from the rest of the environment will be stronger as the decisions of the two neighboring agents will be more decisive on whether the planning agent can secure a chair. In this case, higher prediction accuracy on the decisions of the two neighboring agents will be required for the agent to plan a good action. Figure 3.3 shows the planning performance with all simulators under decreasing p which implies stronger influence strength from the rest of the environment. While the same amount of effort was put into training the approximate influence predictor \hat{I}_θ , the performance difference between planning with $\mathcal{G}_{\text{IALS}}^\theta$ and $\mathcal{G}_{\text{global}}$ is smaller under higher p . This suggests that in environments where the local planning problem is more tightly coupled with the rest of the environment, learning an accurate influence predictor \hat{I}_θ is more important to achieve good planning performance.

3.3.3 REAL-TIME ONLINE PLANNING IN GRID TRAFFIC CONTROL

The primary motivation of our approach is to improve online planning performance in realistic settings where the planning time per step is constrained. For this reason, we conduct real-time planning experiments in a more realistic domain called Grid Traffic Control, which simulates a busy traffic system with 9 intersections, each of which consists of 4 lanes with 6 grids as shown in Figure 3.4a, with more details provided in the appendix.

The traffic lights are equipped with sensors providing 4-bit information on the existence of vehicles in the 4 nearby grids. While the other traffic lights employ a hand-coded switching strategy that prioritizes lanes with vehicles before the lights and without vehicles after the lights, the traffic light in the center is controlled by planning, aiming to minimize

²Note that this leads to a physically unrealistic setting since it is possible that two agents obtain the same chair at a time step. However, it gives us a way to investigate the impact of the influence strength from the rest of the environment.

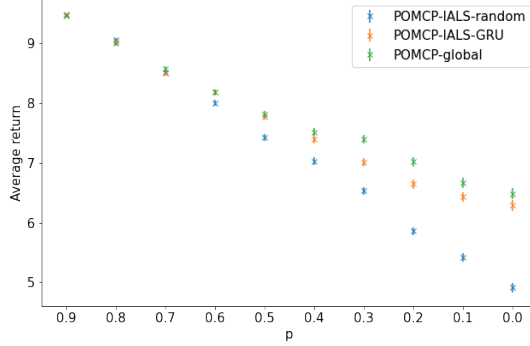


Figure 3.3: Performance of POMCP with different simulators in the modified Grab A Chair game under decreasing p , which implies stronger influence from the rest of the environment. The smaller performance difference between $\mathcal{G}_{\text{IALS}}^{\theta}$ and $\mathcal{G}_{\text{global}}$ under higher p suggests that learning an accurate influence predictor is more important to achieve good planning performance when the local planning problem is more tightly coupled with the rest of the environment.

the total number of vehicles in this intersection for a horizon of 30 steps.

As mentioned in Section 2.3.2, POMCP approximates the belief update with an unweighted particle filter that reuses the simulations performed during the tree search. However, in our preliminary experiments, we observed the particle depletion problem, which occurred when POMCP ran out of particles because none of the existing particles was evidenced by the new observation. While to alleviate this problem we use a workaround inspired by Silver and Veness [2010]³, when particle depletion still occurs at some point during an episode, the agent employs a uniform random policy.

We train an influence predictor with an RNN and evaluate the performance of all three simulators $\mathcal{G}_{\text{IALS}}^{\text{random}}$, $\mathcal{G}_{\text{IALS}}^{\theta}$ and $\mathcal{G}_{\text{global}}$ in settings where the allowed planning time is fixed per step. We posit that $\mathcal{G}_{\text{IALS}}^{\theta}$ will outperform $\mathcal{G}_{\text{global}}$ when the planning time allowed is very constrained because, in that case, the advantage on simulation speed will dominate the disadvantage on simulation accuracy caused by approximating the influence with \hat{I}_{θ} .

Figure 3.4b demonstrates the ability of the IALS to perform more than twice the number of simulations that can be performed by the global simulator within the same fixed time. This is directly translated into the ability of POMCP to plan for more time steps before the particle depletion occurs, as shown in Figure 3.4c. The more important effect of faster simulation is that our approach performs much better than planning on the global simulator especially when the planning time is limited. This suggests that there does exist a trade-off between simulation speed and simulation accuracy that allows planning on the IALS with an approximate influence predictor to achieve better online performance.

Figure 3.6 in the appendix performs a similar time-constrained evaluation in the Grab A Chair domain. The finding there is that the advantage of the IALS on the simulation

³While more advanced particle filters like Sequential Importance Resampling can reduce this problem, we chose to use POMCP in unmodified form to make it easier to interpret the benefits of our approach. Our workaround is that when the search tree is pruned because of a new observation, we add $N/6$ additional particles sampled from the initial belief b_0 to the current particle pool where N is the number of remaining particles.

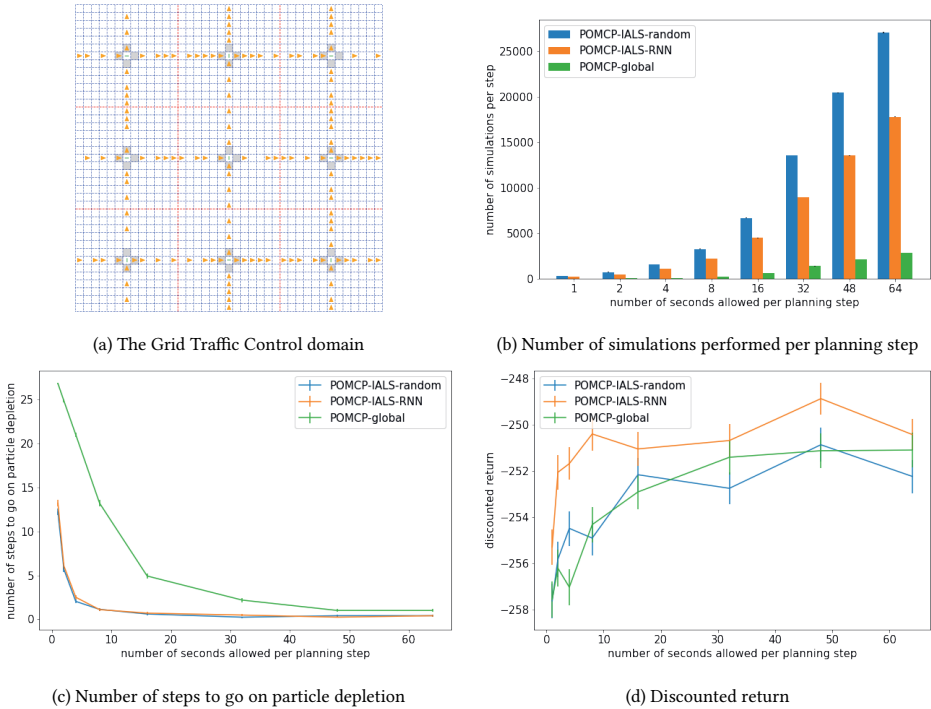


Figure 3.4: Performance of POMCP with different simulators (d) while allowing different numbers of seconds per planning step in the Grid Traffic Control domain (a). The advantage of the IALS on speed allows the POMCP to perform significantly more simulations than with the global simulator, given the same planning budget (b). This allows POMCP to plan for more time steps when particle depletion occurs and achieve better returns (c). While the planning performance of the IALS with trained influence predictor dominates the global simulator when the planning time is constrained, the performance difference decreases when more time is allowed.

speed is clearer when the global model of the problem is more complex, in which cases the IALS with an approximate influence predictor shows a superior performance compared to the global simulator.

3.4 RELATED WORK

The idea of utilizing offline knowledge learning for improved online planning performance has been well-studied [Anthony et al., 2017, Gelly and Silver, 2007, 2011, Silver et al., 2016, 2017b, 2018]. These approaches can be categorized as 1) learning value functions or policies to guide the tree search, 2) improving default policy for more informative rollouts, 3) replacing rollouts with learned value functions and 4) initializing state-action value estimates. Our approach takes a distinct path by speeding up computationally expensive forward simulations, which allows the planner to sample more trajectories for each decision.

Closest to our work is the approach by Chitnis and Lozano-Pérez [2020], which exploits *exogenous variables* to reduce the state space of the model for more efficient simulation

and planning. While both of the approaches learn a more compact model by abstracting away state variables, exogenous variables are fundamentally different from the non-local variables that we abstract away. By definition, exogenous variables refer to those variables that are beyond the control of the agent: they cannot be affected, directly or indirectly, by the agent’s actions [Boutilier et al., 1999, Chitnis and Lozano-Pérez, 2020]. In contrast, the non-local variables that are abstracted away in IBA [Oliehoek et al., 2021] can be chosen more freely, as long as they do not *directly* affect the agent’s observation and reward. Therefore, the exogenous variables and non-local variables are in general two different sets of variables that can be exploited to reduce the state space size. For instance, in the traffic problem of Figure 3.4a, there are no exogenous variables as our action can directly or indirectly effect the transitions at other intersections (by taking or sending vehicles from/to them). This demonstrates that our approach allows us to reduce the state space of this problem beyond the exogenous variables.

The idea of replacing a computationally demanding simulator with an approximate simulator for higher simulation efficiency has been explored in many fields under the name of *surrogate model*, such as computer animation [Grzeszczuk et al., 1999], network simulation [Kazer et al., 2018], the simulation of seismic waves [Moseley et al., 2018] etc. Our work explores this idea in the context of sample-based planning in structured domains.

Recent works in deep model-based reinforcement learning [Farquhar et al., 2018, Hafner et al., 2019, Schrittwieser et al., 2020, van der Pol et al., 2020a] have proposed to learn an approximate model of the environment by interacting with it, and then plan a policy within the learned model for better sample efficiency. Our method considers a very different setting, in which we speed up the simulation for sample-based planning by approximating part of the global simulator, that is, the influence from the rest of the environment, and retain the simulation accuracy by explicitly utilizing a light and accurate local simulator.

3.5 CONCLUSION

In this work we aim to address the problem that simulators modeling the entire environment is often slow and hence not suitable for sample-based planning methods which require a vast number of Monte Carlo simulations to plan a good action. Our approach transforms an expensive factored global simulator into an influence-augmented local simulator (IALS) that is less accurate but much faster. The IALS utilizes a local simulator which accurately models the state variables that are most important to the planning agent and captures the influence from the rest of the environment with an approximate influence predictor learned offline. Our empirical results show that in despite of the simulation inaccuracy caused by approximating the incoming influence with a recurrent neural network, planning on the IALS yields better online performance than planning on the global simulator due to the higher simulation efficiency, especially when the planning time per step is limited. While in this work we collect data from the global simulator with a random exploratory policy to learn the influence, a direction for future work is to study how this offline learning procedure can be improved for better performance during online planning.

3.6 APPENDIX

In this appendix, we provide the details of our experimental setups for reproducibility. Our codebase for this research is open-sourced at <https://github.com/INFLUENCEorg/IAOP>.

3.6.1 GRAB A CHAIR

ENVIRONMENT

The Grab A Chair game is an N -agent game where at every time step, each agent has an action space of two, trying to grab the chair on its left or right side. An agent only secures a chair if its targeted chair is not targeted by a neighboring agent. At the end of a time step t , each agent with $s_{t+1} \in \{0, 1\}$ indicating if this agent obtains a chair receives a reward $r_t = s_{t+1}$ and a noisy observation o_{t+1} on s_{t+1} which has a probability 0.2 to be flipped.

In the experiments of Figure 3.3, when two agents target the same chair, both of them have a probability of $p \in [0, 1]$ to secure the chair, which means that there is a probability that two neighboring agents obtain the same chair. The following setup applies to all the experiments in this domain.

EXPERIMENTAL SETUP

Influence Learning In this domain, the approximate influence predictor \hat{I}_θ is parameterized by a GRU classifier with 8 hidden units. The dataset \mathcal{D} consists of 1000 episodes collected from the global simulator $\mathcal{G}_{\text{global}}$ with a uniform random policy, where 800 episodes are used as the training set and the other 200 episodes are used as the validation set. The hyperparameters used to train the GRU influence predictors in scenarios with $\{5, 9, 17, 33, 65, 129\}$ agents are shown in Table 3.2 and their learning curves are shown in Figure 3.5.

Table 3.2: Hyperparameters used to train the GRU influence predictors for experiments in the Grab A Chair domain, where the weight decay was fine tuned within the range until there is no clear sign of overfitting.

Learning rate	0.0005
Batch size	128
Number of epochs	8000
Weight decay	$[1 \times 10^{-5}, 5 \times 10^{-5}]$

Planning with POMCP The parameters used in the planning experiments with POMCP are shown in Table 3.3.

Real-time Online Planning in Grab A Chair domain We conduct a time-constrained evaluation in this domain with $\{33, 65, 129\}$ agents, similar to the one performed in the Grid Traffic Control domain, where different amount of time is allowed to plan an action. Results in Figure 3.6 show that the advantage of the IALS with GRU influence predictor is clearer when the global model of the planning gets more complex.

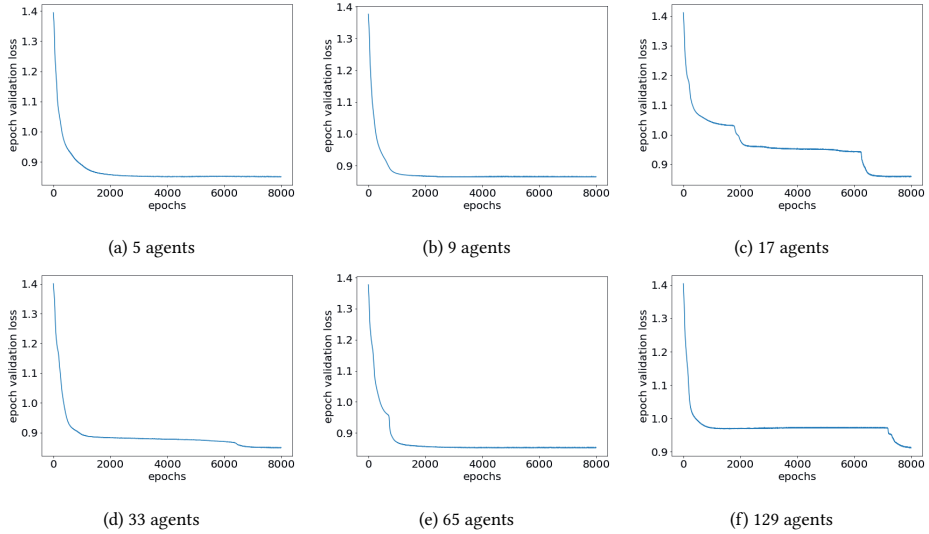


Figure 3.5: Learning curves of influence predictors in the Grab A Chair domain.

Table 3.3: Parameters for the planning experiments with POMCP in the Grab A Chair domain.

Discount factor	1.0
Horizon	10
Number of simulations per step	1000
Number of initial particles	1000
Exploration constant in the UCB1 algorithm (c)	100.0

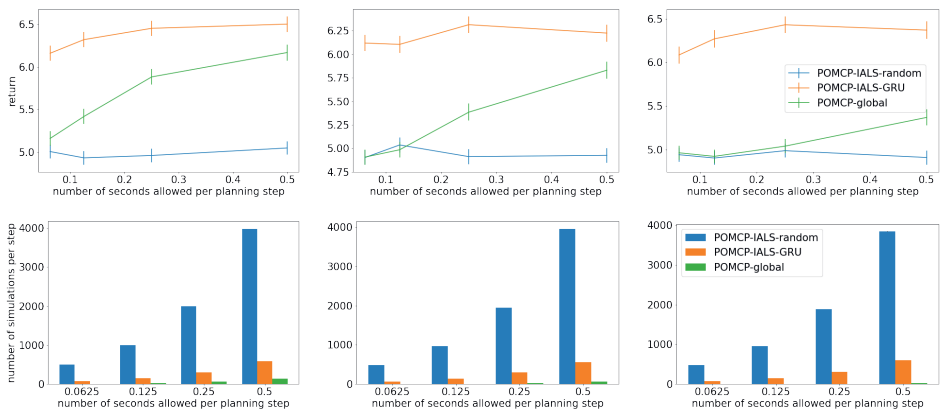


Figure 3.6: Performance of POMCP with different simulators while allowing different numbers of seconds per planning step in Grab A Chair games with 33 (left), 65 (middle), and 129 (right) agents. The advantage of IALS over the global simulator becomes clearer as the planning problem gets more complex.

3.6.2 GRID TRAFFIC CONTROL

ENVIRONMENT

The Grid Traffic Control environment simulates a traffic system of 9 intersections as shown in Figure 3.4a. The vehicles, plotted as yellow arrows, move from the left to right and the bottom to top, governed by the traffic lights in the center of each intersection. While they are initially generated with a probability of 0.7 in each grid, new vehicles will enter the traffic system at entrances on the left and bottom borders whenever they are not occupied at last time step. When reaching the right and bottom borders, with a probability of 0.3, vehicles leave the traffic system.

While the other traffic lights are controlled by fixed switching strategies, the traffic light in the center intersection is controlled by the planning agent, whose action space consists of actions to set the light green for each lane. After an action a_t is taken which results in the movement of vehicles, the agent receives an observation consisting of four Boolean variables $o_{t+1}=\{\text{left_occupied}, \text{right_occupied}, \text{up_occupied}, \text{bottom_occupied}\}$ indicating if the four grids around the traffic light are occupied. The reward r_t is the negative number of grids that are occupied in this intersection after the transition at time step t .

EXPERIMENTAL SETUP

Influence Learning In this domain, the approximate influence predictor \hat{I}_θ is parameterized by a RNN classifier with 2 hidden units. The dataset \mathcal{D} consists of 1000 episodes collected from the global simulator $\mathcal{G}_{\text{global}}$ with a uniform random policy, where 800 episodes are used as the training set and the other 200 episodes are used as the validation set. The hyperparameters used to train the RNN influence predictor are shown in Table 3.4 and its learning curve is shown in Figure 3.7.

Table 3.4: Hyperparameters used to train the RNN influence predictor for experiments in the Grid Traffic Control domain.

Learning rate	0.00025
Batch size	128
Number of epochs	8000
Weight decay	1×10^{-4}

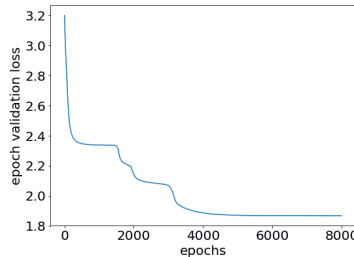


Figure 3.7: The learning curve of the influence predictor in the Grid Traffic Control domain.

Planning with POMCP The parameters used in the planning experiments with POMCP are shown in Table 3.5, where effective horizon is the maximal depth from the root node that a search or a rollout will be performed.

Table 3.5: Parameters for the planning experiments with POMCP in the Grid Traffic Control domain.

Discount factor	0.95
Horizon	30
Number of seconds allowed per planning step	{1, 2, 4, 8, 16, 32, 48, 64}
Number of initial particles	1000
Exploration constant in the UCB1 algorithm (c)	10.0
Effective horizon	18

4

LEARNING AND PLANNING WITH SELF-IMPROVING SIMULATORS

4

"Mental effort, I would argue, is relatively rare. Most of the time we coast."

Daniel Kahneman

How can we plan efficiently in a large and complex environment when the time budget is limited? Given the original simulator of the environment, which may be computationally very demanding, we propose to learn online an approximate but much faster simulator that improves over time. To plan reliably and efficiently while the approximate simulator is learning, we develop a method that adaptively decides which simulator to use for every simulation, based on a statistic that measures the accuracy of the approximate simulator. This allows us to use the approximate simulator to replace the original simulator for faster simulations when it is accurate enough under the current context, thus trading off simulation speed and accuracy. Experimental results in two large domains show that when integrated with POMCP, our approach allows to plan with improving efficiency over time.

4.1 INTRODUCTION

Decision making under uncertainty is one of the key problems in artificial intelligence [Kaelbling et al., 1998, Spaan, 2012]. Online planning methods, such as POMCP [Silver and Veness, 2010] and DESPOT [Ye et al., 2017], have become popular as they enable time-efficient decision making by focusing the computation on the current context. However, these methods rely heavily on fast simulators that can rapidly perform a large number of Monte Carlo simulations. Unfortunately, many real-world domains are complex and thus slow to simulate, which impedes real-time online planning.

To mitigate the simulation cost, researchers have explored learning surrogate models that are computationally less expensive [Buesing et al., 2018, Chitnis and Lozano-Pérez, 2020, Grzeszczuk et al., 1999]. However, one disadvantage of these methods is that they learn a model of the *entire* environment, which may be unnecessary and is often difficult. Our previous approach (Chapter 3; He et al. [2020]) builds on the framework of influence-based abstraction (Chapter 2.2; Oliehoek et al. [2021]) to overcome this problem: We propose to exploit the structure of the environment via a so-called influence-augmented local simulator (IALS), which uses a lightweight simulator to capture the local dynamics that surround the agent and learns an influence predictor \hat{I}_θ that accounts for the interaction between the local dynamics and the rest of the environment.

However, this "two-phase" paradigm, in which a simulator is learned offline and then used as-is for online simulation and planning, has three main limitations. First, no planning is possible until the offline learning phase finishes, which can take a long time. Second, the separation of learning and planning raises a question of what data collection policy should be used during training to ensure good online prediction during planning. We empirically demonstrate that when the training data is collected by a uniform random policy, the learned influence predictors can perform poorly during online planning due to distribution shifts. Third, completely replacing the original simulator with the approximate one after training implies a risk of poor planning performance in certain situations, which is hard to detect in advance.

In this work, we aim to overcome these drawbacks by investigating if we can learn the influence predictor \hat{I}_θ used in an IALS online, *without pretraining*. As the quality of such an IALS with an untrained influence predictor would initially be poor, we investigate if we can selectively use the learning IALS and the accurate but slow global simulator (GS) to balance the simulation speed and accuracy. To address this major challenge, we propose a simulator selection mechanism to choose between the GS and IALS based on the latter's accuracy under the current context, which is estimated online from the simulations of the GS. This enables us to use the fast IALS when it is sufficiently accurate, while reverting back to the GS otherwise.

The experiments reveal that as the influence predictor becomes more accurate, the simulator selection mechanism starts to use the IALS more and more often, which allows for high-accuracy simulations with increasing speed. Planning with such *self-improving simulators* speeds up decisions (given a fixed number of simulations) and increases task performance (given a fixed time budget), without pretraining the influence predictor \hat{I}_θ . Moreover, we find that influence predictors that are trained with online data, i.e., coming from online simulations with the GS, can significantly outperform those trained with offline

data, both in terms of online prediction accuracy and planning performance.

4.2 SELF-IMPROVING SIMULATORS

To address the limitations mentioned earlier of learning the IALS offline (the need for pre-training, the gap between states visited in training vs. planning, and the inherent approximation due to function approximation), we introduce a new approach that 1) starts planning from the start while collecting data to train the surrogate model, 2) trains the influence predictor with this data, which is now more pertinent to planning and 3) reverts back to the global simulator when the IALS is considered inaccurate for the current context (i.e., the current history and future trajectories we are likely to reach). Additionally, by switching back to the global simulator when the IALS is inaccurate, we provide more training data to improve the IALS precisely for those inaccurate contexts. As such, the IALS will improve over time and be used more frequently, improving overall simulation efficiency. Therefore, we refer to our approach as a *self-improving simulator (SIS)*.

4

4.2.1 PLANNING WITH SELF-IMPROVING SIMULATORS

We propose and test the integration of these ideas in POMCP [Silver and Veness, 2010], although the principle is transferable to other planning methods such as DESPOT [Ye et al., 2017], which also involves iterative simulations. Algorithm 1 outlines our approach, and Figure 4.1 illustrates it. The planning begins with an arbitrarily initialized influence predictor \hat{I}_θ , combined with the known¹ local simulator $\mathcal{G}_{\text{local}}$, which models the local dynamics $(\mathcal{T}^{\text{local}}, \mathcal{O}, \mathcal{R})$, to form the IALS $\mathcal{G}_{\text{IALS}}^\theta = (\mathcal{G}_{\text{local}}, \hat{I}_\theta)$. For each simulation, a simulator is chosen between $\mathcal{G}_{\text{global}}$ and $\mathcal{G}_{\text{IALS}}^\theta$ to generate a new trajectory τ_i (line 5). This trajectory is then used to update the statistics and expand a new tree node (line 6), regardless of the simulator used. When τ_i comes from $\mathcal{G}_{\text{global}}$, we extract and store training data from τ_i (line 7). Selecting the action to take is the same as in regular POMCP (line 9).

The data stored, denoted as $\mathcal{D} = \{(d_k, s_k^{\text{src}})\}$ for any $0 \leq k \leq H-1$, can then be used as a replay buffer to further improve \hat{I}_θ regularly, for example, in between real episodes. To train the approximate influence predictor \hat{I}_θ , parameterized as a recurrent neural network, we use the same method as described in Section 3.2.1 and treat it as a sequential classification task where the cross entropy loss is minimized by stochastic gradient descent:

$$\mathcal{L}(\theta; \mathcal{D}) = -\frac{1}{|\mathcal{D}|} \sum_{d_k, s_k^{\text{src}} \in \mathcal{D}} \log \hat{I}_\theta(s_k^{\text{src}} | d_k). \quad (4.1)$$

Overall, we expect to mainly select the global simulator for earlier simulations since \hat{I}_θ is not yet very accurate. Over time, we expect the IALS to become more accurate and thus more frequently used. However, global simulations are needed to assess the accuracy of the IALS, thus leading to a complex exploration/exploitation problem.

¹ $\mathcal{G}_{\text{local}}$ is an extractable subset of the 2DBN representation of $\mathcal{G}_{\text{global}}$, as illustrated in Figure 2.1.

Algorithm 1 Planning with the Self-improving Simulator

```

1: initialize the influence predictor  $\hat{I}_\theta$ 
2: for every real episode do
3:   for every time step  $t$  do
4:     for every simulation  $i = 0, \dots$  do
5:       select either  $\mathcal{G}_{\text{global}}$  or  $\mathcal{G}_{\text{IALS}}^\theta$  and simulate trajectory  $\tau_i$  (Section 4.2.2)
6:       update the search tree with  $\tau_i$ 
7:       if  $\mathcal{G}_{\text{global}}$  generated  $\tau_i$ , add training data to  $\mathcal{D}$ 
8:     end for
9:     take action recommended by POMCP
10:    prune the search tree with new observation
11:  end for
12:  train the influence predictor  $\hat{I}_\theta$  on  $\mathcal{D}$  for  $N$  steps (Section 3.2.1)
13: end for

```

4

4.2.2 ONLINE SIMULATOR SELECTION

The primary challenge in our approach lies in determining which simulator to select *online* for both fast *and* accurate simulations. We need to select simulators online because of two specific difficulties:

1. The IALS learns online between real episodes (but not within them), meaning its accuracy is unknown in advance and must be estimated on the fly when planning a decision.
2. The accuracy of the approximate influence predictor, and consequently the IALS, can vary based on the input distributions (i.e., the history of local states and actions). These distributions are influenced by the tree policy (search policy) defined by the current tree statistics, which evolve during a planning step from simulation to simulation, making them non-stationary.

To address the second difficulty, we introduce a context-conditioned accuracy measure of \hat{I}_θ that considers the expected input distribution. For the first difficulty, even though the exact influence source distribution is not accessible, we demonstrate that it is feasible to estimate this accuracy measure *online* using simulators from the global simulator. Finally, we discuss how to handle the exploration-exploitation trade-off when selecting simulators online, employing the UCB1 algorithm [Auer et al., 2002].

IALS ACCURACY MEASURE

The approximate IALS $\mathcal{G}_{\text{IALS}}^\theta$ consists of an exact local model $\mathcal{G}_{\text{local}}$ and a learned influence predictor \hat{I}_θ . To evaluate the accuracy of the IALS, one effective approach is to measure the Kullback–Leibler (KL) divergence between the true influence source distribution I and the distribution predicted by the influence predictor \hat{I}_θ . This method is motivated by theory [Congeduti et al., 2021], which shows that the maximum KL divergence across any time step and history of local states and actions can provide an upper bound on the value loss.

To properly account for the context in which the influence predictor operates, we need to consider the setting in which it is used. As described in Section 2.3.2, each POMCP

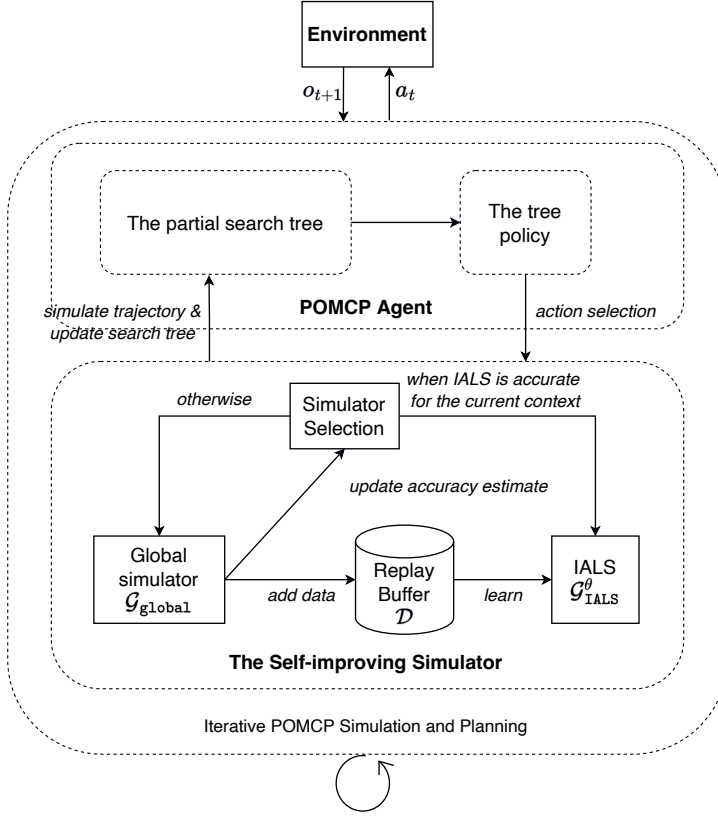


Figure 4.1: General procedure of planning with the self-improving simulator. Note that all simulations from the global simulator are reused for three purposes. First, they are used to update the search tree as in regular online planning. Second, they are used to estimate the accuracy of the IALS for the current context. Third, they are used to construct training data for further improving the approximate influence predictor.

simulation begins by sampling a possible state of the environment from the belief, which is approximated by a particle filter. Following this, a trajectory is sampled from the model, during which the influence predictor is called upon at each simulation step to predict the distribution over influence source states. Therefore, we consider the context for the influence predictor as the *expected* distribution over its input—the history of local states and actions—given the history at the root node and the current state of the search tree, which decides the actions to be selected for simulation.

In the following, we first derive the distribution under which influence prediction occurs, and consequently, the distribution with respect to which the influence predictor should be accurate. Based on this input distribution, we then define an accuracy metric for evaluating the influence predictor.

Formally, at each real time step t , given the history h_t , we denote the (partial) search tree of POMCP at the start of simulation i as tree_i and the corresponding tree policy as π_i^{tree} , which maps a simulated history to a distribution over actions. For each simulated

time step $k \geq t$ in simulation i , the expected distribution over local histories D_k in the true IALM (with an exact influence predictor) is defined as $\Pr(D_k|h_t, \pi^{\text{tree}_i}; \mathcal{M}_{\text{IALM}})$. For every $d_k = (s_{0:k}^{\text{local}}, a_{0:k-1})$, this is expressed as:

$$\Pr(d_k|h_t, \pi^{\text{tree}_i}; \mathcal{M}_{\text{IALM}}) = \Pr(s_{0:k}^{\text{local}}, a_{0:k-1}|h_t, \pi^{\text{tree}_i}; \mathcal{M}_{\text{IALM}}) \quad (4.2)$$

$$= \Pr(d_t = (s_{0:t}^{\text{local}}, a_{0:t-1}), s_{t+1:k}^{\text{local}}, a_{t:k-1}|h_t, \pi^{\text{tree}_i}; \mathcal{M}_{\text{IALM}}) \quad (4.3)$$

$$= \underbrace{\Pr(d_t|h_t; \mathcal{M}_{\text{IALM}})}_{\text{belief}} \underbrace{\Pr(s_{t+1:k}^{\text{local}}, a_{t:k-1}|d_t, h_t, \pi^{\text{tree}_i}; \mathcal{M}_{\text{IALM}})}_{\text{dynamics}} \quad (4.4)$$

The expected KL divergence for simulated time step k in the i -th simulation, between the true influence source distribution I and the approximate one \hat{I}_θ , is then defined as:

$$\mathcal{L}_i^k \stackrel{\text{def}}{=} \mathbb{E}_{D_k \sim \Pr(\cdot|h_t, \pi^{\text{tree}_i}; \mathcal{M}_{\text{IALM}})} [D_{\text{KL}}(I(S_t^{\text{src}}|D_k) \parallel \hat{I}_\theta(S_t^{\text{src}}|D_k))] \quad (4.5)$$

Note that the accuracy measure here is defined with respect to the distribution induced by the exact IALM rather than the approximate IALM. This choice is intentional and will be clarified later: it allows us to use the global simulator to estimate this accuracy measure since the global simulator induces the same distribution as the exact IALM.

However, our accuracy measure should not only consider the accuracy of \hat{I}_θ at one simulated step but should also take the upcoming simulated steps into account. To achieve this, we average the expected KL divergence over all simulated time steps, providing an overall accuracy measure of using the approximate IALS for simulation i :

$$\mathcal{L}_i \stackrel{\text{def}}{=} \frac{1}{H-t} \sum_{k=t}^{H-1} \mathcal{L}_i^k \quad (4.6)$$

In essence, if we are planning a decision for real time step t and have conducted $i-1$ simulations, then \mathcal{L}_i quantifies the expected average KL divergence between the true influence source distribution and the one approximated by \hat{I}_θ for simulation i . In other words, it measures the expected approximate error introduced by using the current IALS for the next simulation instead of the exact IALS or global simulator, which defines the same history MDP as discussed in Section 2.2.

ESTIMATING THE ACCURACY ONLINE

For every simulated step $t \leq k \leq H-1$ of simulation i , where H is the planning horizon,

$$\mathcal{L}_i^k = \mathbb{E}_{D_k \sim \Pr(\cdot|h_t, \pi^{\text{tree}_i}; \mathcal{M}_{\text{IALM}})} [D_{\text{KL}}(I(S_t^{\text{src}}|D_k) \parallel \hat{I}_\theta(S_t^{\text{src}}|D_k))] \quad (4.7)$$

$$= \mathbb{E}_{D_k \sim \Pr(\cdot|h_t, \pi^{\text{tree}_i}; \mathcal{M}_{\text{IALM}})} [\underbrace{H(I(S_t^{\text{src}}|D_k), \hat{I}_\theta(S_t^{\text{src}}|D_k))}_{\text{cross entropy}} - \underbrace{H(I(S_t^{\text{src}}|D_k))}_{\text{entropy}}] \quad (4.8)$$

This metric could be estimated by sampling D_k using the tree policy and exactly computing the KL if we had access to the exact influence source distribution $I(S_k^{\text{src}}|D_k)$ in the discrete case. However, this is not the case. Instead, we demonstrate below how to construct an estimate of this accuracy measure using simulations from the global simulator.

Specifically, Proposition 2 shows that the expected cross entropy in Equation (4.8) can be re-expressed as an expectation under the global model. Following this, Proposition 3 establishes that a lower bound for the expected entropy can also be derived using the global model. By combining these two results, we construct an *upper* bound for the expected KL divergence that relies on the global model (Proposition 4), which is accessible for sampling. Consequently, we can estimate an upper bound for \mathcal{L}_i^k using simulations from the global model.

Proposition 2 (Cross Entropy Equivalence). *For every simulated step $t \leq k \leq \mathcal{H}-1$ of simulation i , the expected cross entropy between the true influence source distribution $I(S_k^{\text{src}}|D_k)$ and its approximation $\hat{I}_\theta(S_k^{\text{src}}|D_k)$ under the exact IALM can be equivalently expressed under the global model $\mathcal{M}_{\text{global}}$:*

$$\mathbb{E}_{D_k \sim \Pr(\cdot|h_t, \pi^{\text{tree}_i}; \mathcal{M}_{\text{IALM}})} [H(I(S_k^{\text{src}}|D_k), \hat{I}_\theta(S_k^{\text{src}}|D_k))] \quad (4.9)$$

$$= -\mathbb{E}_{D_t \sim \Pr(\cdot|h_t; \mathcal{M}_{\text{IALM}})} \left[\mathbb{E}_{S_{t+1:k}^{\text{local}}, A_{t:k-1} \sim \Pr(\cdot|D_t, h_t, \pi^{\text{tree}_i}; \mathcal{M}_{\text{IALM}})} \left[\mathbb{E}_{S_k^{\text{src}} \sim I(\cdot|D_k)} [\log \hat{I}_\theta(S_k^{\text{src}}|D_k)] \right] \right] \quad (4.10)$$

$$= -\mathbb{E}_{D_k, S_k^{\text{src}} \sim P(\cdot|h_t, \pi^{\text{tree}_i}; \mathcal{M}_{\text{global}})} [\log \hat{I}_\theta(S_k^{\text{src}}|d_k)] \quad (4.11)$$

$$= -\underbrace{\mathbb{E}_{D_t, S_t \sim \Pr(\cdot|h_t; \mathcal{M}_{\text{global}})}}_{\text{belief}} \underbrace{\left[\mathbb{E}_{S_{t+1:k}, A_{t:k-1} \sim \Pr(\cdot|S_t, h_t, \pi^{\text{tree}_i}; \mathcal{M}_{\text{global}})} [\log \hat{I}_\theta(S_k[S_k^{\text{src}}]|D_k)] \right]}_{\text{dynamics}} \quad (4.12)$$

(where $S_k[S_k^{\text{src}}]$ represents the S_k^{src} part of S_k and $D_k = D_t A_t S_{t+1}[S_{t+1}^{\text{local}}] \dots A_{k-1} S_k[S_k^{\text{local}}]$)

Proof. See Appendix 4.6.1. □

Equation (4.10) shows how we can estimate the cross entropy with an exact IALS by sampling S_k^{src} and D_k and using the influence predictor to compute the log-likelihood $\hat{I}_\theta(S_k^{\text{src}}|D_k)$. However, we do not have access to the exact influence source distribution and, as such, the exact IALS. Fortunately, Equation (4.12) presents an alternative approach that uses simulations from the global simulator instead. This approach only requires one modification: We need to keep track of not only S_k but also D_k when performing particle filtering and forward simulation with the global simulator.

Unlike cross entropy, we cannot directly estimate entropy from samples due to the lack of an unbiased estimator for entropy [Paninski, 2003]. Here, we choose to use the entropy of the same distribution but instead conditioned on the global state as a lower bound of the entropy to estimate.

Proposition 3 (Entropy Lower Bound). *For every simulated step $t \leq k \leq \mathcal{H}-1$ of simulation i , the expected entropy of the true influence source distribution $I(S_k^{\text{src}}|D_k)$ under the exact IALM can be lower bounded using the global model $\mathcal{M}_{\text{global}}$ by conditioning on the global*

state:

$$\begin{aligned} & \mathbb{E}_{D_k \sim \Pr(\cdot | h_t, \pi^{\text{tree}_i}; \mathcal{M}_{\text{IALM}})} [H(I(S_k^{\text{src}} | D_k))] \\ & \geq \underbrace{\mathbb{E}_{D_t, S_t \sim \Pr(\cdot | h_t; \mathcal{M}_{\text{global}})}}_{\text{belief}} \left[\underbrace{\mathbb{E}_{S_{t+1:k-1}, A_{t:k-1} \sim \Pr(\cdot | S_t, h_t, \pi^{\text{tree}_i}; \mathcal{M}_{\text{global}})}}_{\text{dynamics}} \left[\underbrace{H(S_k^{\text{src}} | S_{k-1}, A_{k-1})}_{\text{from 2DBN}} \right] \right] \quad (4.13) \end{aligned}$$

Proof. See Appendix 4.6.1. \square

4

This inequality holds intuitively because as a Markovian signal, the global state S_k contains more information than the local history D_k on the random variable S_k^{src} . The entropy $H(S_k^{\text{src}} | S_{k-1}, A_{k-1})$ can be directly computed with the 2DBN, which we assume is given. Importantly, the use of a lower bound on the entropy means that we estimate an upper bound on the KL divergence, making sure we do not underestimate the inaccuracy of the IALS simulations.

Overall, Equations (4.12) and (4.13) construct an upper bound on the expected average KL divergence between the true influence source distribution and the one approximated by \hat{I}_θ , for simulation i of a planning step.

Proposition 4 (KL Divergence Upper Bound).

$$\begin{aligned} \mathcal{L}_i \leq \frac{1}{\mathcal{H}-t} \sum_{k=t}^{\mathcal{H}-1} \mathbb{E}_{D_t, S_t \sim \Pr(\cdot | h_t; \mathcal{M}_{\text{global}})} & \left[\mathbb{E}_{S_{t+1:k}, A_{t:k-1} \sim \Pr(\cdot | S_t, h_t, \pi^{\text{tree}_i}; \mathcal{M}_{\text{global}})} \right. \\ & \left. \left[-\log \hat{I}_\theta(S_k[S_k^{\text{src}}] | D_k) - H(S_k^{\text{src}} | S_{k-1}, A_{k-1}) \right] \right] \quad (4.14) \end{aligned}$$

where $D_k = D_t A_t S_{t+1}^{\text{local}} \dots A_{k-1} S_k^{\text{local}}$ and $H(S_k^{\text{src}} | S_{k-1}, A_{k-1})$ is the exact entropy from 2DBN.

Proof. The proof of this Proposition is straightforward by applying Propositions 2 and 3 in Equation (4.8). \square

As mentioned, the quantity above can be estimated with samples from the global simulator by augmenting the state S_k with the history of local states and actions D_k . Specifically, to perform a POMCP simulation at time step t , we start by sampling a possible pair (s_t, d_t) from the root node. For every simulated step k , we first perform the simulation with the global simulator $s_{k+1}, r_k, o_{k+1} \sim \mathcal{G}_{\text{global}}(s_k, a_k)$ and then let $d_{k+1} = d_k a_k s_{k+1}^{\text{local}}$. The empirical average KL divergence is then:

$$l_i = \frac{1}{\mathcal{H}-t} \sum_{k=t}^{\mathcal{H}-1} [-\log \hat{I}_\theta(s_k^{\text{src}} | d_k) - H(S_k^{\text{src}} | s_{k-1}, a_{k-1})] \quad (4.15)$$

DECIDING BETWEEN THE SIMULATORS

Finally, we want to use the estimated accuracy to decide whether to use the IALS or the global simulator for a simulation. However, alluded to before, the simulations from the GS will not only be used for planning but also for estimating the accuracy of the IALS, and this will need to be done every time we plan for a decision: Even if in the past we determined that \hat{I}_θ is accurate for the subtree of the current history h_t , it is possible that due to learning for other histories the accuracy for this h_t has decreased. Moreover, due to the constant updating of the search tree during a planning step, the tree policy also changes, which can also invalidate past accuracy estimates. As such, we face an ongoing exploration (assessing accuracy) and exploitation (of accurate \hat{I}_θ) problem.

To address this, we propose to model the question which simulator to use for the i -th simulation as a bandit problem, and apply the UCB1 algorithm Auer et al. [2002]. In the following, we define the values of using the global simulator $\mathcal{G}_{\text{global}}$ and the IALS $\mathcal{G}_{\text{IALS}}^\theta$ for every simulation i :

$$V_i^{\mathcal{G}_{\text{IALS}}^\theta} = \hat{\mathcal{L}} + c^{\text{meta}} \sqrt{\frac{\log(N^{\mathcal{G}_{\text{IALS}}^\theta})}{i}}$$

$$V_i^{\mathcal{G}_{\text{global}}} = -\lambda + c^{\text{meta}} \sqrt{\frac{\log(N^{\mathcal{G}_{\text{global}}})}{i}}$$

Here, $N^{\mathcal{G}_{\text{global}}}$ and $N^{\mathcal{G}_{\text{IALS}}^\theta}$ are the number of calls to the global simulator and the IALS, respectively. $\hat{\mathcal{L}}$ is an average of the empirical average KL divergence, and λ quantifies the extra computation cost of using the global simulator for a simulation, compared to the IALS. In practice, it is treated as a hyperparameter as it also reflects how willingly the user may sacrifice simulation accuracy for efficiency. Note that even though UCB1 is theoretically well understood, bounding the ‘sample complexity’ here is impossible as both the history-visit distribution as well the accuracy of the IALS can continually change.

4.3 EMPIRICAL ANALYSIS

In this section, we first evaluate the main premise of our approach: *can selecting between a global simulator and an online learning IALS lead to a self-improving simulator?* Such improvement can manifest itself in faster decision making (when fixing the number of simulations per real time step), or in better performance (when fixing the time budget for each real time step). We investigate both. We also compare our on-line learning approach to the existing two-phase approach.

Experimental Setup We perform the evaluation on two large POMDPs introduced by He et al. [2020], the Grab A Chair (GAC) domain and the Grid Traffic Control (GTC) domain, of which descriptions can be found in Appendix 4.6.3. In all planning experiments with self-improving simulators, we start with a IALS that makes use of a completely untrained \hat{I}_θ , implemented by a GRU; after every real episode it is trained for 64 gradient steps with the accumulated data from the global simulations. The results are averaged over 2500

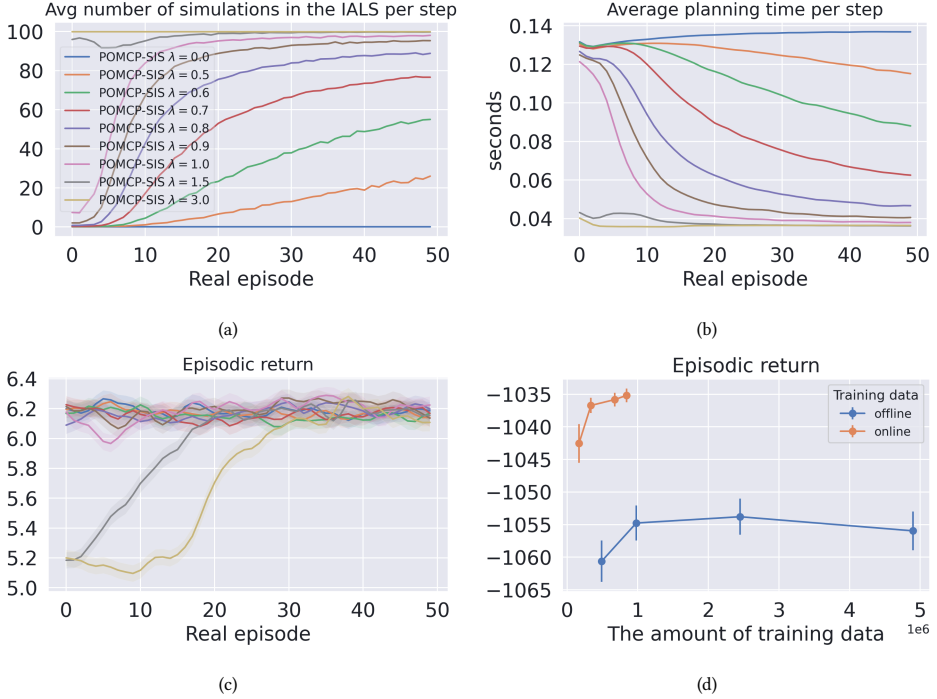


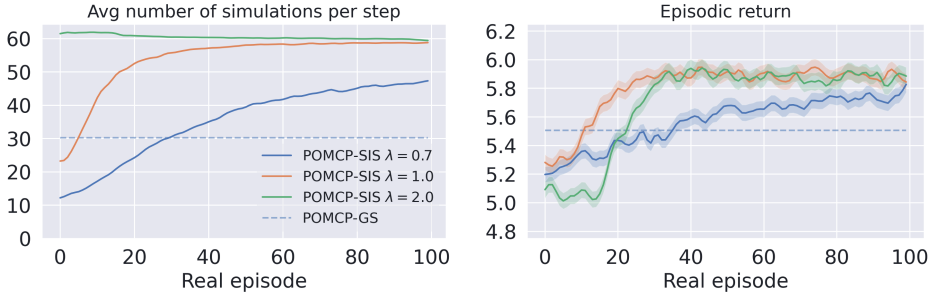
Figure 4.2: (a-c) Simulation controlled planning results for grab a chair. Planning time reduces without significant drop in return for small λ 's. (d) Time controlled planning results for grid traffic control, with IALSs that make use of influence predictors trained on offline (from a uniform random policy) and online (from self-improving simulator with $\lambda=0.7$) data. This experiment is repeated for 20 times.

and 1000 individual runs for the GAC and GTC domains, respectively. Further details are provided in Appendix 4.6.2.

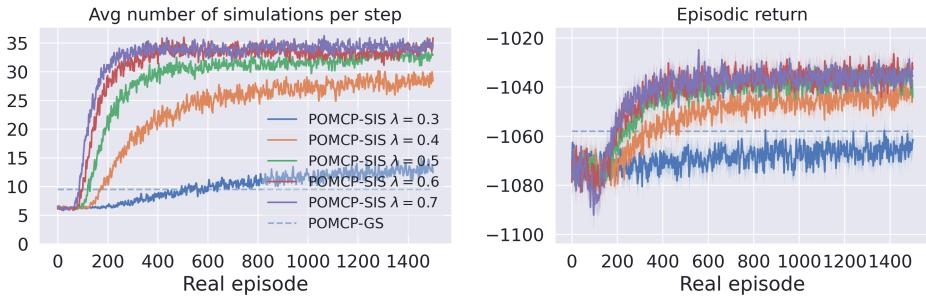
Online Planning with Fixed Number of Simulations In this experiment, we fix the number of POMCP simulations to 100 per planning step and investigate two questions: 1) can planning with self-improving simulators enable increasing planning efficiency? and 2) how does the choice of the hyperparameter λ affect the performance of planning? Since traffic domains have fixed time constraints, we only evaluate on GAC. In addition, GAC as the simpler domain allows us to explore many different settings for lambda λ .

Figure 4.2a shows how many of the 100 simulations are performed using the fast IALS (on average over all real time steps in the episode) for different λ . Remember that higher λ implies the planner is more willing to sacrifice the simulation accuracy for efficiency. For moderate λ s, there is a clear trend that the use of IALS is increasing over time, due to the increasing accuracy of the approximate influence predictor \hat{I}_θ , which is confirmed by the reduced training loss provided in Appendix 4.6.4. This translates into the increasing planning speed shown in Figure 4.2b.² Note that large λ s like 1.5 and 3.0 produce very fast

²Planning time with $\lambda=0.0$ grows slowly due to the practical issue of memory fragmentation, which can be fixed



(a) Grab a chair results.



(b) Grid traffic control results.

Figure 4.3: Time controlled planning results. Performance improves because of increasingly accurate IALS and more simulations.

planning directly from the first episode. However, as shown in Figure 4.2c, the price for that is poor performance at the earlier episodes, because they begin by heavily using the IALS with untrained influence predictors. Moreover, Figures 2b and 2c show that planning time for $\lambda = 1.0$ becomes comparable to $\lambda = 1.5$ and $\lambda = 3.0$ after roughly 10 episodes, while having much better performance at that point. This can be explained by the lack of training data caused by not using the global simulator often enough. Importantly, it seems that there is no clear sign of decreasing performance for moderate λ s, which supports our key argument that in this setting, our approach enables planning with increasing speed without sacrificing task performance.

Real-Time Online Planning In this experiment, we evaluate the performance of our approach under limited time budget. On the one hand, the IALS is an approximate model that introduces errors, certainly when the accuracy of its \hat{I}_θ is still poor in early episodes. On the other hand, using the faster IALS means that we can do many more simulations per real time step. As such, we investigate the ability of our proposed simulator selection mechanism to *dynamically* balance between these possible risks and benefits of the IALS.

We conduct planning experiments in both the grab a chair domain and the grid traffic

with the memory pool technique from David Silver’s implementation of POMCP.

control domain, allocating 1/64 and 1/16 seconds per decision, respectively. Results are shown in Figure 4.3. As a baseline, we include the global simulator (without abstraction), with its performance shown as dotted lines. We observe an initial overhead when using the self-improving simulators, reflected in a lower number of simulations at the start. This overhead arises from the simulator selection mechanism, which estimates KL divergence using the approximate influence predictor \hat{I}_θ . While this adds computational cost, it could be reduced with a more efficient RNN implementation and is expected to become less significant in larger domains.

In the grab a chair domain, we see that the self-improving simulators can perform more simulations for later episodes. This is expected: for later episodes the accuracy of the approximate influence predictor \hat{I}_θ improves, which means that the IALS is selected more frequently, thus enabling more simulations per real time step on average. The figure also shows that this larger number of planning simulations translates to better performance: for appropriate values of λ , after planning for a number of real episodes, the performance increases to a level that planning with the global simulator alone cannot reach. We stress that, in line with the results in the previous subsection, the increase in return is not *only* explained by doing more simulations: the quality of the \hat{I}_θ does matter, as is clearly demonstrated by $\lambda = 2.0$ in GAC: even though it performs many simulations per step from episode 1, its performance only surpasses that of the global simulator baseline after 20 episodes. Overall, these results show that self-improving simulators enables planning with increasing efficiency, and can outperform using an exact global simulator without pre-training an influence predictor.

Further comparison to the two-phase approach Another limitation of the two-phase approach is: the offline learned influence predictors may generalize poorly when making online predictions due to the distribution shift issue caused by training with data collected by an exploration policy. We investigate if this is a real issue that can affect performance and if learning with online data, as in our approach, can help.

We reuse the real-time planning setting in GTC. Following He et al. [2020], we collect offline datasets of varying sizes using a uniform random policy in the global simulator. For comparison, we also collect online datasets—of similar sizes—from the replay buffer of the self-improving simulator (with $\lambda=0.7$) after 1500 episodes of planning. These datasets are then used to train influence predictors offline, which are subsequently used as part of the IALSs for online planning. We show the results in Figure 4.2d.

While it seems that more offline training data can be helpful, overall the performance of the influence predictors that are trained with online data completely dominates those that are trained with offline data: with much less online data, we can achieve a level of performance that is impossible with much more offline data. Figure 4.8 in appendix shows that this indeed might be caused by distribution shift issue: when evaluated on a test dataset that is collected by running POMCP on the global simulator, the influence predictors being trained on the offline data can have an increasing test loss. Moreover, training on the online data from the self-improving simulator results in a much lower test loss. As such, we can conclude that there is indeed a distribution shift issue with the two-phase approach that can harm the planning performance, which is addressed in our approach.

4.4 RELATED WORK

Surrogate models have been widely used in many applications, such as aerodynamic design prediction Pehlivanoglu and Yagiz [2012], spacecraft path planning Peng and Wang [2016] and satellite reconfiguration Chen et al. [2020]. It has been proposed as a general approach to constraint optimization problems Audet et al. [2000]. For fast simulations in large factored POMDPs, He et al. [2020] propose to construct influence-augmented local simulators as surrogate models to a slow global simulator, which we extend in this work.

There is a strong connection between our work and multi-fidelity methods Kennedy and O’Hagan [2000], Peherstorfer et al. [2018], which combine models with varying costs and fidelities to accelerate outer-loop applications such as optimization and inference. In essence, the simulator selection mechanism in our approach does the same: It selects between a slow high-fidelity global simulator with high cost and a fast low-fidelity IALS, to accelerate planning. In our case however, the fidelity of the IALS may change over time due to the training of the influence predictor, and also depends on the current context. While our simulator selection mechanism is based on an accuracy measure that is theoretically inspired Congeduti et al. [2021], future work could incorporate ideas from multi-fidelity methods more explicitly.

There is also a body of work that applies abstraction methods inside MCTS Anand et al. [2015a,b, 2016], Bai et al. [2016], Hostetler et al. [2014], Jiang et al. [2014]. Such methods perform abstraction on the MCTS search tree: they aggregate search tree nodes, thus reducing the search tree. This is complementary to our technique, which speeds up the simulations themselves.

4.5 CONCLUSION

In this paper, we investigated two questions. First, given a global simulator (GS) of a complex environment formalized as a factored POMDP, can we learn online a faster approximate model, i.e., a so-called IALS? Second, can we balance between such a learning IALS and the GS for fast *and* accurate simulations to improve the planning performance?

The answers to these questions lead to a new approach called *self-improving simulators*, which selects between the learning IALS and the GS for every simulation, based on an accuracy measure for IALS that can be estimated online with the GS simulations. As the accuracy of the influence predictor \hat{l}_θ would be poor initially, the GS will be used mainly for simulations in earlier episodes, which can then also be used to train \hat{l}_θ . This leads to increasing simulation efficiency as the IALS becomes more accurate and is used more often, which improves planning performance (time and return) over time. Our approach has three main advantages against the two-phase approach by He et al. [2020], which trains the influence predictor \hat{l}_θ offline before using it for planning. First, the planning can start without pre-training. Second, learning \hat{l}_θ online prevents the distribution shift issue that occurs when transferring the \hat{l}_θ learned offline to make online predictions. Third, selecting the simulator online enables reverting back to the global simulator when the IALS is not sufficiently accurate for the current context.

4.6 APPENDIX

In this appendix, we provide proofs for Proposition 2, 3 and 4. In Section 4.6.2, we provide details of our experimental setup. In Section 4.6.4, we present some additional results.

4.6.1 PROOFS

Proposition 2 (Cross Entropy Equivalence). *For every simulated step $t \leq k \leq \mathcal{H} - 1$ of simulation i , the expected cross entropy between the true influence source distribution $I(S_k^{\text{src}}|D_k)$ and its approximation $\hat{I}_\theta(S_k^{\text{src}}|D_k)$ under the exact IALM can be equivalently expressed under the global model $\mathcal{M}_{\text{global}}$:*

$$\begin{aligned} & \mathbb{E}_{D_k \sim \Pr(\cdot|h_t, \pi^{\text{tree}_i}; \mathcal{M}_{\text{IALM}})} [H(I(S_k^{\text{src}}|D_k), \hat{I}_\theta(S_k^{\text{src}}|D_k))] \\ &= \mathbb{E}_{D_t \sim \Pr(\cdot|h_t; \mathcal{M}_{\text{IALM}})} \left[\mathbb{E}_{S_{t+1:k}^{\text{local}}, A_{t:k-1} \sim \Pr(\cdot|D_t, h_t, \pi^{\text{tree}_i}; \mathcal{M}_{\text{IALM}})} [H(I(S_k^{\text{src}}|D_k), \hat{I}_\theta(S_k^{\text{src}}|D_k))] \right] \quad (4.16) \\ &= - \mathbb{E}_{D_t \sim \Pr(\cdot|h_t; \mathcal{M}_{\text{IALM}})} \left[\mathbb{E}_{S_{t+1:k}^{\text{local}}, A_{t:k-1} \sim \Pr(\cdot|D_t, h_t, \pi^{\text{tree}_i}; \mathcal{M}_{\text{IALM}})} \left[\mathbb{E}_{S_k^{\text{src}} \sim I(\cdot|D_k)} [\log \hat{I}_\theta(S_k^{\text{src}}|D_k)] \right] \right] \quad (4.17) \end{aligned}$$

$$= - \mathbb{E}_{D_k, S_k^{\text{src}} \sim P(\cdot|h_t, \pi^{\text{tree}_i}; \mathcal{M}_{\text{global}})} [\log \hat{I}_\theta(S_k^{\text{src}}|d_k)] \quad (4.18)$$

$$= - \underbrace{\mathbb{E}_{D_t, S_t \sim \Pr(\cdot|h_t; \mathcal{M}_{\text{global}})}}_{\text{belief}} \left[\underbrace{\mathbb{E}_{S_{t+1:k}, A_{t:k-1} \sim \Pr(\cdot|S_t, h_t, \pi^{\text{tree}_i}; \mathcal{M}_{\text{global}})}}_{\text{dynamics}} [\log \hat{I}_\theta(S_k[S_k^{\text{src}}]|D_k)] \right] \quad (4.19)$$

(where $S_k[S_k^{\text{src}}]$ represents the S_k^{src} part of S_k and $D_k = D_t A_t S_{t+1}[S_{t+1}^{\text{local}}] \dots A_{k-1} S_k[S_k^{\text{local}}]$)

Proof. The key insight from this proposition is that we can construct a sampling distribution with the global model $\mathcal{M}_{\text{global}}$ that is equivalent to the distribution $\Pr(D_k|h_t, \pi^{\text{tree}_i}; \mathcal{M}_{\text{IALM}})$ under the exact IALM, which we cannot access. Below, we show that this is the case.

For every $d_k = (s_{0:k}^{\text{local}}, a_{0:k-1})$ where $a_{0:t-1}$ are from $h_t = (o_{0:t}, a_{0:t-1})$:

$$\Pr(d_k|h_t, \pi^{\text{tree}_i}; \mathcal{M}_{\text{IALM}}) = \Pr(s_{0:t}^{\text{local}}|h_t; \mathcal{M}_{\text{IALM}}) \Pr(s_{t+1:k}^{\text{local}}, a_{t:k-1}|d_t, h_t, \pi^{\text{tree}_i}; \mathcal{M}_{\text{IALM}}) \quad (4.20)$$

For the first part of the right-hand side, we have that

$$\begin{aligned} & \Pr(s_{0:t}^{\text{local}}|h_t; \mathcal{M}_{\text{IALM}}) \\ &= \frac{\Pr(o_{0:t}, s_{0:t}^{\text{local}}|a_{0:t-1}; \mathcal{M}_{\text{IALM}})}{\Pr(o_{0:t}|a_{0:t-1}; \mathcal{M}_{\text{IALM}})} \quad (4.21) \end{aligned}$$

(conditional probability)

$$= \frac{(\prod_{j=0}^{t-1} \mathcal{O}^{\text{local}}(o_{j+1}|a_j, s_{j+1}^{\text{local}})) \Pr(s_{0:t}^{\text{local}}|a_{0:t-1}; \mathcal{M}_{\text{IALM}})}{\Pr(o_{0:t}|a_{0:t-1}; \mathcal{M}_{\text{IALM}})} \quad (4.22)$$

$$= \frac{(\prod_{j=0}^{t-1} \mathcal{O}^{\text{local}}(o_{j+1}|a_j, s_{j+1}^{\text{local}})) \Pr(s_{0:t}^{\text{local}}|a_{0:t-1}; \mathcal{M}_{\text{IALM}})}{\Pr(o_{0:t}|a_{0:t-1}; \mathcal{M}_{\text{global}})} \quad (4.23)$$

(as the global model and the IALM induce the same history MDP)

$$= \frac{\Pr(s_0^{\text{local}}) \prod_{j=0}^{t-1} \mathcal{O}^{\text{local}}(o_{j+1}|a_j, s_{j+1}^{\text{local}}) \mathbb{E}_{S^{\text{src}} \sim I(\cdot|d_j)} [\mathcal{T}^{\text{local}}(s_{j+1}^{\text{local}}|s_j^{\text{local}}, S_j^{\text{src}}, a_j)]}{\Pr(o_{0:t}|a_{0:t-1}; \mathcal{M}_{\text{global}})} \quad (4.24)$$

$$= \frac{\Pr(s_0^{\text{local}}) \prod_{j=0}^{t-1} \mathcal{O}^{\text{local}}(o_{j+1}|a_j, s_{j+1}^{\text{local}}) \mathbb{E}_{S^{\text{src}} \sim \Pr(\cdot|d_j; \mathcal{M}_{\text{global}})} [\mathcal{T}^{\text{local}}(s_{j+1}^{\text{local}}|s_j^{\text{local}}, S_j^{\text{src}}, a_j)]}{\Pr(o_{0:t}|a_{0:t-1}; \mathcal{M}_{\text{global}})} \quad (4.25)$$

$$= \frac{\Pr(s_0^{\text{local}}) \prod_{j=0}^{t-1} \mathcal{O}^{\text{local}}(o_{j+1}|a_j, s_{j+1}^{\text{local}}) \Pr(s_{j+1}^{\text{local}}|d_j; \mathcal{M}_{\text{global}})}{\Pr(o_{0:t}|a_{0:t-1}; \mathcal{M}_{\text{global}})} \quad (4.26)$$

$$= \frac{\Pr(o_{0:t}, s_{0:t}^{\text{local}}|a_{0:t-1}; \mathcal{M}_{\text{global}})}{\Pr(o_{0:t}|a_{0:t-1}; \mathcal{M}_{\text{global}})} \quad (4.27)$$

$$= \Pr(s_{0:t}^{\text{local}}|h_t; \mathcal{M}_{\text{global}}) \quad (4.28)$$

$$= \Pr(d_t = (s_{0:t}^{\text{local}}, a_{0:t-1})|h_t = (o_{0:t}, a_{0:t-1}); \mathcal{M}_{\text{global}}) \quad (4.29)$$

This shows that the history of local states $s_{0:t}^{\text{local}}$ can be equivalently tracked in the global model as in the exact IALM.

For the second part of the right-hand side of Equation (4.20), we have that

$$\begin{aligned} & \Pr(s_{t+1:k}^{\text{local}}, a_{t:k-1}|d_t, h_t, \pi^{\text{tree}_i}; \mathcal{M}_{\text{IALM}}) \\ &= \sum_{o_{t+1:k}} \Pr(s_{t+1:k}^{\text{local}}, a_{t:k-1}, o_{t+1:k}|d_t, h_t, \pi^{\text{tree}_i}; \mathcal{M}_{\text{IALM}}) \\ &= \left(\sum_{o_{t+1:k}} \prod_{j=t}^{k-1} \pi^{\text{tree}_i}(a_j|h_j) \left(\mathbb{E}_{S_j^{\text{src}} \sim I(\cdot|d_j)} [\mathcal{T}^{\text{local}}(s_{j+1}^{\text{local}}|s_j^{\text{local}}, S_j^{\text{src}}, a_j)] \right) \right. \\ & \quad \left. \mathcal{O}^{\text{local}}(o_{j+1}|a_j, s_{j+1}^{\text{local}}) \right) \quad (4.31) \end{aligned}$$

$$\begin{aligned} &= \left(\sum_{o_{t+1:k}} \prod_{j=t}^{k-1} \pi^{\text{tree}_i}(a_j|h_j) \left(\mathbb{E}_{S_j^{\text{src}} \sim \Pr(\cdot|d_j; \mathcal{M}_{\text{global}})} [\mathcal{T}^{\text{local}}(s_{j+1}^{\text{local}}|s_j^{\text{local}}, S_j^{\text{src}}, a_j)] \right) \right. \\ & \quad \left. \mathcal{O}^{\text{local}}(o_{j+1}|a_j, s_{j+1}^{\text{local}}) \right) \quad (4.32) \end{aligned}$$

$$= \left(\sum_{o_{t+1:k}} \prod_{j=t}^{k-1} \pi^{\text{tree}_i}(a_j|h_j) \Pr(s_{j+1}^{\text{local}}|d_j, a_j; \mathcal{M}_{\text{global}}) \mathcal{O}^{\text{local}}(o_{j+1}|a_j, s_{j+1}^{\text{local}}) \right) \quad (4.33)$$

$$= \sum_{o_{t+1:k}} \Pr(s_{t+1:k}^{\text{local}}, a_{t:k-1}, o_{t+1:k}|d_t, h_t, \pi^{\text{tree}_i}; \mathcal{M}_{\text{global}}) \quad (4.34)$$

$$= \Pr(d_k|d_t, h_t, \pi^{\text{tree}_i}; \mathcal{M}_{\text{global}}) \quad (4.35)$$

This shows that the future of local states can be similarly simulated in the global model as in the exact IALM. Combining the first and second part, we have that

$$\Pr(d_k|h_t, \pi^{\text{tree}_i}; \mathcal{M}_{\text{IALM}}) = \Pr(d_t|h_t; \mathcal{M}_{\text{global}}) \Pr(d_k|d_t, h_t, \pi^{\text{tree}_i}; \mathcal{M}_{\text{global}}) \quad (4.36)$$

$$= \Pr(d_k|h_t, \pi^{\text{tree}_i}; \mathcal{M}_{\text{global}}) \quad (4.37)$$

This shows that given an action-observation history h_t and a partial search tree, the global model and the exact IALM induce the same distribution over histories of local states and actions D_k at simulated time step k . Furthermore, we can sample D_k from this distribution by performing particle filtering on (D_t, S_t) and doing forward simulation with the global simulator to obtain the rest of D_k .

Using this result, we have that

$$\begin{aligned} & \mathbb{E}_{D_k \sim \Pr(\cdot | h_t, \pi^{\text{tree}i}; \mathcal{M}_{\text{IALM}})} [H(I(S_k^{\text{src}} | D_k), \hat{I}_\theta(S_k^{\text{src}} | D_k))] \\ &= \mathbb{E}_{D_k \sim \Pr(\cdot | h_t, \pi^{\text{tree}i}; \mathcal{M}_{\text{IALM}})} \left[\mathbb{E}_{S_k^{\text{src}} \sim I(\cdot | D_k)} [-\log \hat{I}_\theta(S_k^{\text{src}} | D_k)] \right] \end{aligned} \quad (4.38)$$

$$= \mathbb{E}_{D_k \sim \Pr(\cdot | h_t, \pi^{\text{tree}i}; \mathcal{M}_{\text{IALM}})} \left[\mathbb{E}_{S_k^{\text{src}} \sim \Pr(\cdot | D_k; \mathcal{M}_{\text{global}})} [-\log \hat{I}_\theta(S_k^{\text{src}} | D_k)] \right] \quad (4.39)$$

$$= \mathbb{E}_{D_k \sim \Pr(\cdot | h_t, \pi^{\text{tree}i}; \mathcal{M}_{\text{global}})} \left[\mathbb{E}_{S_k^{\text{src}} \sim \Pr(\cdot | D_k; \mathcal{M}_{\text{global}})} [-\log \hat{I}_\theta(S_k^{\text{src}} | D_k)] \right] \quad (4.40)$$

$$= \mathbb{E}_{D_k, S_k^{\text{src}} \sim \Pr(\cdot | h_t, \pi^{\text{tree}i}; \mathcal{M}_{\text{global}})} [-\log \hat{I}_\theta(S_k^{\text{src}} | D_k)] \quad (4.41)$$

$$= \mathbb{E}_{D_t, S_t \sim \Pr(\cdot | h_t; \mathcal{M}_{\text{global}})} \left[\mathbb{E}_{S_{t+1:k}, A_{t:k-1} \sim \Pr(\cdot | S_t, h_t, \pi^{\text{tree}i}; \mathcal{M}_{\text{global}})} [-\log \hat{I}_\theta(S_k[S_k^{\text{src}}] | D_k)] \right] \quad (4.42)$$

(where $S_k[S_k^{\text{src}}]$ represents the S_k^{src} part of S_k and $D_k = D_t A_t S_{t+1}[S_{t+1}^{\text{local}}] \dots A_{k-1} S_k[S_k^{\text{local}}]$)

which concludes our derivation for estimating the cross entropy with the global simulator. \square

Proposition 3 (Entropy Lower Bound). *For every simulated step $t \leq k \leq H-1$ of simulation i , the expected entropy of the true influence source distribution $I(S_k^{\text{src}} | D_k)$ under the exact IALM can be lower bounded using the global model $\mathcal{M}_{\text{global}}$ by conditioning on the global state:*

$$\begin{aligned} & \mathbb{E}_{D_k \sim \Pr(\cdot | h_t, \pi^{\text{tree}i}; \mathcal{M}_{\text{IALM}})} [H(I(S_k^{\text{src}} | D_k))] \\ & \geq \underbrace{\mathbb{E}_{D_t, S_t \sim \Pr(\cdot | h_t; \mathcal{M}_{\text{global}})}}_{\text{belief}} \underbrace{\left[\mathbb{E}_{S_{t+1:k-1}, A_{t:k-1} \sim \Pr(\cdot | S_t, h_t, \pi^{\text{tree}i}; \mathcal{M}_{\text{global}})} \right]}_{\text{dynamics}} \underbrace{\left[H(S_k^{\text{src}} | S_{k-1}, A_{k-1}) \right]}_{\text{from 2DBN}} \end{aligned} \quad (4.43)$$

Proof.

$$\mathbb{E}_{D_k \sim \Pr(\cdot | h_t, \pi^{\text{tree}i}; \mathcal{M}_{\text{IALM}})} [H(I(S_k^{\text{src}} | D_k))] \quad (4.44)$$

$$= \mathbb{E}_{D_k \sim \Pr(\cdot | h_t, \pi^{\text{tree}i}; \mathcal{M}_{\text{global}})} [H(I(S_k^{\text{src}} | D_k))] \quad (4.45)$$

(as shown in the proof of Proposition 2)

$$= \mathbb{E}_{D_k \sim \Pr(\cdot | h_t, \pi^{\text{tree}i}; \mathcal{M}_{\text{global}})} [H(S_k^{\text{src}} | D_k; \mathcal{M}_{\text{global}})] \quad (4.46)$$

(by definition of I (Definition 8): $I(S_k^{\text{src}} | D_k) = \Pr(S_k^{\text{src}} | D_k; \mathcal{M}_{\text{global}})$)

$$= \mathbb{E}_{D_t, S_t \sim \Pr(\cdot | h_t; \mathcal{M}_{\text{global}})} \left[\mathbb{E}_{S_{t+1:k-1}, A_{t:k-1}, S_k^{\text{local}} \sim \Pr(\cdot | S_t, h_t, \pi^{\text{tree}i}; \mathcal{M}_{\text{global}})} [H(S_k^{\text{src}} | D_k; \mathcal{M}_{\text{global}})] \right] \quad (4.47)$$

(law of total expectation)

$$\geq \mathbb{E}_{D_t, S_t \sim \Pr(\cdot | h_t; \mathcal{M}_{\text{global}})} \left[\mathbb{E}_{S_{t+1:k-1}, A_{t:k-1}, S_k^{\text{local}} \sim \Pr(\cdot | S_t, h_t, \pi^{\text{tree}i}; \mathcal{M}_{\text{global}})} [H(S_k^{\text{src}} | D_k, S_{t+1:k-1}, A_{t:k-1}; \mathcal{M}_{\text{global}})] \right] \quad (4.48)$$

(adding more information does not increase entropy)

$$= \mathbb{E}_{D_t, S_t \sim \Pr(\cdot|h_t; \mathcal{M}_{\text{global}})} \left[\mathbb{E}_{S_{t+1:k}, A_{t:k-1}, S_k^{\text{local}} \sim \Pr(\cdot|S_t, h_t, \pi^{\text{tree}_i}; \mathcal{M}_{\text{global}})} \left[H(S_k^{\text{src}} | S_{k-1}, A_{t-1}, S_k^{\text{local}}; \mathcal{M}_{\text{global}}) \right] \right] \quad (4.49)$$

(Markov property)

$$= \mathbb{E}_{D_t, S_t \sim \Pr(\cdot|h_t; \mathcal{M}_{\text{global}})} \left[\mathbb{E}_{S_{t+1:k}, A_{t:k-1}, S_k^{\text{local}} \sim \Pr(\cdot|S_t, h_t, \pi^{\text{tree}_i}; \mathcal{M}_{\text{global}})} \left[H(S_k^{\text{src}} | S_{k-1}, A_{k-1}) \right] \right] \quad (4.50)$$

(here we use the assumption there is no inter-stage dependency between state variables)

$$= \mathbb{E}_{D_t, S_t \sim \Pr(\cdot|h_t; \mathcal{M}_{\text{global}})} \left[\mathbb{E}_{S_{t+1:k}, A_{t:k-1} \sim \Pr(\cdot|S_t, h_t, \pi^{\text{tree}_i}; \mathcal{M}_{\text{global}})} \left[H(S_k^{\text{src}} | S_{k-1}, A_{k-1}) \right] \right] \quad (4.51)$$

where $H(S_k^{\text{src}} | S_{k-1}, A_{k-1})$ is the entropy of the influence source state distribution given the previous state and action, which can be computed from the 2DBN. \square

4.6.2 DETAILS OF THE EXPERIMENTAL SETUP

In the following, we describe further details of the experimental setup, including the planning domains and the hyperparameters used. Our codebase for this research is open-sourced at <https://github.com/INFLUENCEorg/POMCP-SIS>.

4.6.3 DOMAINS

4

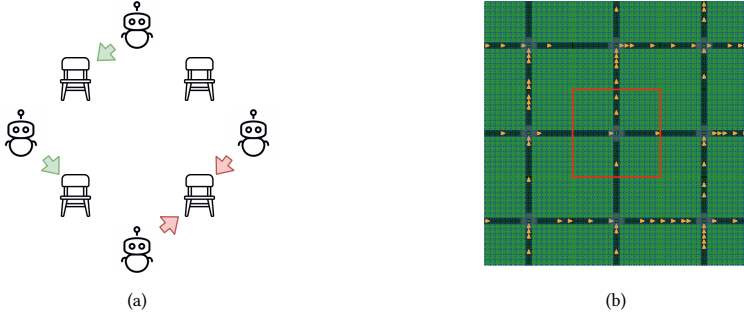


Figure 4.4: (a) An example of the grab a chair domain with 4 agents. For this time step, the agents on the top and left successfully obtain a chair and receive a reward 1 while the other two do not because they target the same chair. (b) A visualization of the grid traffic control domain with yellow arrows representing the moving cars. A red bounding box highlights the intersection controlled by the agent.

Grab a chair domain In our experiments, we make use of a grab a chair domain that lasts for 10 steps with one planning agent and 64 other agents executing fixed policies. At time step t , each agent has two possible actions which are grabbing the chair on its left and right side. When all the decisions are made, every agent which makes the decision to grab a chair that is not targeted by another neighboring agent will get the chair, and receive a reward of 1, otherwise none of the agents get the chair and both of them will receive 0 reward. The agent can only observe whether itself has obtained the chair or not, with a noise rate of 0.2. The fixed policy that is used by the 64 fixed agents make decisions based on the empirical frequency of obtaining a chair on the left and right side, according to their action-observation histories so far. See Figure 4.4a for an example.

Grid traffic control domain As illustrated by Figure 4.4b, the grid traffic control domain is made of 9 traffic intersections in total where the planning agent controls the one in the middle. At every time step t , the agent can take an action to switch the traffic light of the local intersection that it controls. The agent has access to a sensor data on if there are cars at the four grids that surround the traffic light. The reward of the agent is the negative total number of cars in this intersection at this time step. For the other 8 intersections in this intersection, they use a policy that switches their traffic lights every 9 time steps. New cars will enter the system with a probability 0.7 and cars at the borders of the system will

leave with a probability 0.3. Initially, cars are generated with a probability 0.7 in every grid. The horizon of this domain is 50.

HYPERPARAMETERS

The following hyperparameters are used for the experiments that correspond to Figure 4.2 (a-c) and Figure 4.3.

Influence Predictor In all experiments, the approximate influence predictor \hat{I}_θ is parameterized by a gated recurrent unit (GRU) Cho et al. [2014], a variant of recurrent neural networks, with 8 hidden states. We use Adam Kingma and Ba [2014] as the optimizer for training with stochastic gradient descent.

Training The online training of the influence predictors \hat{I}_θ occurs after every real episode. During every training, we perform 64 steps of stochastic gradient descent with the Adam optimizer. The data is sampled from the replay buffer that stores all the data collected so far, with a batch size of 128. The learning rates we use for the grab a chair and grid traffic control domains are 0.001 and 0.00025, respectively.

Planning In Table 4.1 and Table 4.2 we list the hyperparameters for planning with self-improving simulators in the grab a chair and grid traffic control domains.

Discount factor γ	1.0
Number of initial particles	1000
Exploration constant in the UCB1 algorithm c	100.0
Meta exploration constant for simulation selection c^{meta}	0.3

Table 4.1: Hyperparameters for planning with self-improving simulators in the grab a chair domain.

Discount factor γ	0.95
Effective Horizon	36
Number of initial particles	1000
Exploration constant in the UCB1 algorithm c	10.0
Meta exploration constant for simulation selection c^{meta}	0.1

Table 4.2: Hyperparameters for planning with self-improving simulators in the grid traffic control domain.

4.6.4 ADDITIONAL RESULTS

THE SIMULATION CONTROLLED PLANNING EXPERIMENTS

In Figure 4.5 we provide the additional results for the planning experiments with fixed number of simulations per step in the grab a chair domain. The left figure shows the learning curves of the influence predictor over real episodes. The right figure shows the estimated inaccuracy for the IALS. As we can see, due to training, generally the estimated inaccuracy is decreasing, which is expected. This leads to the increasing use of the IALS as

shown in the main text, thus speeding up planning. We can also see that with $\lambda = 3.0$ and $\lambda = 1.5$, the estimated inaccuracy is much lower in the beginning than the other λ s. This is because with a large λ , the accuracy threshold to use the IALS is much lower, which results that the untrained IALSs are used exclusively right from start. This limits the number of global simulations, causing poor accuracy of the inaccuracy estimation itself.

THE (TIME CONTROLLED) REAL-TIME PLANNING EXPERIMENTS

In Figure 4.6 and 4.7, we provide the additional results for the real-time planning experiments. The common trend is that, over real episodes, the training loss of the approximate influence predictor \hat{l}_θ (used in the IALS) decreases, which is translated into the decreasing inaccuracy estimate. This leads to more planning time spent on the IALS, which results in more simulations being done within the same time limit since it is significantly faster than the global simulator.

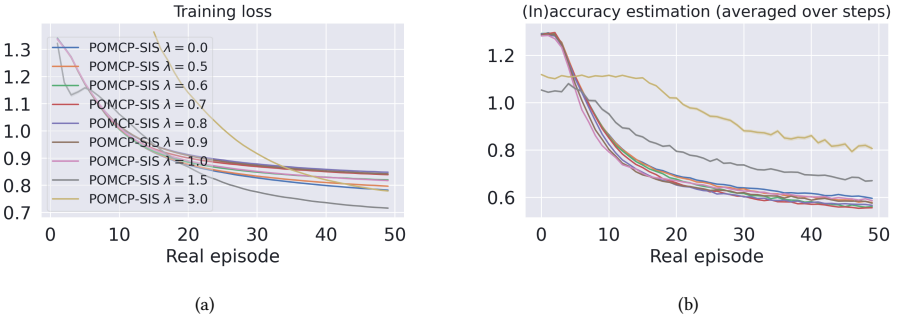


Figure 4.5: Additional results for the simulation controlled planning experiments in the grab a chair domain (accompanying Figure 4.2 (a-c)).

THE COMPARISON TO THE TWO-PHASE APPROACH

Figure 4.2d shows that the planning performance is dramatically different when training the influence predictors with the offline and online data. Note that in both cases, the influence predictors are trained offline and then used as-is for online planning. In the following we investigate the issue behind the failure of the influence predictors that are trained with data collected by a uniform random policy. To evaluate the influence predictors, we collect two independent datasets. One contains again the offline data that is collected by a uniform random policy, which should have the same distribution as the offline training data. To understand the behavior of the influence predictors when they make online predictions during planning, we execute POMCP with the global simulator to collect the second dataset. We consider the POMCP-GS dataset as the test set here because that represents the distribution that we may encounter when planning with an exact simulator. We plot the learning curves of the influence predictors that are trained on $\{10K, 20K, 50K, 100K\}$ episodes of offline data and on the "online" data that is collected when planning with the self-improving simulator $\lambda = 0.7$, importantly, evaluated on both datasets. The dotted lines with circles represent the evaluations with the offline dataset, and the solid lines with triangles represent the evaluations with the POMCP-GS dataset. We can see that first of

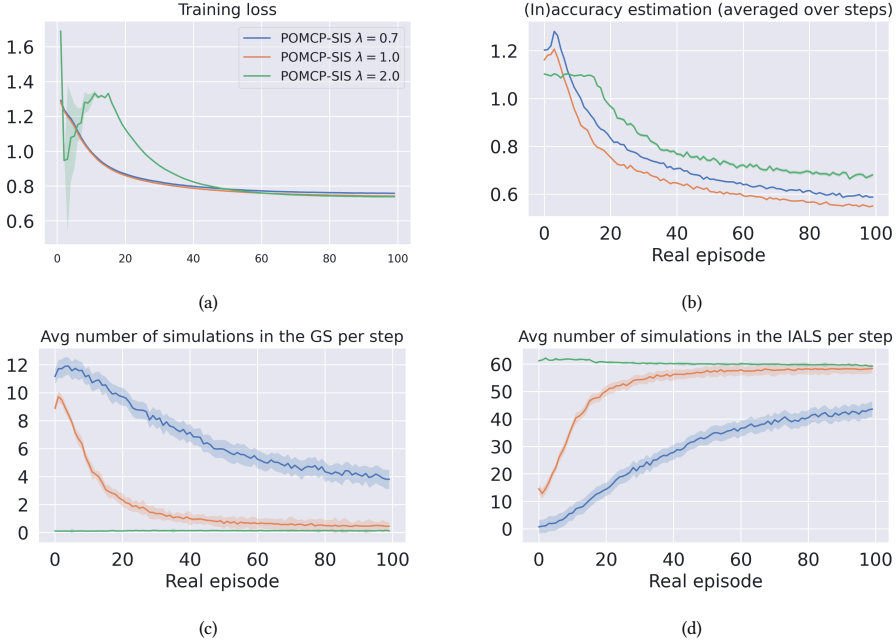


Figure 4.6: Additional results for the real-time planning experiments in the grab a chair domain (accompanying Figure 4.3a).

all, for all the influence predictors that are trained with offline data, there is a trend that as training goes on, the training error (evaluated on the offline dataset) is decreasing while the test error (on the POMCP-GS) dataset is increasing, a classic indicator of overfitting. This is strong evidence that there is indeed a distribution shift when training the influence predictors with data collected offline. Moreover, we see that this does not happen to the influence predictors that are trained with online data from the self-improving simulators. In the end, they can converge to a much lower test error, which can explain their much better planning performance. As such, we can conclude that this experiment demonstrates the distribution shift issue of the two-phase approach by He et al. [2020] and shows that planning with self-improving simulators can fix the issue.

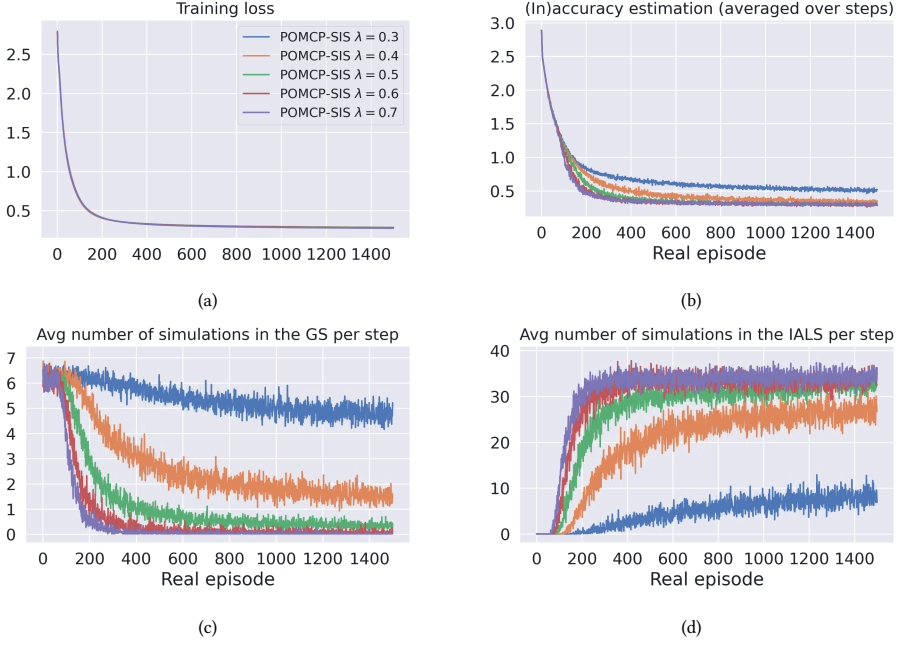


Figure 4.7: Additional results for the real-time planning experiments in the grid traffic control domain (accompanying Figure 4.3b).

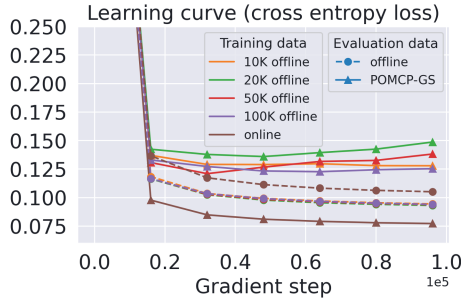


Figure 4.8: Learning curves of the influence predictors that are trained in an offline manner with the offline and online data (accompanying Figure 4.2d).

ABLATION STUDY: THE EFFECT OF THE META EXPLORATION CONSTANT

To understand the effect of the meta exploration constant c^{meta} , we repeat the simulation controlled experiment with a set of different values $c^{\text{meta}} = \{0.0, 0.3, 1.0, 2.0\}$. The results are shown in Figure 4.9. In the main text, we describe that there is an exploration and exploitation problem when selecting the simulators online, which we address with the UCB1 algorithm by formulating it as a bandit problem. Here we investigate the effect of the meta exploration constant on the planning performance. As we can see from Figure 4.9, with a meta exploration constant $c^{\text{meta}} = 0.0$, which effectively removes the UCB1 action selection, there is a period at the beginning of planning, during which the agent does not perform well, for many values of λ s. The number of IALS simulations suggests this is due to already the IALS a lot while it has not been trained much. However, this does not happen to the other values of the meta exploration constant. Our understanding is that this is due to the poor estimation of the IALS inaccuracy, i.e., a lack of exploration. On the other hand, from Figure 4.9, we can see that the use of a large meta exploration constant $c^{\text{meta}} = 2.0$ results in a limited exploitation of the faster IALS, even when it is sufficiently trained, preventing the planning speed to increase further after reaching certain level.

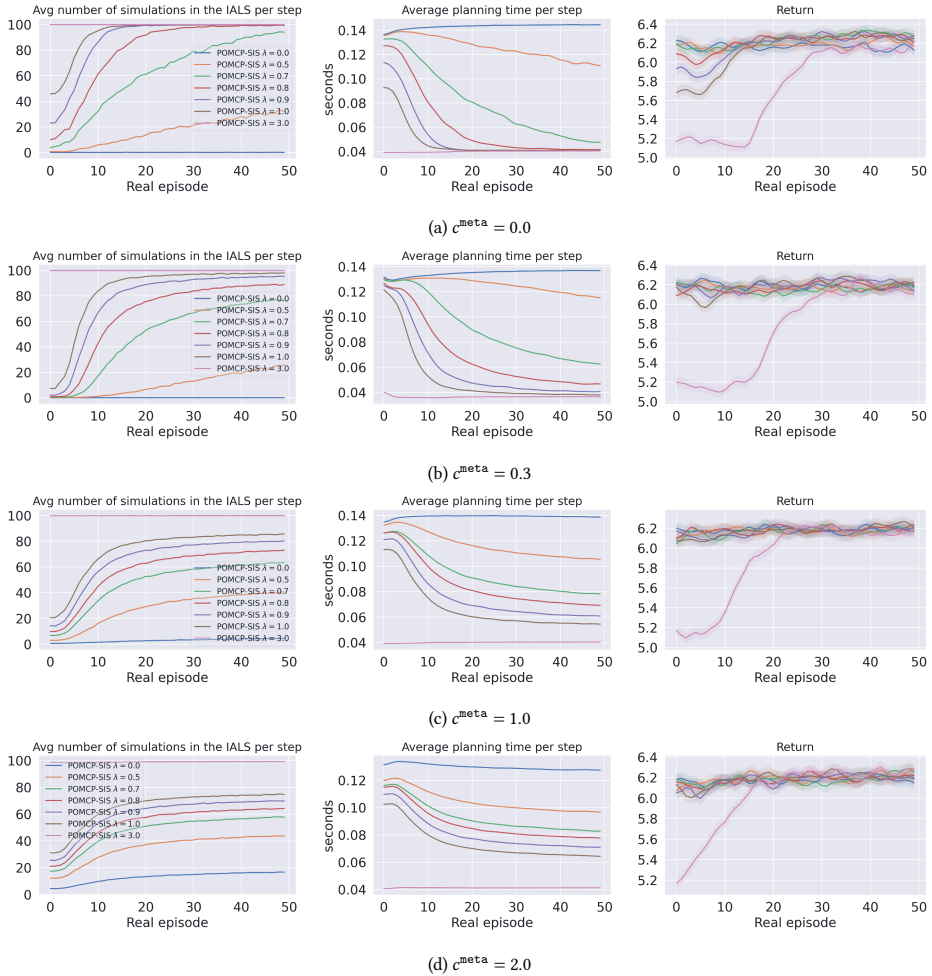


Figure 4.9: Simulation controlled planning experiments in the grab a chair domain with different values of the meta exploration constant.

5

WHAT MODEL DOES MUZERO LEARN?

5

"The purpose of abstracting is not to be vague, but to create a new semantic level in which one can be absolutely precise."

Edsger W. Dijkstra

Model-based reinforcement learning (MBRL) has drawn considerable interest in recent years, given its promise to improve sample efficiency. Moreover, when using deep-learned models, it is possible to learn compact and generalizable models from data. In this work, we study MuZero, a state-of-the-art deep model-based reinforcement learning algorithm that distinguishes itself from existing algorithms by learning a value-equivalent model. Despite MuZero's success and impact in the field of MBRL, existing literature has not thoroughly addressed why MuZero performs so well in practice. Specifically, there is a lack of in-depth investigation into the value-equivalent model learned by MuZero and its effectiveness in model-based credit assignment and policy improvement, which is vital for achieving sample efficiency in MBRL. To fill this gap, we explore two fundamental questions through our empirical analysis: 1) to what extent does MuZero achieve its learning objective of a value-equivalent model, and 2) how useful are these models for policy improvement? Among various other insights, we conclude that MuZero's learned model cannot effectively generalize to evaluate unseen policies. This limitation constrains the extent to which we can additionally improve the current policy by planning with the model.

5.1 INTRODUCTION

In recent years, deep reinforcement learning (DRL) [François-Lavet et al., 2018] has achieved remarkable progress, finding applications in a variety of real-world problems such as video compression [Mandhane et al., 2022], chip design [Mirhoseini et al., 2021], inventory management [Madeka et al., 2022] and plasma control for nuclear fusion [Degrave et al., 2022]. Despite these advancements, sample inefficiency remains a significant obstacle that limits the broader applicability of deep reinforcement learning in practical settings.

Model-based reinforcement learning (MBRL) [Moerland et al., 2023] addresses sample inefficiency in DRL by learning predictive models of the environment. In the typical RL cycle, an agent interacts with the environment (acting) and uses the collected data to refine its policy (learning). Accordingly, MBRL methods fall into two non-mutually exclusive categories [van Hasselt et al., 2019]: (1) those that use the learned model to improve acting and (2) those that use the learned model to improve learning.

One notable example of the first category is model-based exploration. In environments where rewards are sparse, shallow exploration techniques such as epsilon-greedy exploration often fail. Model-based exploration addresses this by not only allowing the agent to use model prediction errors or model uncertainty as intrinsic learning signals, but also enabling more effective exploration of "interesting" regions of the environment by "planning to explore" [Brafman and Tennenholtz, 2002, Henaff, 2019, Lowrey et al., 2018, Pathak et al., 2017, 2019, Sekar et al., 2020, Shyam et al., 2019]. In addition, decision-time planning methods such as Monte Carlo Tree Search (MCTS) [Browne et al., 2012] compute local policies online by planning with the model. A representative class of methods that use MCTS for decision-time planning is AlphaZero [Silver et al., 2016, 2017b, 2018], which defeated a human Go world champion for the first time in human history.

The other class of MBRL methods aims to improve the agent's policy and value functions without consuming additional data through model-based credit assignment. At a high level, these methods generate synthetic data through the learned model to simulate potential outcomes of specific actions or policies. This synthetic data is then used to update the agent's value estimates and improve the policy with a policy improvement operator. Notably, the DYNA architecture [Sutton, 1991] treats synthetic data as real data, integrating it into model-free learning algorithms, whereas the Dreamer methods [Hafner et al., 2019, 2020, 2023] use synthetic data exclusively for policy computation. In addition to using synthetic data directly, AlphaZero employs MCTS to compute a refined local policy and improved value estimates, which then serve as better learning targets for policy and value functions in replacement of model-free targets. Furthermore, van Hasselt et al. [2019] advocate for backward planning, which assigns credits to hypothetical states through a learned inverse model. They argue that planning backward for credit assignment can be more robust to model errors than planning forward, as updating fictional states can be less harmful than updating real states with fictional values.

Model-based exploration and credit assignment improve the data efficiency of RL methods by collecting more useful data and extracting more information from it. Beyond these two categories, learning a predictive model of the environment can also serve as an auxiliary task for representation learning [Hessel et al., 2022, Jaderberg et al., 2022]. In addition, in model-free planning, differentiable computation graphs that resemble the structure of

planning with a model (i.e., implicit models) have been found useful as architectural priors for value and policy functions, demonstrating improved performance in combinatorial planning domains while trained via model-free losses [Farquhar et al., 2018, Guez et al., 2018, 2019, Oh et al., 2017, Tamar et al., 2016].

Unlike tabular methods that learn the dynamics and reward for each state-action pair, deep model-based RL (DMBRL) approaches typically learn a state representation, on top of which dynamics and reward functions are estimated. However, determining what relevant information to include in these state representations and how to learn them effectively remains challenging. Namely, not all features of the observations are relevant [Li et al., 2006], and missing relevant features may lead to history dependence [McCallum, 1996]. A prevalent approach within DMBRL for learning these state representations is to model the next observation [Ha and Schmidhuber, 2018, Hafner et al., 2019, 2020, 2023, Kaiser et al., 2020]. However, accurately predicting high-dimensional observations, such as images, requires considerable effort in designing and training high-capacity neural networks with effective inductive biases [Kaiser et al., 2020]. This challenge has long been a barrier for MBRL methods, as the efficacy of model-based policy optimization strongly depends on the quality of these representations. Moreover, this approach often wastes significant representational power and training resources on encoding task-irrelevant information within the state representations, causing inefficiency in learning.

One approach to this challenge is the development of value-equivalent models [Farquhar et al., 2018, Grimm et al., 2020, 2021, 2022, Oh et al., 2017, Silver et al., 2017a, Tamar et al., 2016]. These models are specifically trained to predict the (multi-step) Bellman update, focusing solely on value-relevant aspects of the task dynamics without needing to reconstruct any observation. MuZero [Schrittwieser et al., 2020], a well-known MBRL algorithm, exemplifies this approach by achieving state-of-the-art performance in Atari games [Bellemare et al., 2013] and superhuman performance in Go, Chess, and Shogi.

MuZero inherits much of its structure from AlphaZero, which uses MCTS guided by both a learned policy network and a learned value network to make decisions and generate learning targets. However, MuZero distinguishes itself by integrating a model learned jointly with the value and policy networks, contrasting AlphaZero’s use of a ground-truth model for simulation and search. For clarity, throughout this work, we use the term learned model to refer specifically to the combination of MuZero’s learned dynamics and reward functions. Importantly, MuZero’s model is not trained to predict the next state or observation, but instead focuses on predicting task-relevant quantities such as future rewards, policies, and values. This approach of learning implicit models sets MuZero apart from traditional MBRL algorithms and promises a shift towards more efficient model learning. Despite MuZero’s empirical success and its considerable impact on MBRL [Antonoglou et al., 2021, Hubert et al., 2021, Mandhane et al., 2022, Ye et al., 2021], a recent study by [de Vries et al., 2021] shows that MuZero’s dynamics model can diverge significantly from real transitions, highlighting a gap in our understanding of how these models function and their efficacy in model-based credit assignment and planning. This discrepancy underscores the necessity for a detailed investigation into the capabilities and limitations of value-equivalent models within MuZero, which motivates this work.

The most relevant study in this direction is by Hamrick et al. [2022], who studied the role of planning in MuZero’s learning. They found that planning primarily boosts the

policy network’s learning by generating more informative data and constructing better training targets. Surprisingly, their findings also reveal that, in most domains, planning at evaluation time does not significantly improve performance compared to using the policy network alone, even with large search budgets. One explanation is that the policy network has converged to the optimal policy, rendering planning less useful. Another important consideration is that MuZero’s planning performance may not solely rely on its learned model: the value network, which is used to evaluate leaf nodes in MuZero’s MCTS, may also play a crucial role in driving strong empirical performance. Furthermore, Danihelka et al. [2021] show that improving MuZero’s planning enables strong performance even with extremely small search budgets ($n = 2$, or $n = 4$) in Go. This raises the question of what the contribution of MuZero’s learned model is.

In this study, we aim to bridge the gap in our understanding of MuZero’s learned model by exploring two fundamental questions:

1. To what extent does MuZero learn a value-equivalent model?
2. To what extent does MuZero learn a model that supports effective policy improvement (through planning)?

5

Learning a truly value-equivalent model is essential for model-based credit assignment, which directly influences the potential for improving existing policies through model-based planning. The more effectively we can improve existing policies through planning with the learned model, the greater the sample efficiency achieved by an MBRL method. As such, addressing these questions can help us better understand MuZero’s empirical success and inform the design of future algorithms or extensions.

Through our empirical analysis, we find that MuZero’s learned model is generally not accurate enough for policy evaluation, and the accuracy of the model decreases as the policy to evaluate deviates further from MuZero’s data collection policy. Consequently, this limits the extent to which we can find a good policy via planning. However, we find that MuZero’s incorporation of the policy prior in MCTS alleviates this problem, which biases the search towards actions where the model is more accurate. Based on these findings, we speculate that the role of the model in MuZero may be similar to that in model-free planning, providing a more powerful representation of value and policy functions, as the extent to which it can support policy improvement is rather limited. Moreover, using the policy prior indirectly takes model uncertainty into account during planning, which results in a form of regularized policy optimization with the learned model.

We introduce the essential background in Section 2.4. In Section 5.2, we study trained MuZero models in the policy *evaluation* setting (the objective for which the models were trained). In Section 5.3, we extend the analysis to the policy *improvement* setting (planning). Finally, we discuss the limitations and outlook of this work in Section 5.4.

5.2 POLICY EVALUATION EXPERIMENTS

We are interested in the extent to which MuZero’s learned model supports additional policy improvement, which is crucial for MuZero’s sample efficiency as an MBRL method. Our hypothesis is that, since MuZero’s model is trained on data collected by previous policies, it is not generally value equivalent for all policies, especially those that have not been executed. As accurate policy evaluation is the basis for effective policy improvement, this

will limit the extent to which MuZero can additionally improve its policy through planning. In this section, we validate this hypothesis.

5.2.1 TRAINING MUZERO AGENTS

For our empirical analysis, we used three fully observable deterministic environments, as MuZero was designed for this setting: Cart Pole, a deterministic version of Lunar Lander, and Atari Breakout [Bellemare et al., 2013]. We trained 30 MuZero agents with different random seeds for Cart Pole and Lunar Lander and 20 for Atari Breakout. For each agent, we saved the model weights at different training steps. In the figures below, we aggregate results from different seeds/agents and plot their means, with the corresponding standard errors represented as confidence intervals.

In Cart Pole and Lunar Lander, we extensively trained the MuZero agents for 100K and 1M steps. This way, we can conduct our analysis on the trained agents throughout their lifecycle of learning. For Atari Breakout, we adopted the same setup as EfficientZero [Ye et al., 2021] but extended the training from 100K steps to 500K steps.

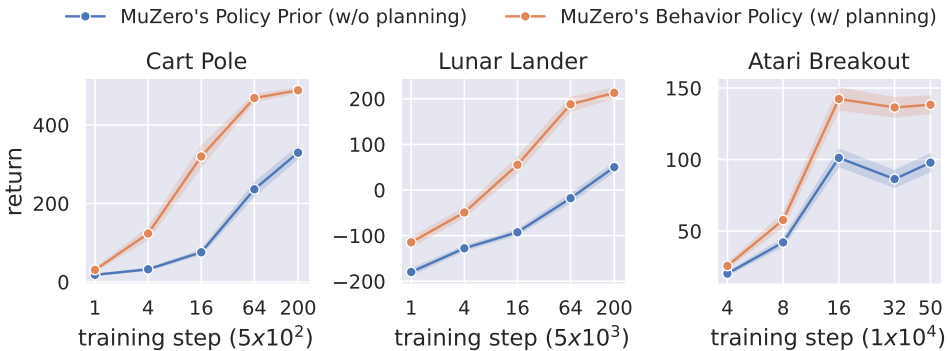


Figure 5.1: Online performance of MuZero agents in Cart Pole (Left), Lunar Lander (Middle), and Atari Breakout (Right). For the same agent, we plot the performance of MuZero’s behavior policy (with planning) and policy prior (without planning). Note that the X-axes are in logarithmic scales. Planning at decision time substantially improves the performance of an agent, compared to taking actions directly from the policy prior.

Figure 5.1 shows the online performance of MuZero agents at various training steps, which make decisions by performing MCTS planning (Equation 2.34). For reference, we also plot the performance of the policy prior of the same agent, which samples actions directly from the policy network, $a_t \sim \pi_\theta(\cdot|h_\theta(s_t))$. Throughout training, planning enables MuZero to achieve a substantially better performance than directly using the policy prior. It is important to note that the improved performance may not only come from the learned model but may also come from the value network, which is used to estimate the value of leaf nodes in MuZero’s MCTS. In the following, we will take a deeper look at how much the learned model contributes to this.

5.2.2 EVALUATING THE LEARNED MODEL

To assess MuZero’s learned model in the context of policy evaluation, we will compare it to the ground-truth model. Through experiments, we aim to answer questions in the format of *how much value prediction error should we expect when using MuZero’s learned model to evaluate a policy π ?*

Considering that MuZero does not train or employ the model for infinite-horizon rollouts, we impose a limit on the evaluation horizon when assessing the value prediction error. For each state s_t , we define the discounted sum of future rewards for taking an action sequence (a_t, \dots, a_{t+h-1}) as:

$$v^{a_{t:t+h-1}}(s_t) = \sum_{k=0}^{h-1} \gamma^k r_{t+k} \quad (5.1)$$

where $r_{t+k} = \mathcal{R}(s_{t+k}, a_{t+k})$ and $s_{t+k+1} = \mathcal{T}(s_{t+k}, a_{t+k})$, assuming the environment is deterministic. As mentioned in Section 2.4, MuZero predicts the value of this action sequence by first encoding the state into a latent state $z_t^0 = h_\theta(s_t)$ and then rolling out the model from the latent state:

$$\hat{v}^{a_{t:t+h-1}}(s_t) = \sum_{k=0}^{h-1} \gamma^k u_t^k \quad (5.2)$$

where $z_t^{k+1}, u_t^k = g_\theta(z_t^k, a_{t+k})$. Then, we can define the value prediction error of the learned model for the action sequence $a_{t:t+h-1}$.

Definition 10. *The value prediction error of using the learned model to evaluate the action sequence $a_{t:t+h-1}$ at state s_t is:*

$$|v^{a_{t:t+h-1}}(s_t) - \hat{v}^{a_{t:t+h-1}}(s_t)| \quad (5.3)$$

As a stationary policy $\pi : \mathcal{S} \rightarrow \Delta(\mathcal{A})$ that operates in the original environment defines a distribution over such action sequences:

$$\Pr(a_{t:t+h-1} | s_t, \pi) = \prod_{k=0}^{h-1} \pi(a_{t+k} | s_{t+k} = \mathcal{T}(s_{t+k-1}, a_{t+k-1})) \quad (5.4)$$

We can define the value prediction error for evaluating π .

Definition 11. *The value prediction error of using the learned model to evaluate a stationary policy π , which operates in the original environment, for horizon h is:*

$$|v_h^\pi(s_t) - \hat{v}_h^\pi(s_t)| = |\mathbb{E}_{a_{t:t+k-1} \sim \Pr(\cdot | s_t, \pi)} [v^{a_{t:t+h-1}}(s_t) - \hat{v}^{a_{t:t+h-1}}(s_t)]| \quad (5.5)$$

5.2.3 HOW ACCURATELY CAN MUZERO’S LEARNED MODEL PREDICT THE VALUE OF ITS OWN BEHAVIOR POLICY?

As the model is trained on data collected by MuZero’s behavior policy π^{MuZero} , we expect it to be at least accurate on this data collection policy. Therefore, we begin our investigation by examining the model’s prediction performance on this policy. Due to the continuous

state spaces of our environments, it is not feasible to enumerate all states and compute the error for each. Instead, we sample states from MuZero’s on-policy state distribution $d_{\pi^{\text{MuZero}}}$. For each sampled state, we conduct the evaluation and aggregate the errors. To facilitate tractable evaluation, at every state s , we use Monte Carlo sampling to estimate both the true value $v_h^{\pi^{\text{MuZero}}}(s)$ and the value predicted by MuZero’s model $\hat{v}_h^{\pi^{\text{MuZero}}}(s)$.

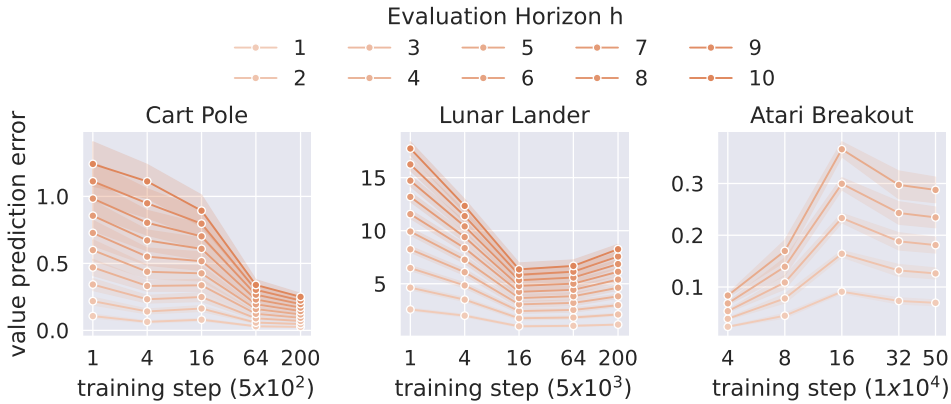


Figure 5.2: Value prediction error of using MuZero’s learned model to evaluate its own behavior policy.

In Figure 5.2, we report the value prediction error (Y-axis) under various evaluation horizons (lines) across different training steps (X-axis). The maximum evaluation horizons are set to the number of unrolling steps during training: 10, 10, 5 for Cart Pole, Lunar Lander, and Atari Breakout, respectively. In all environments, value predictions for short horizons are highly accurate, with errors approaching zero. However, as the evaluation horizon increases, the errors consistently grow larger. On the one hand, this is not surprising because learned models are known prone to accumulate errors during long rollouts [Lambert et al., 2022]. On the other hand, this shows that models learned by MuZero cannot be fully value equivalent as they are not even accurate enough to predict values for the policy that collects the training data. Errors at different training steps are generally not comparable due to the different state distributions, but the decreasing errors observed at the end of training suggest the convergence of the policy as a possible explanation.

5.2.4 HOW ACCURATELY CAN MUZERO’S LEARNED MODEL EVALUATE POLICIES THAT ARE DIFFERENT FROM THE BEHAVIOR POLICY?

To assess whether MuZero’s learned model effectively supports planning, we will investigate its capacity to generalize and accurately predict values beyond its own data collection policy. Our hypothesis is that the model will exhibit increasing inaccuracies when evaluating policies that differ significantly from the behavior policy, which is responsible for collecting the training data.

To test this hypothesis, we conduct an experiment focusing on the relationship between the value prediction error for an action sequence $|v^{a_t:t+h-1}(s_t) - \hat{v}^{a_t:t+h-1}(s_t)|$ and the

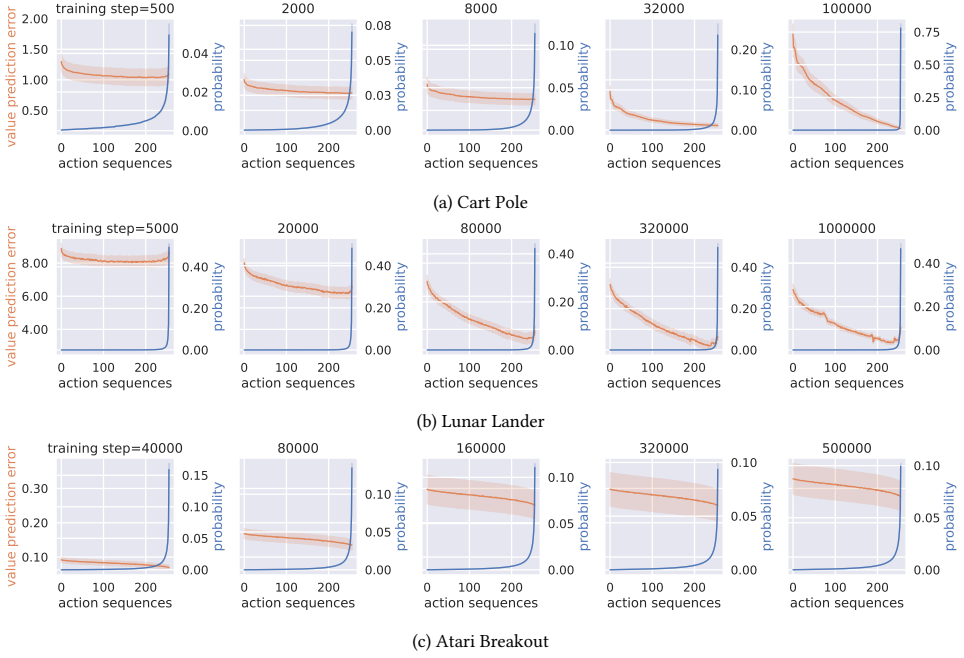


Figure 5.3: X-axis: action sequences sorted by their probabilities of being taken by the behavior policy (from unlikely to likely reading from left to right). Y-axis: the probabilities (blue, from small to large) and the corresponding value prediction errors (yellow, from small to large). Action sequences with higher probabilities to be taken by MuZero’s behavior policy correlate with lower value prediction errors by MuZero’s learned model. This implies that the learned model is less accurate for policies that are different from the data collection policy.

probability of the behavior policy selecting this action sequence $\Pr(a_{t:t+h-1}|s_t, \pi^{\text{MuZero}})$. We again sample states from MuZero’s on-policy state distribution $d_{\pi^{\text{MuZero}}}$ and compute the probabilities and value prediction errors for *all* action sequences of length $\{8, 4, 4\}$ for Cart Pole, Lunar Lander and Breakout (limited by computation budget). Considering that both probabilities and errors are real-valued and non-uniformly distributed, we aggregate results as follows: first, for each state, we rank action sequences by their probabilities of being chosen by MuZero’s behavior policy. Then, we compile statistics for action sequences with the same ranks. Finally, we combine results from different agents and report their means and standard errors.

In Figure 5.3, we present the results. Here, the X-axis represents the action sequences that are sorted by their likelihood of being taken by the current behavior policy, ranging from unlikely to likely. On the Y-axis, we plot both the probabilities (blue) and the corresponding value prediction errors (yellow). The results clearly show that, as the likelihood of the behavior policy selecting the action sequence decreases, the value prediction error for that action sequence increases. This trend is consistently observed across different environments and training steps. Moreover, it seems to become more evident with more training, possibly because the behavior policy becomes more deterministic. This finding supports our hypothesis that the model is more reliable in predicting values for the behavior

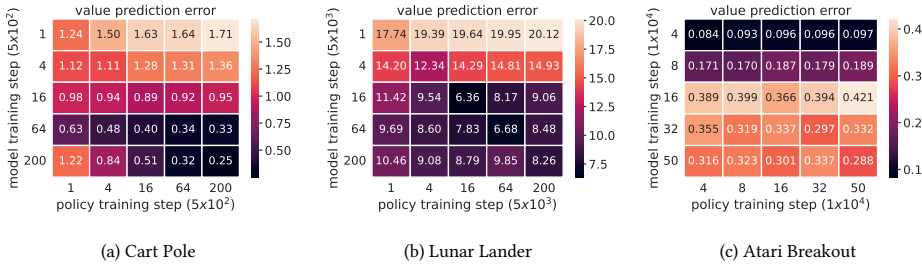


Figure 5.4: Cross model policy evaluation. We evaluate MuZero’s behavior policy at training step Y (column) with the learned model at training step X (row) and measure the value prediction error. Results are aggregated over states sampled from MuZero’s on-policy state distribution at training step X (same as the model).

policy, which collects the training data. Consequently, evaluating a policy using MuZero’s learned model may yield increasingly inaccurate results as the policy to evaluate deviates further from the behavior policy.

5.2.5 HOW ACCURATELY CAN MUZERO’S LEARNED MODEL FROM ONE TRAINING STEP EVALUATE THE BEHAVIOR POLICY OF THE SAME AGENT FROM ANOTHER TRAINING STEP?

In our previous experiment, we investigated the generalization capability of the learned model in predicting values for policies that differ from the behavior policy. We accomplished this by considering all action sequences of length h at each state, possibly including actions that are unlikely to be taken by any sensible policy. In this experiment, we conduct a similar analysis but with a focus on more interesting policies. Specifically, we assess the accuracy with which the model at training step X (row) can evaluate the policy (of the same agent) at training step Y (column). The idea is that if a model cannot accurately evaluate high-performing future policies, planning with it would not be effective in finding a good policy. For this experiment, we set the evaluation horizon for each environment to the number of unrolling steps during training. Results are aggregated over states sampled from MuZero’s on-policy state distribution at training step X (same per row as the model). Note that errors at different model steps (rows) are not directly comparable due to the different state distributions.

It is clear from Figure 5.4 that models at all training steps are most accurate when evaluating policies at the same training steps, as the error per row is smallest at the diagonal. This aligns with our earlier finding: learned models are more accurate when assessing action sequences with a higher selection likelihood by the current behavior policy. Notably, we can see that using early models to evaluate future policies results in large errors in Cart Pole and Lunar Lander. This may have important implications for policy improvement: *if the model at the current training step cannot accurately evaluate a future policy that performs better, then the extent to which we can improve our current policy will be limited.* Intuitively, if the model does not know a high-performing policy is good, then a deeper search in the model would not help us find the policy. This phenomenon is less evident in Breakout,

possibly because the evaluation horizon is too short in this environment to allow for an observable difference in rewards across the policies (5 vs 10 in the other environments).

5.3 POLICY IMPROVEMENT EXPERIMENTS

Our policy evaluation experiments indicate that MuZero’s learned model can become increasingly inaccurate when evaluating policies differing from the data collection policy, particularly those unseen during training. In this section, we investigate the natural follow-up question, the question that we are most interested in: what is the effect of this on policy improvement through planning?

Intuitively, if the agent is given the ground-truth model and has an infinite budget for planning, it can act optimally by exhaustively searching with the model. Sample efficiency is maximized in this case as the agent needs no sample from the environment to learn. However, in real-world scenarios, the planning budget is always constrained, both during training and deployment. To improve planning with these constraints, AlphaZero and MuZero employ a learned policy prior to guide the action simulation in MCTS. If the policy prior is well-trained, it can greatly enhance planning, but if not, it might harm efficiency.

We evaluate planning using MuZero’s learned model both with and without the guidance of the policy prior. In the latter case, we replace the policy prior with a uniform prior, allowing for a form of “free search”. To focus on evaluating the contribution of the learned model, we modify MuZero’s MCTS by replacing the value network’s predictions at leaf nodes with random rollouts in the model, a standard approach to estimating the value of a leaf node in MCTS. To address the computational costs of long rollouts in neural models, we restrict the planning horizon to $\{16, 128, 32\}$ in Cart Pole, Lunar Lander, and Atari Breakout. For comparison, we include the baselines of the policy prior and planning with the ground-truth model using the same MCTS. The goal of this experiment is to answer two questions:

- (a) How effectively does MuZero’s learned model support free search?
- (b) To what extent can MuZero improve its policy by planning with the learned model?

We present our results in Figure 5.6. In Cart Pole, MuZero’s learned model exhibits some degree of support for free search (orange dashed) after some training. However, it significantly underperforms compared to free search with the ground-truth model (green dashed) in terms of both planning efficiency and asymptotic performance. In Lunar Lander and Atari Breakout, free search with the learned model (orange dashed) fails completely. Consequently, the potential for finding a good policy through planning with the learned model will be effectively limited.

In Cart Pole and Lunar Lander, with enough MCTS simulations, planning with the learned model improves performance over using the policy prior alone. However, compared to the ground-truth model, it is evident that the extent to which we can improve upon the policy prior via planning is still very limited. This is a clear sign that the model error is restricting the extent to which we can further improve the policy by planning. In Atari Breakout, while planning with more simulations using the learned model improves the agent’s performance, it cannot outperform the policy prior.

Interestingly, when comparing planning with the policy prior to planning with the uniform prior, the learned model consistently shows a larger gap compared to the ground-truth model except in Atari Breakout. Furthermore, unlike the ground-truth model, the gap for the learned model does not seem to diminish quickly with more simulations, suggesting an additional role of the policy prior to accelerating MCTS. As shown in Figure 5.3, action sequences with higher probabilities of being selected by the behavior policy tend to be more accurately evaluated by the learned model. As the policy prior is directly learned to match the behavior policy, it is reasonable to assume that the values of action sequences favored by the policy prior can also be more accurately predicted by the learned model.

To verify this hypothesis, in Figure 5.5, we plot the learned model's value prediction error for MCTS's simulated trajectories under the guidance of the policy prior (blue) and uniform prior (orange) in CartPole after 2000 training steps. Meanwhile, we also plot the total variation $TV[\pi_\theta, \hat{\pi}]$ and KL divergence $KL[\pi_\theta, \hat{\pi}]$ between the policy prior π_θ and MCTS's empirical visit distribution $\hat{\pi} = \frac{1+N(a)}{\sum_b |A|+N(b)}$ at the root node. Clearly, the policy prior regularizes MCTS to visit actions that are more favored by it, as suggested by the lower total variation and KL divergence (blue, middle, and right), which results in a smaller value prediction error (blue, left), when compared to the uniform prior, which explores more out-of-the-box (orange, middle, and right) and incurs more value prediction errors (orange, left). This suggests that *apart from biasing the search, the policy prior may also serve to prevent the search from exploring directions where the learned model is less accurate.*

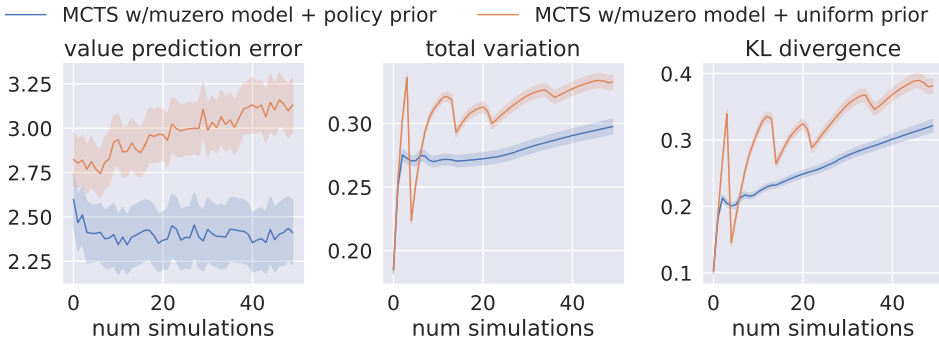


Figure 5.5: Value prediction error of MCTS's simulated trajectories using the learned model (left), the total variation between the policy prior and MCTS's empirical visit distribution at the root node (middle), and the KL divergence between them (right).

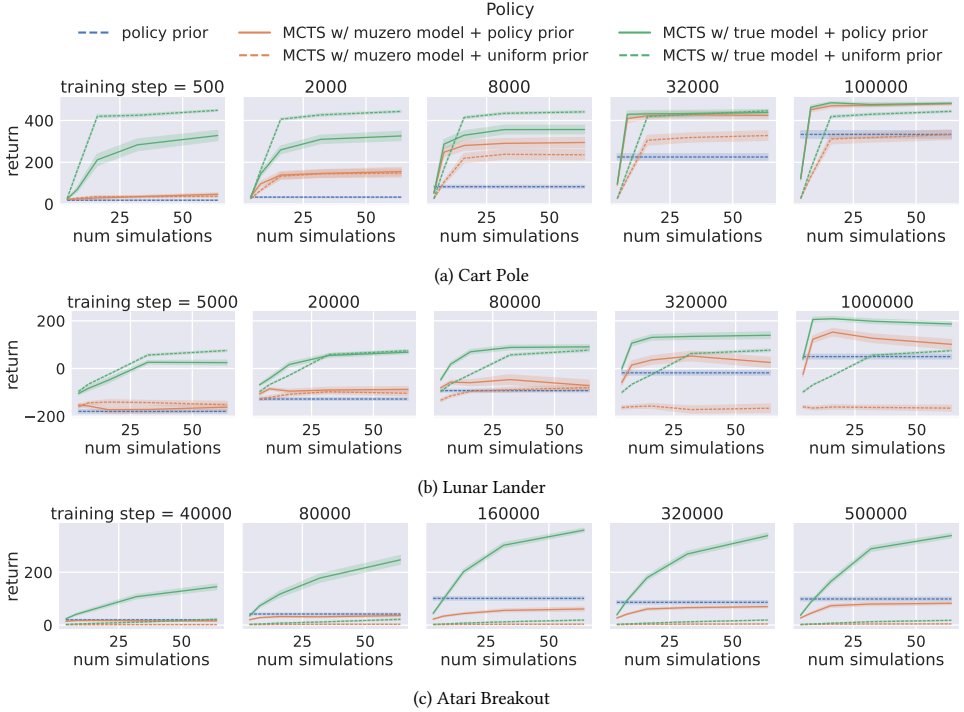


Figure 5.6: MCTS planning with (i) MuZero’s learned model (orange) and (ii) the ground-truth model (green), under the guidance of (a) MuZero’s policy prior (solid line) and (b) the uniform prior (dotted line)). The X and Y axes are the number of simulations per step and the return.

5.4 DISCUSSION

Limitations In this study, we analyzed MuZero, one of the most successful DMBRL methods that is grounded in the value equivalence principle. The conclusions we draw can, therefore, not be directly generalized to other DMBRL methods, certainly not those that employ other auxiliary loss functions on top, e.g., [Gelada et al., 2019, van der Pol et al., 2020a, Ye et al., 2021]. However, our analysis clearly shows that, despite its simplicity, using the value equivalence principle does not imply that we learn models that are actually value equivalent. Even when restricting the predictions to its own behavior policy, MuZero’s learned model leads to prediction errors that quickly grow with the horizon of prediction. As such, our study does serve as a warning for other DMBRL methods that aim to use value equivalence as their guiding principle.

Outlook We have demonstrated that MuZero’s model, trained with a value-equivalence-based loss, struggles to generalize and predict values accurately for unseen or unfamiliar policies. This raises the question of how different losses for model learning would behave

and compare in our analysis, such as the reconstruction-based loss [Hafner et al., 2019, 2020, 2023, Kaiser et al., 2020] and the temporal-consistency [de Vries et al., 2021, van der Pol et al., 2020a, Ye et al., 2021]. Specifically, we hypothesize that, while the value equivalence loss is the most flexible loss as it does not directly impose requirements on state representations, other losses may have an advantage for generalization in low-data regimes due to a richer supervision signal. We consider this potential trade-off between representation complexity and generalization an exciting direction for future research.

5.5 CONCLUSION

In this work, we empirically studied the models learned by MuZero, which are trained based on the value equivalence principle. Our analysis focused on two fundamental questions: (1) To what extent does MuZero learn a value-equivalent model? and (2) To what extent does the learned model support effective policy improvement by planning? We find that MuZero’s learned model cannot generally evaluate policies accurately, especially those that further deviate from the data collection policy. Consequently, the failure of the model to predict values for policies that are out of the training distribution prevents effective planning from scratch, which limits the extent to which MuZero can additionally improve its policy via planning. Moreover, we uncover that apart from accelerating the search as in AlphaZero, the policy prior in MuZero serves another crucial function: it regularizes the search towards areas where the learned model is more accurate, which effectively reduces the model error that is accumulated into planning. From these findings, we speculate that because MuZero’s model itself is limited in its ability for policy improvement, the empirical success of MuZero may be a result of the model providing the algorithm with a more powerful representation of the values and policies compared to single-lookahead methods like deep Q-learning [Mnih et al., 2015]. This implies a role for the model that is similar to that in model-free planning but extends it with the policy prior to make planning conservative [Kumar et al., 2020].

5.6 APPENDIX

In this appendix, we describe the training of MuZero agents in detail and present additional results.

5.6.1 TRAINING OF MUZERO AGENTS

SETUP AND HYPERPARAMETERS

Hyperparameter	Cart Pole	Lunar Lander
Random seeds	0 to 29	0 to 29
Discount factor	0.997	0.999
Total training steps	100000 (10000)	1000000 (200000)
Optimizer	Adam	Adam
Initial Learning Rate	0.02	0.005
Learning Rate Decay Rate	0.1 (0.8)	No decay
Learning Rate Decay Steps	50000 (1000)	No decay
Weight Decay	1e-4	1e-4
Momentum	0.9	0.9
Batch Size	128	64
Encoding size	8	10
Fully-connected Layer Size	16	64
Root Dirichlet Alpha	0.25	0.25
Root Dirichlet Fraction	0.25	0.25
Prioritized Experience Replay Alpha	0.5	0.5
Num Unroll Steps	10	10
TD Steps	50	30
Support Size	10	10
Value Loss Weight	1.0	1.0
Replay Buffer Size	500	500 (2000)
Visit Softmax Temperature Fn	1.0 \rightarrow (5e4) 0.5 \rightarrow (7.5e4) 0.25	0.35

Table 5.1: Hyperparameters for training MuZero agents in Cart Pole and Lunar Lander. We used default values from [Duvaud and Hainaut, 2019] for most of the hyperparameters. The bold values are those that we tuned to improve the convergence of the agents, with the default values shown in brackets.

For Cart Pole and Lunar Lander, we trained 30 MuZero agents using an open-source implementation of MuZero [Duvaud and Hainaut, 2019], which is available on GitHub (under the MIT license). The implementation has been extensively tested on various classic RL environments, including Cart Pole and Lunar Lander. While we mostly used the default recommended hyperparameter values from [Duvaud and Hainaut, 2019], we fine-tuned a few of them to improve the convergence of agents. The comprehensive list of hyperparameters for training MuZero agents in these environments can be found in Table 5.1. For Atari Breakout, we trained 20 MuZero agents using the official implementation of EfficientZero [Ye et al., 2021], *excluding* the additional improvements introduced by

Hyperparameter	Atari Breakout
Random seeds	0 to 9
Total training steps	500000
Learning Rate	0.1 \rightarrow 0.01 with an exponential decay rate of 0.1
Replay Buffer Size	100000
Visit Softmax Temperature Fn	1.0 (fixed throughout training)

Table 5.2: Hyperparameters for training MuZero agents in Atari Breakout. We used default values from EfficientZero [Ye et al., 2021] (see Appendix A.1 Table 6 of [Ye et al., 2021]) for those hyperparameters that are not mentioned in the table.

EfficientZero. This implementation is also available on GitHub (under the GPL-3.0 license). At the time of this research, this was the only plausible way to train MuZero agents in Atari games. See Table 5.2 for the hyperparameters.

Regarding computation, training each MuZero agent took around 3 hours for Cart Pole and 20 hours for Lunar Lander, using 8 CPUs. For Atari Breakout, it took around 40 hours per agent using 2 GPUs and 48 CPUs. The training was conducted on a shared internal cluster equipped with a variety of CPUs and GPUs (Nvidia 2080Ti/V100/A40).

5

LEARNING CURVES

In Figure 5.1, we plot the online performance of MuZero agents at various training steps for running both the policy prior and the behavior policy. There, actions are sampled from policies. In Figure 5.7, we plot the full learning curves with actions both sampled and taken greedily from the policies. When running MuZero’s behavior policy greedily, we do not add Dirichlet noise to the policy prior in the tree search.

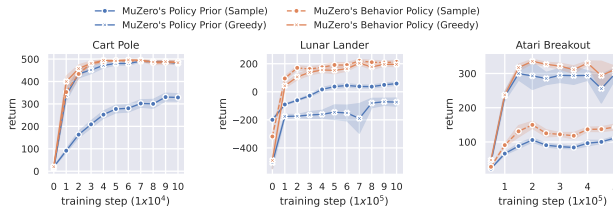


Figure 5.7: Full learning curves of MuZero agents in Cart Pole (Left), Lunar Lander (Middle), and Atari Breakout (Right).

5.6.2 ADDITIONAL RESULTS

POLICY EVALUATION EXPERIMENTS WITH LONGER EVALUATION HORIZONS

In the main paper, we used relatively short evaluation horizons in the policy evaluation experiments because the models were only unrolled for these numbers of steps during training: 10, 10, 5 for Cart Pole, Lunar Lander, and Atari Breakout, respectively. Here, we present results using a longer evaluation horizon of 50 steps.

Specifically, we replicate the experiment from Section 5.2.5 and Figure 5.4, where we measure the value prediction error using the model learned at training step X to evaluate MuZero’s behavior policy at training step Y , but with an extended evaluation horizon of 50 steps across all domains. The results are shown in Figure 5.8.

Compared to Figure 5.4, which uses shorter evaluation horizons, the value prediction errors here are significantly larger due to compounding errors. However, the overall trend remains consistent: models and policies at different training steps generally show incompatibility in value prediction. This trend is more pronounced here, as the value discrepancies between different policies are more evident with a longer horizon.

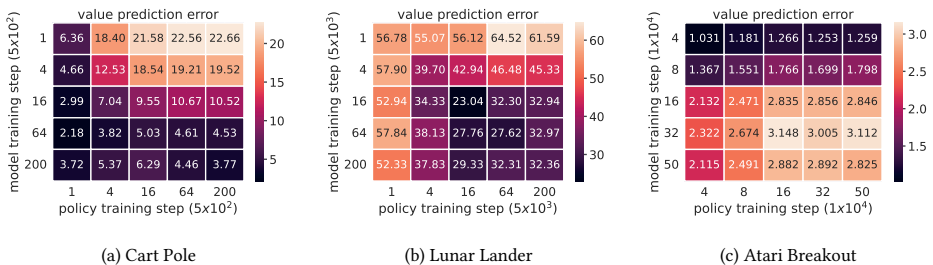


Figure 5.8: Cross model policy evaluation with an evaluation horizon of 50. We evaluate MuZero’s behavior policy at training step Y (column) with the learned model at training step X (row) and measure the value prediction error. Results are aggregated over states sampled from MuZero’s on-policy state distribution at training step X (same as the model).

6

CONCLUSION

"From the first stirrings of life beneath water... to the great beasts of the Stone Age... to humankind taking their first upright steps, you have come far. Now begins your greatest quest: from this early cradle of civilization on towards the stars."

Civilization VI

6

In this chapter, we describe the big picture of automated decision-making, summarize the contributions of this thesis, and outline potential directions for future work.

In this thesis, we set out to improve decision-making in complex environments using abstractions. This raises the question: where do we now stand in the broader context of supported and automated decision-making? Here, we sketch a big-picture overview of the past and current state of supported/automated decision-making and what we see as the main challenges moving forward.

6.1 THE BIG PICTURE: THE EVOLUTION OF HUMAN DECISION-MAKING

The history of humankind is, in many ways, the history of decision-making. From the earliest days, humans have been observing their environment, processing information, and making decisions to shape the world according to their needs and desires. Initially driven by the imperative to survive—seeking food, shelter, and safety—and later by ambition and curiosity, our capacity for decision-making has evolved significantly over time.

6.1.1 EXTENDING HUMAN ACTION AND PERCEPTION

To enhance this capacity, humans have developed tools that extend their natural abilities. The first major leap was the creation of implements such as stone tools and weapons, which allowed us to manipulate the environment in ways that bare hands could not achieve. These tools enabled more complex actions, leading to advancements in agriculture with the development of farming equipment, construction with architectural innovations, and craftsmanship with specialized instruments for various trades.

The next significant progression involved expanding our ability to gather and process information. The invention of devices such as cameras, sensors, and satellites, along with communication systems like the telegraph, telephone, and the internet, revolutionized how we observe and understand the world. These technologies increased the volume and variety of data available, necessitating more sophisticated methods for processing information.

As the scale of both action and observation spaces grew, so did the need for tools that could assist in processing vast amounts of data and making intricate decisions. Early efforts focused on prediction tools that simplified decision-making processes for human operators. For example, statistical models were used in weather forecasting to predict climatic conditions, aiding farmers and policymakers. However, the exponential growth of computational power, epitomized by Moore's Law [Moore, 1998] in the mid-20th century, paved the way for developing systems capable of making decisions automatically based on potentially real-time information collected from the environment.

6.1.2 EXPERT SYSTEMS

Early advancements in automated decision-making were marked by the development of rule-based, logic, and expert systems in the 1970s and 1980s. A notable example is the MYCIN system [van Melle, 1978], designed to diagnose bacterial infections and recommend antibiotics by following explicit rules defined by medical experts. These systems processed information and made decisions by applying predefined rules, relying heavily on

expert human knowledge. Compared to individual human experts, these systems had the potential to process larger volumes of information and make decisions more consistently and efficiently. While effective in specific domains, these systems were limited by their dependence on exhaustive human expertise and struggled with scalability and adaptability in more complex or dynamic environments.

6.1.3 OPTIMIZATION-BASED SYSTEMS

To overcome these limitations and reduce the reliance of decision-making systems on human expertise, researchers began to explore optimization-based approaches such as dynamic programming [Bellman, 1957a, Puterman, 1994], search/planning [Browne et al., 2012, García et al., 1989, Hart et al., 1968, Russell and Norvig, 2016], and reinforcement learning [Sutton and Barto, 2018]. These methods focused on optimizing a well-defined objective—often represented as a reward function—within a model of the environment. By employing exact computations or leveraging data-driven approaches, machines could compute policies determining the optimal actions to maximize this objective. Notably, this approach reduced the need for human experts to specify explicit actions, allowing machines to discover optimal strategies independently and, in some cases, outperform human decision-makers.

However, the success of these methods relies on three critical components: (1) a well-defined model of the task or a reliable source of data, (2) efficient optimization algorithms, and (3) substantial computational resources to carry out the optimization.

6.1.4 HARDWARE

In recent decades, significant progress has been made on all three fronts. The development of faster and specialized hardware, such as Graphics Processing Units (GPUs) and Tensor Processing Units (TPUs), has significantly improved the scalability of existing methods and enabled the emergence of large-scale learning algorithms based on neural networks. While neural networks have existed since the 1950s [McCulloch and Pitts, 1943], it was the advent of GPUs that made training deep neural networks feasible due to their ability to perform massive parallel computations efficiently [Krizhevsky et al., 2012]. This capability made it possible to train deep models that were previously impractical because of computational limitations, which paved the way for decision-making systems capable of processing data and deciding actions in high-dimensional spaces.

6.1.5 LEARNING ALGORITHMS

While deep learning [Lecun et al., 2015] mostly focuses on prediction, the combination of deep learning with reinforcement learning has resulted in deep reinforcement learning [François-Lavet et al., 2018, Mnih et al., 2015], a powerful paradigm for data-driven decision-making in high-dimensional state and action spaces that previous methods could not handle without imposing restrictive assumptions. This fusion has led to remarkable successes in various domains. For instance, AlphaZero mastered the games of Go, chess, and shogi from scratch using self-play reinforcement learning, surpassing the performance of both human

experts and specialized programs [Silver et al., 2016, 2017b, 2018]. Similarly, OpenAI Five achieved professional-level performance in the complex multiplayer game Dota 2, defeating top human teams by learning strategies through extensive self-play [OpenAI et al., 2019b].

6.1.6 HARDWARE-ALGORITHM COEVOLUTION

Despite these achievements, a significant challenge of deep reinforcement learning has been the need for massive data to learn effective policies. While algorithmic improvements (e.g., off-policy learning, model-based learning, exploration strategies) have addressed this to some extent, recent advancements in GPU-based simulations [Bradbury et al., 2018, Freeman et al., 2021, Makovychuk et al., 2021] have significantly accelerated data sampling and overall efficiency of simulated RL, where policies are learned in simulators and then lifted to testing environments. This advancement has inspired research towards more data-driven approaches, enabling improved learning efficiency by utilizing the hardware capabilities better [Gallici et al., 2024, Li et al., 2023].

Interestingly, the development of more capable decision-making algorithms has also spurred advancements in hardware. For example, deep reinforcement learning algorithms have been employed to optimize data center cooling systems, leading to significant energy savings. Additionally, reinforcement learning has been applied to chip design optimization [Mirhoseini et al., 2021], including the development of TPUs themselves. Researchers used reinforcement learning to automate the placement of components on a chip, achieving results comparable to or better than those of human experts in a fraction of the time.

These developments highlight a synergistic relationship between hardware and algorithms, where advancements in one domain drive progress in the other. As automated decision-making methods become more sophisticated, they demand greater computational power, which motivates the development of specialized hardware. Conversely, new hardware capabilities enable the exploration of new algorithms, pushing the boundaries of what is possible in automated decision-making. Together, these advancements have significantly enhanced the decision-making capabilities of machines, enabling them to tackle increasingly complex tasks with greater efficiency and effectiveness.

6.1.7 THE DEPENDENCE ON RELIABLE AND CHEAP DATA

Despite the remarkable successes of automated decision-making systems, their successes largely occur in settings where *reliable data is cheap and abundant*—typically in the digital world with access to fast, accurate simulators. In well-defined domains like board games, systems such as AlphaZero can simulate millions of games against themselves without incurring real-world costs. Even complex video games can often be simulated faster than real time, with computation being the main constraint [Mnih et al., 2015, Wurman et al., 2022]. Similar success has been seen in tasks like discovering faster sorting [Fawzi et al., 2022] and matrix multiplication [Mankowitz et al., 2023] algorithms.

Another key factor in these successes is the alignment between training and testing environments. In such domains, data is often sourced directly from the testing environment, reducing the challenges associated with mismatches between training and deployment.

6.1.8 CHALLENGES

However, even within the digital world, not all tasks allow for cheap and reliable data collection. For example, in e-commerce platforms aiming to maximize profit through personalized recommendations or pricing strategies, there are costs associated with each data point. Implementing reinforcement learning directly in the live system can lead to suboptimal user experiences or revenue loss during the learning phase.

In contrast to the digital world, obtaining reliable data for real-world tasks is often significantly more costly due to several factors. First, the real world operates much slower than the digital world, and exploration typically involves risks, safety concerns, and additional expenses. For example, robotics tasks that can be learned in simulated environments like Mujoco [Todorov et al., 2012] within hours may require weeks or months in the real world, along with the costs and risks of damaged robots. While small-scale, controlled tasks—such as robotic manipulation in laboratory settings—can sometimes be learned directly in the testing environment, these exceptions highlight the broader challenges that make direct learning in real-world settings impractical and unscalable for most applications.

6.1.9 USING A MODEL/SIMULATOR

A more practical approach for real-world tasks is to build a model of the task and then use it for planning or learning to construct a decision-making system before deployment in the real world. This paradigm has been successful in various instances [Bellemare et al., 2020, Degraeve et al., 2022, OpenAI et al., 2019a, Richard Evans and Jim Gao, 2024, Tan et al., 2018, Tobin et al., 2017].

However, this approach also presents challenges, particularly in constructing accurate simulators that can run efficiently. Real-world tasks are inherently complex and often involve numerous interacting subsystems, making it difficult to build precise models, especially when the underlying dynamics are not fully understood or are subject to change. Even when a model can be built, it may be computationally intensive to simulate, hindering the speed of learning and planning.

6.1.10 THE CHALLENGE OF MODEL ACCURACY

Most models or simulators are either hand-crafted by domain experts or learned from data—both of which can introduce inaccuracies. While model fidelity is often domain-specific, many decision-making algorithms are designed to account for model errors and remain effective despite them.

A common technique is to introduce pessimism during policy learning: prioritizing decisions in regions where the model is more certain, thereby reducing overfitting to inaccurate predictions. This idea is central in offline reinforcement learning [Levine et al., 2020], where policies are trained on fixed datasets. Pessimism has proven crucial for good performance in this setting [Buckman et al., 2020, Jin et al., 2021, Kidambi et al., 2020, Kumar et al., 2020, Yu et al., 2020].

Alternatively, robust optimization [Ben-Tal et al., 2009, Bertsimas et al., 2011] and robust reinforcement learning [Nilim and Ghaoui, 2005, Pinto et al., 2017, Wiesemann et al.,

2013] address uncertainty by optimizing worst-case performance over a set of possible models. While these methods can be overly conservative, they avoid relying on precise uncertainty estimates.

A related strategy is domain randomization [Peng et al., 2018, Tobin et al., 2017], where agents are trained on diverse simulated environments with randomized parameters to encourage generalization to the real world despite simulator imperfections.

6.1.11 THE CHALLENGE OF MODEL COMPLEXITY

Beyond model fidelity, computational efficiency is a major concern in automated decision-making systems. For example, in chip design [Mirhoseini et al., 2021], much of the training time is spent evaluating designs with slow, industry-standard simulators. Similarly, neural architecture search requires training thousands of models, making data collection costly and time-intensive [Pham et al., 2018, Zoph and Le, 2016].

Improving the sample efficiency of algorithms helps, but a more fundamental solution is to accelerate simulations themselves. One approach is to leverage GPUs or specialized hardware for parallelized simulation [Bradbury et al., 2018, Freeman et al., 2021]. However, this often demands low-level access to simulators, which is infeasible with black-box systems. Even with GPU acceleration, complex simulations involving many interacting components can still be prohibitively slow.

In such cases, surrogate models offer an effective alternative. Widely used in engineering [Alizadeh et al., 2020, Kudela and Matousek, 2022, Razavi et al., 2012, Viana et al., 2021], these models approximate expensive simulators with reduced complexity—often using neural networks. When guided by domain knowledge, surrogate models can focus only on task-relevant aspects, improving efficiency without sacrificing accuracy.

This selective modeling is often overlooked but crucial: while everything in a complex system may interact, not all details are necessary for effective decision-making. By identifying the components that matter most, we can construct models that are both tractable and effective.

Abstraction is key to this process. It provides a principled way to construct minimal models that retain essential task dynamics while ignoring irrelevant details. Our work (Chapters 3–4) builds on this idea, using influence-based abstraction to combine exact local models with learned approximations of external influences. This results in fast, reliable surrogate models particularly well-suited for structured, networked environments.

6.1.12 SUMMARY

Automated decision-making systems have evolved from relying heavily on human expertise to systems capable of discovering strategies that outperform human decision-makers. This shift has been propelled by advancements in hardware and algorithms, which have a symbiotic relationship wherein each drives progress in the other.

However, recent successes in automated decision-making have predominantly occurred in domains where reliable data can be obtained at low cost, such as in the digital realm. A key question for the future is how to extend these successes to economically and practically challenging real-world tasks, where we may not fully understand the system and may not be able to model it accurately and efficiently.

6.1.13 FUTURE

In the near term, human knowledge remains crucial in deciding what aspects of a system to model and what to ignore and in building partial or complete models of the task. However, to scale up decision-making systems and enhance their applicability in the long run, we need to rely more on data-driven approaches that can learn directly from data without the necessity of human-defined models.

This shift is particularly important for tasks that are not well understood and for further removing humans from the loop, leading to more powerful decision-making systems capable of tackling tasks previously inaccessible due to complexity or lack of understanding. This progression aligns with the historical trajectory from expert systems—heavily reliant on human input—to more autonomous systems grounded in planning and learning, which inherently aim to minimize human involvement in the decision-making process.

This raises the question: How do we build reliable models directly from data? We believe this requires methods that can learn from large volumes of data, potentially encompassing multiple modalities. Recent advancements in large language models (LLMs) [OpenAI, 2024, Vaswani et al., 2017] and multimodal models have provided a glimpse of what a general-purpose model might look like. Models such as GPT-4 have demonstrated the ability to absorb vast amounts of information and perform a wide range of tasks, suggesting a pathway toward more generalized decision-making systems.

While a general-purpose model may enable planning policies for a wide range of tasks, one practical challenge is the high inference cost associated with these models. A promising research direction is to develop sparse or factored models that can focus computation on only the relevant parts of the environment for a specific task—for instance, making a cup of coffee. This approach avoids unnecessary feedforward passes through the entire neural network, thereby reducing computational overhead. Inspiration for such an approach may be drawn from the human brain.

Many topics are not covered in this "big picture". For example, the alignment problem, which refers to the challenge of ensuring that the learned policy behaves as intended when deployed in the real world. Other topics include fairness, interpretability, and the societal impact of automated decision-making systems. While we have mainly focused on the technical aspects of decision-making, these broader issues are equally important and can be the bottleneck for the deployment of decision-making systems in practice.

6.2 CONTRIBUTIONS AND INSIGHTS

Chapter 3 introduces an approach to constructing efficient surrogate models for factored POMDPs with many interacting subsystems. We propose the influence-augmented local simulator (IALS), a hybrid model that combines an exact local simulator with a learned influence model. The local simulator captures the agent’s immediate dynamics relevant to rewards and observations, while the influence model—parameterized by neural networks—approximates the effect of external subsystems. This hybrid design enables accurate simulation of critical task components while abstracting away less important details.

What we learned is that approximation must be applied selectively. Not all approximations are equally tolerable—some matter more for decision quality. When constrained by data, representation, or compute, the trade-off between precision and efficiency should be made deliberately. This can be guided either by domain knowledge—such as task structure, as in influence-based abstraction—or by domain-agnostic principles, such as value equivalence, which emphasizes preserving task-relevant behavior without requiring structural assumptions.

Chapter 4 builds on this with a general paradigm for learning and using abstract simulators during planning. Instead of relying on a fixed offline-trained surrogate, we propose learning the model online and dynamically choosing between the abstract and full simulator based on estimated accuracy. Though applied here to IALS, the paradigm is broadly applicable to cases where planning must be done under computational constraints and imperfect simulators.

This introduces a meta-reasoning challenge: deciding when to trust the approximate simulator. Since solving this problem optimally is intractable—and often more expensive than solving the original decision problem itself—we rely on heuristics: estimating divergence between simulators and using a bandit algorithm to arbitrate. While imperfect, this approach performs well in practice. It mirrors how humans switch between fast and slow reasoning systems: the meta-decision is heuristic, quick, and good enough—not necessarily optimal.

Chapter 5 analyzes the MuZero algorithm, a model-based reinforcement learning method that has achieved strong empirical performance. We investigate whether MuZero’s success stems from its learned model and how this model contributes to planning. Our results show that MuZero’s model often lacks the accuracy needed for general-purpose planning, especially under distribution shift. Yet planning with MuZero remains effective, primarily due to the use of a learned policy prior that focuses the search on regions where the model is more reliable.

What we learned is that in end-to-end systems, it is difficult—and often misleading—to assume the function of individual components. Although MuZero’s model appears central, its planning effectiveness depends largely on how it is constrained by the policy prior. This shows that the model’s value lies in its interaction with other components, not its standalone fidelity. Simplified views—like treating MuZero as AlphaZero plus a learned model—miss this nuance. Understanding model-based methods may require reframing them through the lenses of representation learning, exploration, and architectural inductive biases.

6.3 LIMITATIONS AND FUTURE WORK

Dependence on a factored simulator A key limitation of our work on influence-augmented local simulators is the assumption of access to a factored simulator. This enables us to 1) identify the local state variables, 2) construct the local simulator, and 3) collect data to train the influence model.

In practice, constructing such a simulator can be difficult, as it requires prior knowledge of the task’s structure and dynamics. Below, we examine to what extent this assumption might be relaxed.

Constructing the local simulator and training the influence model can, in some cases, be done from interaction data collected in the real environment. If the agent only interacts with a small and well-understood part of the environment, the local simulator may also be constructed manually by a domain expert.

In contrast, identifying the local state variables—the core of the factorization—is substantially harder. Without this step, it is not even possible to define what the local and influence models should capture. In many structured systems, this decomposition is available or can be provided by experts. Where this is not the case, the structure must be learned from data—posing challenges in causal discovery or unsupervised factorization.

In summary, while our reliance on a fully factored simulator can be partially relaxed, some knowledge of the task structure remains essential—either provided explicitly or learned through structure discovery methods.

Generalizing our approach of self-improving simulators Chapter 4 introduces a general paradigm for learning and using abstract simulators during planning. While our work applies this to influence-based abstraction, the underlying idea—adapting simulator usage based on online accuracy estimates—is broadly applicable.

Generalizing this paradigm to other forms of abstraction is a promising direction. The key promise is to preserve decision quality throughout learning by dynamically deciding when to trust the abstract simulator. A central challenge here is estimating simulator accuracy in context, using as few samples as possible from the expensive base simulator.

What model to learn in MBRL In model-based reinforcement learning, the choice of training loss determines the level and kind of abstraction the model captures [Li et al., 2006]. Value-equivalent models [Schrittwieser et al., 2020] have shown strong performance in high-data regimes, while reconstruction- and temporal-consistency-based models [Ye et al., 2021] tend to perform better in low-data settings.

A deeper understanding—both theoretical and empirical—of how different losses influence abstraction quality would be valuable. In particular, it remains an open question how these trade-offs depend on the complexity of the task, the model capacity, the available data, and the intended model usage (e.g., planning vs. policy learning). Clarifying these relationships could help guide the design of learning objectives in future model-based reinforcement learning methods.

BIBLIOGRAPHY

- David Abel, David Hershkowitz, and Michael Littman. Near Optimal Behavior via Approximate State Abstraction. In *Proceedings of The 33rd International Conference on Machine Learning*, pages 2915–2923. PMLR, 2016.
- Joshua Achiam, Harrison Edwards, Dario Amodei, and Pieter Abbeel. Variational Option Discovery Algorithms. *arXiv preprint arXiv:1807.10299*, 2018.
- Alekh Agarwal, Nan Jiang, Sham M Kakade, and Wen Sun. Reinforcement learning: Theory and algorithms. *CS Dept., UW Seattle, Seattle, WA, USA, Tech. Rep*, 32:96, 2019.
- Reza Alizadeh, Janet K. Allen, and Farrokh Mistree. Managing computational complexity using surrogate models: A critical review. *Research in Engineering Design*, 31(3):275–298, 2020.
- Ankit Anand, Aditya Grover, Mausam, and Parag Singla. A Novel Abstraction Framework for Online Planning: Extended Abstract. In *Proceedings of the 2015 International Conference on Autonomous Agents and Multiagent Systems, AAMAS '15*, pages 1901–1902, Richland, SC, 2015a. International Foundation for Autonomous Agents and Multiagent Systems.
- Ankit Anand, Aditya Grover, Mausam Mausam, and Parag Singla. ASAP-UCT: Abstraction of state-action pairs in UCT. In *Proceedings of the 24th International Conference on Artificial Intelligence, IJCAI'15*, pages 1509–1515, Buenos Aires, Argentina, 2015b. AAAI Press.
- Ankit Anand, Ritesh Noothigattu, Mausam, and Parag Singla. OGA-UCT: On-the-Go Abstractions in UCT. In *Proceedings of the International Conference on Automated Planning and Scheduling*, volume 26, pages 29–37, 2016.
- Thomas Anthony, Zheng Tian, and David Barber. Thinking Fast and Slow with Deep Learning and Tree Search. In *Advances in Neural Information Processing Systems*, pages 5360–5370, 2017.
- Ioannis Antonoglou, Julian Schrittwieser, Sherjil Ozair, Thomas K. Hubert, and David Silver. Planning in Stochastic Environments with a Learned Model. In *International Conference on Learning Representations*, 2021.
- Charles Audet, J. Denni, Douglas Moore, Andrew Booker, and Paul Frank. A surrogate-model-based method for constrained optimization. In *8th Symposium on Multidisciplinary Analysis and Optimization*. American Institute of Aeronautics and Astronautics, 2000.
- Peter Auer, Nicolò Cesa-Bianchi, and Paul Fischer. Finite-time analysis of the multiarmed bandit problem. *Machine Learning*, 47(2-3):235–256, 2002.

- Pierre-Luc Bacon, Jean Harb, and Doina Precup. The Option-Critic Architecture. In *Thirty-First AAAI Conference on Artificial Intelligence*, pages 1726–1734, 2017.
- Akhil Bagaria and George Konidaris. Option Discovery using Deep Skill Chaining. In *International Conference on Learning Representations*, 2019.
- Aijun Bai, Siddharth Srivastava, and Stuart Russell. Markovian state and action abstractions for MDPS via hierarchical MCTS. In *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence, IJCAI’16*, pages 3029–3037, New York, New York, USA, 2016. AAAI Press.
- Raphen Becker, Shlomo Zilberstein, Victor Lesser, and Claudia V Goldman. Transition-Independent Decentralized Markov Decision Processes. In *Proceedings of the International Conference on Autonomous Agents*, volume 2, pages 41–48, 2003.
- Raphen Becker, Shlomo Zilberstein, and Victor Lesser. Decentralized Markov decision processes with event-driven interactions. In *Proceedings of the Third International Joint Conference on Autonomous Agents and Multiagent Systems, AAMAS 2004*, volume 1, pages 302–309, 2004.
- Marc G Bellemare, Joel Veness, and Michael Bowling. The Arcade Learning Environment: An Evaluation Platform for General Agents. *Journal of Artificial Intelligence Research*, 47: 253–279, 2013.
- Marc G. Bellemare, Salvatore Candido, Pablo Samuel Castro, Jun Gong, Marlos C. Machado, Subhdeep Moitra, Sameera S. Ponda, and Ziyu Wang. Autonomous navigation of stratospheric balloons using reinforcement learning. *Nature*, 588(7836):77–82, 2020.
- Richard Bellman. *Dynamic Programming*. Princeton University Press, 1957a.
- Richard Bellman. A Markovian Decision Process. *Journal of Mathematics and Mechanics*, 6 (5):679–684, 1957b.
- Aharon Ben-Tal, Laurent El Ghaoui, and Arkadi Nemirovski. *Robust Optimization*. Princeton University Press, 2009.
- Dimitri Bertsekas. *Lessons from AlphaZero for Optimal, Model Predictive, and Adaptive Control*. Athena Scientific, 2022.
- Dimitris Bertsimas, David B. Brown, and Constantine Caramanis. Theory and applications of robust optimization. *SIAM Review*, 53(3):464–501, 2011.
- Craig Boutilier, Thomas Dean, and Steve Hanks. Decision-Theoretic Planning: Structural Assumptions and Computational Leverage. *Journal of Artificial Intelligence Research*, 11: 1–94, 1999.
- James Bradbury, Roy Frostig, Peter Hawkins, Matthew James Johnson, Chris Leary, Dougal Maclaurin, George Necula, Adam Paszke, Jake VanderPlas, Skye Wanderman-Milne, and Qiao Zhang. JAX: Composable transformations of Python+NumPy programs, 2018.

- Ronen I. Brafman and Moshe Tennenholtz. R-MAX - A General Polynomial Time Algorithm for Near-Optimal Reinforcement Learning. *Journal of Machine Learning Research*, 3(Oct): 213–231, 2002.
- Cameron B. Browne, Edward Powley, Daniel Whitehouse, Simon M. Lucas, Peter I Cowling, Philipp Rohlfshagen, Stephen Tavener, Diego Perez, Spyridon Samothrakis, and Simon Colton. A survey of Monte Carlo tree search methods. *IEEE Transactions on Computational Intelligence and AI in games*, 4(1):1–43, 2012.
- Jacob Buckman, Carles Gelada, and Marc G. Bellemare. The Importance of Pessimism in Fixed-Dataset Policy Optimization. In *International Conference on Learning Representations*, 2020.
- Lars Buesing, Theophane Weber, Sebastien Racaniere, S. M. Ali Eslami, Danilo Rezende, David P. Reichert, Fabio Viola, Frederic Besse, Karol Gregor, Demis Hassabis, and Daan Wierstra. Learning and Querying Fast Generative Models for Reinforcement Learning. *arXiv preprint arXiv:1802.03006*, 2018.
- Qifeng Chen, Caisheng Wei, Yongfeng Shi, and Yunhe Meng. Satellite Swarm Reconfiguration Planning Based on Surrogate Models. *Journal of Guidance, Control, and Dynamics*, 43(9):1750–1756, 2020.
- Yangkun Chen, Joseph Suarez, Junjie Zhang, Chenghui Yu, Bo Wu, Hanmo Chen, Hengman Zhu, Rui Du, Shanliang Qian, Shuai Liu, Weijun Hong, Jinke He, Yibing Zhang, Liang Zhao, Clare Zhu, Julian Togelius, Sharada Mohanty, Jiaxin Chen, Xiu Li, Xiaolong Zhu, and Phillip Isola. Benchmarking Robustness and Generalization in Multi-Agent Systems: A Case Study on Neural MMO. *arXiv preprint arXiv:2308.15802*, 2023.
- Rohan Chitnis and Tomás Lozano-Pérez. Learning Compact Models for Planning with Exogenous Processes. In *Proceedings of the Conference on Robot Learning*, pages 813–822. PMLR, 2020.
- Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using RNN encoder-decoder for statistical machine translation. In *Conference on Empirical Methods in Natural Language Processing (EMNLP 2014)*, pages 1724–1734, 2014.
- Elena Congeduti, Alexander Mey, and Frans A. Oliehoek. Loss Bounds for Approximate Influence-Based Abstraction. In *Proceedings of the 20th International Conference on Autonomous Agents and MultiAgent Systems*, AAMAS ’21, pages 377–385, Richland, SC, 2021. International Foundation for Autonomous Agents and Multiagent Systems.
- Rémi Coulom. Efficient selectivity and backup operators in Monte-Carlo tree search. In *International Conference on Computers and Games*, volume 4630 LNCS, pages 72–83, 2006.
- Ivo Danihelka, Arthur Guez, Julian Schrittwieser, and David Silver. Policy improvement by planning with Gumbel. In *International Conference on Learning Representations*, 2021.
- Joery de Vries, Ken Voskuil, Thomas M. Moerland, and Aske Plaat. Visualizing MuZero Models. In *ICML 2021 Workshop on Unsupervised Reinforcement Learning*, 2021.

- Jonas Degraeve, Federico Felici, Jonas Buchli, Michael Neunert, Brendan Tracey, Francesco Carpanese, Timo Ewalds, Roland Hafner, Abbas Abdolmaleki, Diego de las Casas, Craig Donner, Leslie Fritz, Cristian Galperti, Andrea Huber, James Keeling, Maria Tsimpoukelli, Jackie Kay, Antoine Merle, Jean-Marc Moret, Seb Noury, Federico Pesamosca, David Pfau, Olivier Sauter, Cristian Sommariva, Stefano Coda, Basil Duval, Ambrogio Fasoli, Pushmeet Kohli, Koray Kavukcuoglu, Demis Hassabis, and Martin Riedmiller. Magnetic control of tokamak plasmas through deep reinforcement learning. *Nature*, 602(7897): 414–419, 2022.
- Marc Peter Deisenroth and Carl Edward Rasmussen. PILCO: A model-based and data-efficient approach to policy search. In *Proceedings of the 28th International Conference on Machine Learning, ICML 2011*, pages 465–472, 2011.
- E. W. Dijkstra. A note on two problems in connexion with graphs. *Numerische Mathematik*, 1(1):269–271, 1959.
- Finale Doshi-velez, David Wingate, Nicholas Roy, and Joshua Tenenbaum. Nonparametric Bayesian Policy Priors for Reinforcement Learning. In *Advances in Neural Information Processing Systems*, volume 23. Curran Associates, Inc., 2010.
- Arnaud Doucet and Adam M. Johansen. A tutorial on particle filtering and smoothing: Fifteen years later. *Handbook of nonlinear filtering*, 12(656-704):3, 2009.
- Michael O’Gordon Duff. *Optimal Learning: Computational Procedures for Bayes-Adaptive Markov Decision Processes*. PhD thesis, University of Massachusetts Amherst, 2002.
- Werner Duvaud and Aurèle Hainaut. MuZero general: Open reimplement of MuZero. <https://github.com/werner-duvaud/muzero-general>, 2019.
- Jonathan St B. T. Evans. Dual-Processing Accounts of Reasoning, Judgment, and Social Cognition. *Annual Review of Psychology*, 59(Volume 59, 2008):255–278, 2008.
- Benjamin Eysenbach, Abhishek Gupta, Julian Ibarz, and Sergey Levine. Diversity is All You Need: Learning Skills without a Reward Function. In *International Conference on Learning Representations*, 2018.
- Gregory Farquhar, Tim Rocktäschel, Maximilian Igl, and Shimon Whiteson. TreeQN and ATreeC: Differentiable Tree-Structured Models for Deep Reinforcement Learning. In *International Conference on Learning Representations*, 2018.
- Alhussein Fawzi, Matej Balog, Aja Huang, Thomas Hubert, Bernardino Romera-Paredes, Mohammadamin Barekatain, Alexander Novikov, Francisco J. R. Ruiz, Julian Schrittwieser, Grzegorz Swirszcz, David Silver, Demis Hassabis, and Pushmeet Kohli. Discovering faster matrix multiplication algorithms with reinforcement learning. *Nature*, 610(7930):47–53, 2022.
- Daniele Foffano, Jinke He, and Frans A Oliehoek. Robust Ensemble Adversarial Model-Based Reinforcement Learning. In *Adaptive and Learning Agents Workshop at AAMAS*, 2022.

- Roy Fox, Sanjay Krishnan, Ion Stoica, and Ken Goldberg. Multi-Level Discovery of Deep Options. *arXiv preprint arXiv:1703.08294*, 2017.
- Vincent François-Lavet, Peter Henderson, Riashat Islam, Marc G. Bellemare, and Joelle Pineau. An Introduction to Deep Reinforcement Learning. *Foundations and Trends® in Machine Learning*, 11(3-4):219–354, 2018.
- C. Daniel Freeman, Erik Frey, Anton Raichuk, Sertan Girgin, Igor Mordatch, and Olivier Bachem. Brax - A Differentiable Physics Engine for Large Scale Rigid Body Simulation. In *Advances in Neural Information Processing Systems*, 2021.
- Matteo Gallici, Mattie Fellows, Benjamin Ellis, Bartomeu Pou, Ivan Masmitja, Jakob Nicolaus Foerster, and Mario Martin. Simplifying Deep Temporal Difference Learning. *arXiv preprint arXiv:2407.04811*, 2024.
- Carlos E. García, David M. Prett, and Manfred Morari. Model predictive control: Theory and practice—A survey. *Automatica*, 25(3):335–348, 1989.
- Matthieu Geist, Bruno Scherrer, and Olivier Pietquin. A Theory of Regularized Markov Decision Processes. *arXiv preprint arXiv:1901.11275*, 2019.
- Carles Gelada, Saurabh Kumar, Jacob Buckman, Ofir Nachum, and Marc G. Bellemare. DeepMDP: Learning Continuous Latent Space Models for Representation Learning. In *Proceedings of the 36th International Conference on Machine Learning*, pages 2170–2179. PMLR, 2019.
- Sylvain Gelly and David Silver. Combining online and offline knowledge in UCT. In *ACM International Conference Proceeding Series*, volume 227, pages 273–280, New York, New York, USA, 2007. ACM Press.
- Sylvain Gelly and David Silver. Monte-Carlo tree search and rapid action value estimation in computer Go. *Artificial Intelligence*, 175:1856–1875, 2011.
- Samuel J. Gershman. Reinforcement learning and causal models. In *The Oxford Handbook of Causal Reasoning*, Oxford Library of Psychology, pages 295–306. Oxford University Press, New York, NY, US, 2017.
- Mohammad Ghavamzadeh, Shie Mannor, Joelle Pineau, and Aviv Tamar. *Bayesian Reinforcement Learning: A Survey*, volume 8. 2015.
- Jean-Bastien Grill, Florent Althé, Yunhao Tang, Thomas Hubert, Michal Valko, Ioannis Antonoglou, and Remi Munos. Monte-Carlo Tree Search as Regularized Policy Optimization. In *Proceedings of the 37th International Conference on Machine Learning*, pages 3769–3778. PMLR, 2020.
- Christopher Grimm, Andre Barreto, Satinder Singh, and David Silver. The Value Equivalence Principle for Model-Based Reinforcement Learning. In *Advances in Neural Information Processing Systems*, volume 33, pages 5541–5552. Curran Associates, Inc., 2020.

- Christopher Grimm, André Barreto, Gregory Farquhar, David Silver, and Satinder Singh. Proper value equivalence. In *Advances in Neural Information Processing Systems*, volume 34, pages 7773–7786, 2021.
- Christopher Grimm, Andre Barreto, and Satinder Singh. Approximate Value Equivalence. In *Advances in Neural Information Processing Systems*, volume 35, 2022.
- Radek Grzeszczuk, Demetri Terzopoulos, and Geoffrey Hinton. Fast neural network emulation of dynamical systems for computer animation. In *Advances in Neural Information Processing Systems*, pages 882–888, 1999.
- Arthur Guez, David Silver, and Peter Dayan. Efficient Bayes-adaptive reinforcement learning using sample-based search. In *Advances in Neural Information Processing Systems*, volume 2, pages 1025–1033, 2012.
- Arthur Guez, Theophane Weber, Ioannis Antonoglou, Karen Simonyan, Oriol Vinyals, Daan Wierstra, Remi Munos, and David Silver. Learning to search with MCTSnets. In *Proceedings of the 35th International Conference on Machine Learning*, pages 1822–1831. PMLR, 2018.
- Arthur Guez, Mehdi Mirza, Karol Gregor, Rishabh Kabra, Sebastien Racaniere, Theophane Weber, David Raposo, Adam Santoro, Laurent Orseau, Tom Eccles, Greg Wayne, David Silver, and Timothy Lillicrap. An Investigation of Model-Free Planning. In *Proceedings of the 36th International Conference on Machine Learning*, pages 2464–2473. PMLR, 2019.
- David Ha and Jürgen Schmidhuber. Recurrent World Models Facilitate Policy Evolution. In *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018.
- Danijar Hafner, Timothy Lillicrap, Jimmy Ba, and Mohammad Norouzi. Dream to Control: Learning Behaviors by Latent Imagination. In *International Conference on Learning Representations*, 2019.
- Danijar Hafner, Timothy P. Lillicrap, Mohammad Norouzi, and Jimmy Ba. Mastering Atari with Discrete World Models. In *International Conference on Learning Representations*, 2020.
- Danijar Hafner, Jurgis Pasukonis, Jimmy Ba, and Timothy Lillicrap. Mastering Diverse Domains through World Models. *arXiv preprint arXiv:2301.04104*, 2023.
- Jessica B. Hamrick, Abram L. Friesen, Feryal Behbahani, Arthur Guez, Fabio Viola, Sims Witherspoon, Thomas Anthony, Lars Holger Buesing, Petar Veličković, and Theophane Weber. On the role of planning in model-based deep reinforcement learning. In *International Conference on Learning Representations*, 2022.
- E. Hansen and Z. Feng. Dynamic Programming for POMDPs Using a Factored State Representation. In *International Conference on Artificial Intelligence Planning Systems*, 2000.

- Eric A Hansen, Daniel S Bernstein, and Shlomo Zilberstein. Dynamic programming for partially observable stochastic games. In *AAAI*, volume 4, pages 709–715, 2004.
- Jean Harb, Pierre Luc Bacon, Martin Klissarov, and Doina Precup. When waiting is not an option: Learning options with a deliberation cost. In *32nd AAAI Conference on Artificial Intelligence, AAAI 2018*, pages 3165–3172, 2018.
- Peter E. Hart, Nils J. Nilsson, and Bertram Raphael. A Formal Basis for the Heuristic Determination of Minimum Cost Paths. *IEEE Transactions on Systems Science and Cybernetics*, 4(2):100–107, 1968.
- Jinke He, Miguel Suau de Castro, and Frans Oliehoek. Influence-Augmented Online Planning for Complex Environments. In *Advances in Neural Information Processing Systems*, volume 33, pages 4392–4402. Curran Associates, Inc., 2020.
- Jinke He, Miguel Suau, Hendrik Baier, Michael Kaisers, and Frans A. Oliehoek. Online Planning in POMDPs with Self-Improving Simulators. In *Proceedings of the Thirty-First International Joint Conference on Artificial Intelligence*, pages 4628–4634, Vienna, Austria, 2022. International Joint Conferences on Artificial Intelligence Organization.
- Nicolas Heess, Greg Wayne, Yuval Tassa, Timothy Lillicrap, Martin Riedmiller, and David Silver. Learning and Transfer of Modulated Locomotor Controllers, 2016.
- Mikael Henaff. Explicit Explore-Exploit Algorithms in Continuous State Spaces. In *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019.
- Peter Henderson, Riashat Islam, Philip Bachman, Joelle Pineau, Doina Precup, and David Meger. Deep Reinforcement Learning That Matters. *Proceedings of the AAAI Conference on Artificial Intelligence*, 32(1), 2018.
- Matteo Hessel, Ivo Danihelka, Fabio Viola, Arthur Guez, Simon Schmitt, Laurent Sifre, Theophane Weber, David Silver, and Hado van Hasselt. Muesli: Combining Improvements in Policy Optimization. *arXiv preprint arXiv:2104.06159*, 2022.
- Sepp Hochreiter and Jürgen Schmidhuber. Long Short-Term Memory. *Neural Computation*, 9(8):1735–1780, 1997.
- Jesse Hostetler, Alan Fern, and Tom Dietterich. State Aggregation in Monte Carlo Tree Search. *Proceedings of the AAAI Conference on Artificial Intelligence*, 28(1), 2014.
- Ronald A. Howard. *Dynamic Programming and Markov Processes*. Dynamic Programming and Markov Processes. John Wiley, Oxford, England, 1960.
- Shengyi Huang and Santiago Ontañón. A Closer Look at Invalid Action Masking in Policy Gradient Algorithms. *The International FLAIRS Conference Proceedings*, 35, 2022.
- Thomas Hubert, Julian Schrittwieser, Ioannis Antonoglou, Mohammadamin Barekatain, Simon Schmitt, and David Silver. Learning and Planning in Complex Action Spaces. In *Proceedings of the 38th International Conference on Machine Learning*, pages 4476–4486. PMLR, 2021.

Maximilian Igl, Andrew Gambardella, J. He, Nantas Nardelli, N. Siddharth, J. W. Böhmer, and Shimon Whiteson. Multitask Soft Option Learning. In *Proceedings of the 36th Conference on Uncertainty in Artificial Intelligence (UAI)*, volume 124, 2020.

Alex Irpan. Deep reinforcement learning doesn't work yet, 2018.

Riashat Islam, Peter Henderson, Maziar Gomrokchi, and Doina Precup. Reproducibility of Benchmarked Deep Reinforcement Learning Tasks for Continuous Control. *arXiv preprint arXiv:1708.04133*, 2017.

Max Jaderberg, Volodymyr Mnih, Wojciech Marian Czarnecki, Tom Schaul, Joel Z. Leibo, David Silver, and Koray Kavukcuoglu. Reinforcement Learning with Unsupervised Auxiliary Tasks. In *International Conference on Learning Representations*, 2022.

Nan Jiang, Satinder Singh, and Richard Lewis. Improving UCT planning via approximate homomorphisms. In *Proceedings of the 2014 International Conference on Autonomous Agents and Multi-Agent Systems, AAMAS '14*, pages 1289–1296, Richland, SC, 2014. International Foundation for Autonomous Agents and Multiagent Systems.

Nan Jiang, Alex Kulesza, Satinder Singh, and Richard Lewis. The Dependence of Effective Planning Horizon on Model Accuracy. In *Proceedings of the 2015 International Conference on Autonomous Agents and Multiagent Systems, AAMAS '15*, pages 1181–1189, Richland, SC, 2015. International Foundation for Autonomous Agents and Multiagent Systems.

Ying Jin, Zhuoran Yang, and Zhaoran Wang. Is Pessimism Provably Efficient for Offline RL? In *Proceedings of the 38th International Conference on Machine Learning*, pages 5084–5096. PMLR, 2021.

Yuu Jinnai, David Abel, David Hershkowitz, Michael Littman, and George Konidaris. Finding Options that Minimize Planning Time. In *Proceedings of the 36th International Conference on Machine Learning*, pages 3120–3129. PMLR, 2019.

Leslie Pack Kaelbling, Michael L. Littman, and Anthony R. Cassandra. Planning and acting in partially observable stochastic domains. *Artificial Intelligence*, 101(1-2):99–134, 1998.

Daniel Kahneman. *Thinking, Fast and Slow*. Macmillan, 2011.

Łukasz Kaiser, Mohammad Babaeizadeh, Piotr Miłoś, Błażej Osinski, Roy H. Campbell, Konrad Czechowski, Dumitru Erhan, Chelsea Finn, Piotr Kozakowski, Sergey Levine, Afroz Mohiuddin, Ryan Sepassi, George Tucker, and Henryk Michalewski. Model Based Reinforcement Learning for Atari. In *International Conference on Learning Representations*, 2020.

Anssi Kanervisto, Christian Scheller, and Ville Hautamäki. Action Space Shaping in Deep Reinforcement Learning. In *2020 IEEE Conference on Games (CoG)*, pages 479–486, 2020.

Sammie Katt, Frans A. Oliehoek, and Christopher Amato. Learning in POMDPs with Monte Carlo Tree Search. In *Proceedings of the 34th International Conference on Machine Learning*, pages 1819–1827. PMLR, 2017.

- Charles W Kazer, João Sedoc, Kelvin K.W. Ng, Vincent Liu, and Lyle H Ungar. Fast network simulation through approximation or: How blind men can describe elephants. In *HotNets 2018 - Proceedings of the 2018 ACM Workshop on Hot Topics in Networks*, pages 141–147, 2018.
- Marc C Kennedy and Anthony O’Hagan. Predicting the output from a complex computer code when fast approximations are available. *Biometrika*, 87(1):1–13, 2000.
- Rahul Kidambi, Aravind Rajeswaran, Praneeth Netrapalli, and Thorsten Joachims. MOREL: Model-Based Offline Reinforcement Learning. In *Advances in Neural Information Processing Systems*, volume 33, pages 21810–21823. Curran Associates, Inc., 2020.
- J. Kiefer. Sequential Minimax Search for a Maximum. *Proceedings of the American Mathematical Society*, 4(3):502–506, 1953.
- Diederik P Kingma and Jimmy Ba. Adam: A Method for Stochastic Optimization. *arXiv preprint arXiv:1412.6980*, 2014.
- Jens Kober, J. Andrew Bagnell, and Jan Peters. Reinforcement learning in robotics: A survey. *The International Journal of Robotics Research*, 32(11):1238–1274, 2013.
- Mykel J. Kochenderfer, Tim A. Wheeler, and Kyle H. Wray. *Algorithms for Decision Making*. MIT press, 2022.
- Levente Kocsis and Csaba Szepesvári. Bandit based Monte-Carlo planning. In *European Conference on Machine Learning*, volume 4212 LNAI, pages 282–293, 2006.
- Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. ImageNet Classification with Deep Convolutional Neural Networks. In *Advances in Neural Information Processing Systems*, volume 25. Curran Associates, Inc., 2012.
- Jakub Kudela and Radomil Matousek. Recent advances and applications of surrogate models for finite element method computations: A review. *Soft Computing*, 26(24):13709–13733, 2022.
- Aviral Kumar, Aurick Zhou, George Tucker, and Sergey Levine. Conservative Q-Learning for Offline Reinforcement Learning. In *Advances in Neural Information Processing Systems*, volume 33, pages 1179–1191. Curran Associates, Inc., 2020.
- Nathan Lambert, Kristofer Pister, and Roberto Calandra. Investigating Compounding Prediction Errors in Learned Dynamics Models. *arXiv preprint arXiv:2203.09637*, 2022.
- Yann Lecun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *Nature*, 521(7553):436–444, 2015.
- Sergey Levine. Reinforcement Learning and Control as Probabilistic Inference: Tutorial and Review. *arXiv preprint arXiv:1805.00909*, 2018.
- Sergey Levine, Aviral Kumar, George Tucker, and Justin Fu. Offline Reinforcement Learning: Tutorial, Review, and Perspectives on Open Problems. *arXiv preprint arXiv:2005.01643*, 2020.

- Lihong Li, Thomas J. Walsh, and Michael L. Littman. Towards a Unified Theory of State Abstraction for MDPs. In *International Symposium on Artificial Intelligence and Mathematics, AI&Math*, 2006.
- Zechu Li, Tao Chen, Zhang-Wei Hong, Anurag Ajay, and Pulkit Agrawal. Parallel \$Q\$-Learning: Scaling Off-policy Reinforcement Learning under Massively Parallel Simulation. In *Proceedings of the 40th International Conference on Machine Learning*, pages 19440–19459. PMLR, 2023.
- Kendall Lowrey, Aravind Rajeswaran, Sham Kakade, Emanuel Todorov, and Igor Mordatch. Plan Online, Learn Offline: Efficient Learning and Exploration via Model-Based Control. In *International Conference on Learning Representations*, 2018.
- Jerry Luo, Cosmin Paduraru, Octavian Voicu, Yuri Chervonyi, Scott Munns, Jerry Li, Crystal Qian, Praneet Dutta, Jared Quincy Davis, Ningjia Wu, Xingwei Yang, Chu-Ming Chang, Ted Li, Rob Rose, Mingyan Fan, Hootan Nakhost, Tinglin Liu, Brian Kirkman, Frank Altamura, Lee Cline, Patrick Tonker, Joel Gouker, Dave Uden, Warren Buddy Bryan, Jason Law, Deeni Fatiha, Neil Satra, Juliet Rothenberg, Mandeep Waraich, Molly Carlin, Satish Tallapaka, Sims Witherspoon, David Parish, Peter Dolan, Chenyu Zhao, and Daniel J. Mankowitz. Controlling Commercial Cooling Systems Using Reinforcement Learning. *arXiv preprint arXiv:2211.07357*, 2022.
- Marlos C. Machado, Marc G. Bellemare, and Michael Bowling. A Laplacian Framework for Option Discovery in Reinforcement Learning. In *Proceedings of the 34th International Conference on Machine Learning*, pages 2295–2304. PMLR, 2017.
- Dhruv Madeka, Kari Torkkola, Carson Eisenach, Anna Luo, Dean P. Foster, and Sham M. Kakade. Deep Inventory Management. *arXiv preprint arXiv:2210.03137*, 2022.
- Viktor Makoviychuk, Lukasz Wawrzyniak, Yunrong Guo, Michelle Lu, Kier Storey, Miles Macklin, David Hoeller, Nikita Rudin, Arthur Allshire, Ankur Handa, and Gavriel State. Isaac gym: High performance GPU-based physics simulation for robot learning, 2021.
- Amol Mandhane, Anton Zhernov, Maribeth Rauh, Chenjie Gu, Miaosen Wang, Flora Xue, Wendy Shang, Derek Pang, Rene Claus, Ching-Han Chiang, Cheng Chen, Jingning Han, Angie Chen, Daniel J. Mankowitz, Jackson Broshear, Julian Schrittwieser, Thomas Hubert, Oriol Vinyals, and Timothy Mann. MuZero with Self-competition for Rate Control in VP9 Video Compression. *arXiv preprint arXiv:2202.06626*, 2022.
- Daniel J. Mankowitz, Andrea Michi, Anton Zhernov, Marco Gelmi, Marco Selvi, Cosmin Paduraru, Edouard Leurent, Shariq Iqbal, Jean-Baptiste Lespiau, Alex Ahern, Thomas Köppe, Kevin Millikin, Stephen Gaffney, Sophie Elster, Jackson Broshear, Chris Gamble, Kieran Milan, Robert Tung, Minjae Hwang, Taylan Cemgil, Mohammadamin Barekatain, Yujia Li, Amol Mandhane, Thomas Hubert, Julian Schrittwieser, Demis Hassabis, Pushmeet Kohli, Martin Riedmiller, Oriol Vinyals, and David Silver. Faster sorting algorithms discovered using deep reinforcement learning. *Nature*, 618(7964):257–263, 2023.
- Shie Mannor and Aviv Tamar. Towards Deployable RL – What’s Broken with RL Research and a Potential Fix. *arXiv preprint arXiv:2301.01320*, 2023.

- Andrew Kachites McCallum. *Reinforcement Learning with Selective Perception and Hidden State*. PhD thesis, University of Rochester, 1996.
- Warren S. McCulloch and Walter Pitts. A logical calculus of the ideas immanent in nervous activity. *The bulletin of mathematical biophysics*, 5(4):115–133, 1943.
- Azalia Mirhoseini, Anna Goldie, Mustafa Yazgan, Joe Wenjie Jiang, Ebrahim Songhori, Shen Wang, Young-Joon Lee, Eric Johnson, Omkar Pathak, Azade Nazi, Jiwoo Pak, Andy Tong, Kavya Srinivasa, William Hang, Emre Tuncer, Quoc V. Le, James Laudon, Richard Ho, Roger Carpenter, and Jeff Dean. A graph placement methodology for fast chip design. *Nature*, 594(7862):207–212, 2021.
- Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A. Rusu, Joel Veness, Marc G. Bellemare, Alex Graves, Martin Riedmiller, Andreas K. Fidjeland, Georg Ostrovski, Stig Petersen, Charles Beattie, Amir Sadik, Ioannis Antonoglou, Helen King, Dharmashan Kumaran, Daan Wierstra, Shane Legg, and Demis Hassabis. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, 2015.
- Thomas M. Moerland, Joost Broekens, Aske Plaat, and Catholijn M. Jonker. Model-based Reinforcement Learning: A Survey. *Foundations and Trends® in Machine Learning*, 16(1): 1–118, 2023.
- Andrew W. Moore and Christopher G. Atkeson. Prioritized sweeping: Reinforcement learning with less data and less time. *Machine Learning*, 13(1):103–130, 1993.
- Gordon E Moore. Cramming More Components onto Integrated Circuits. *PROCEEDINGS OF THE IEEE*, 86(1), 1998.
- Benjamin Moseley, Andrew Markham, and Tarje Nissen-Meyer. Fast approximate simulation of seismic waves with deep learning. *arXiv preprint arXiv:1807.06873*, 2018.
- Ofir Nachum, Shixiang (Shane) Gu, Honglak Lee, and Sergey Levine. Data-Efficient Hierarchical Reinforcement Learning. In *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018.
- Arnab Nilim and Laurent El Ghaoui. Robust Control of Markov Decision Processes with Uncertain Transition Matrices. *Operations Research*, page 20, 2005.
- Junhyuk Oh, Satinder Singh, and Honglak Lee. Value Prediction Network. In *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017.
- Frans Oliehoek, Stefan Witwicki, and Leslie Kaelbling. A Sufficient Statistic for Influence in Structured Multiagent Environments. *Journal of Artificial Intelligence Research*, 70: 789–870, 2021.
- Frans A Oliehoek and Christopher Amato. *A Concise Introduction to Decentralized POMDPs*. Springer, 2016.
- OpenAI. GPT-4 Technical Report. *arXiv preprint arXiv:2303.08774*, 2024.

- OpenAI, Ilge Akkaya, Marcin Andrychowicz, Maciek Chociej, Mateusz Litwin, Bob McGrew, Arthur Petron, Alex Paino, Matthias Plappert, Glenn Powell, Raphael Ribas, Jonas Schneider, Nikolas Tezak, Jerry Tworek, Peter Welinder, Lilian Weng, Qiming Yuan, Wojciech Zaremba, and Lei Zhang. Solving Rubik's Cube with a Robot Hand. *arXiv preprint arXiv:1910.07113*, 2019a.
- OpenAI, Christopher Berner, Greg Brockman, Brooke Chan, Vicki Cheung, Przemysław Dębniak, Christy Dennison, David Farhi, Quirin Fischer, Shariq Hashme, Chris Hesse, Rafal Józefowicz, Scott Gray, Catherine Olsson, Jakub Pachocki, Michael Petrov, Henrique P. d O. Pinto, Jonathan Raiman, Tim Salimans, Jeremy Schlatter, Jonas Schneider, Szymon Sidor, Ilya Sutskever, Jie Tang, Filip Wolski, and Susan Zhang. Dota 2 with Large Scale Deep Reinforcement Learning. *arXiv preprint arXiv:1912.06680*, 2019b.
- Liam Paninski. Estimation of Entropy and Mutual Information. *Neural Computation*, 15(6): 1191–1253, 2003.
- Deepak Pathak, Pulkit Agrawal, Alexei A. Efros, and Trevor Darrell. Curiosity-driven Exploration by Self-supervised Prediction. *arXiv preprint arXiv:1705.05363*, 2017.
- Deepak Pathak, Dhiraj Gandhi, and Abhinav Gupta. Self-Supervised Exploration via Disagreement. In *Proceedings of the 36th International Conference on Machine Learning*, pages 5062–5071. PMLR, 2019.
- Benjamin Peherstorfer, Karen Willcox, and Max Gunzburger. Survey of multifidelity methods in uncertainty propagation, inference, and optimization. *Siam Review*, 60(3): 550–591, 2018.
- Y. Volkan Pehlivanoglu and Bedri Yagiz. Aerodynamic design prediction using surrogate-based modeling in genetic algorithm architecture. *Aerospace Science and Technology*, 23(1):479–491, 2012.
- Haijun Peng and Wei Wang. Adaptive surrogate model-based fast path planning for spacecraft formation reconfiguration on libration point orbits. *Aerospace Science and Technology*, 54:151–163, 2016.
- Xue Bin Peng, Marcin Andrychowicz, Wojciech Zaremba, and Pieter Abbeel. Sim-to-Real Transfer of Robotic Control with Dynamics Randomization. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 3803–3810, 2018.
- Marek Petrik and Shlomo Zilberstein. A Bilinear programming approach for multiagent planning. *Journal of Artificial Intelligence Research*, 35:235–274, 2009.
- Hieu Pham, Melody Y. Guan, Barret Zoph, Quoc V. Le, and Jeff Dean. Efficient Neural Architecture Search via parameter Sharing. In *International Conference on Machine Learning*, pages 4092–4101, 2018.
- Lerrel Pinto, James Davidson, Rahul Sukthankar, and Abhinav Gupta. Robust Adversarial Reinforcement Learning. *arXiv:1703.02702 [cs]*, 2017.

- Martin L. Puterman. *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. John Wiley & Sons, Inc., New York, NY, USA, 1st edition, 1994.
- Rahul Ramesh, Manan Tomar, and Balaraman Ravindran. Successor Options: An Option Discovery Framework for Reinforcement Learning. In *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence*, pages 3304–3310, Macao, China, 2019. International Joint Conferences on Artificial Intelligence Organization.
- B. Ravindran and A. G. Barto. Symmetries and Model Minimization in Markov Decision Processes. Technical Report, University of Massachusetts, USA, 2001.
- Saman Razavi, Bryan A. Tolson, and Donald H. Burn. Review of surrogate modeling in water resources. *Water Resources Research*, 48(7), 2012.
- Richard Evans and Jim Gao. DeepMind AI Reduces Google Data Centre Cooling Bill by 40%. <https://deepmind.google/discover/blog/deepmind-ai-reduces-google-data-centre-cooling-bill-by-40/>, 2024.
- Christopher D. Rosin. Multi-armed bandits with episode context. *Annals of Mathematics and Artificial Intelligence*, 61(3):203–230, 2011.
- Stephane Ross, Brahim Chaib-draa, and Joelle Pineau. Bayes-Adaptive POMDPs. In *Advances in Neural Information Processing Systems*, volume 20. Curran Associates, Inc., 2007.
- Stéphane Ross, Joelle Pineau, Brahim Chaib-draa, and Pierre Kreitmann. A Bayesian Approach for Learning and Planning in Partially Observable Markov Decision Processes. *Journal of Machine Learning Research*, 12(48):1729–1770, 2011.
- Sebastian Ruder. An overview of gradient descent optimization algorithms. *arXiv preprint arXiv:1609.04747*, 2016.
- Stuart J Russell and Peter Norvig. *Artificial Intelligence: A Modern Approach*. pearson, 2016.
- Julian Schrittwieser, Ioannis Antonoglou, Thomas Hubert, Karen Simonyan, Laurent Sifre, Simon Schmitt, Arthur Guez, Edward Lockhart, Demis Hassabis, and Thore Graepel. Mastering atari, go, chess and shogi by planning with a learned model. *Nature*, 588(7839): 604–609, 2020.
- John Schulman, Philipp Moritz, Sergey Levine, Michael I Jordan, and Pieter Abbeel. High-Dimensional Continuous Control Using Generalized Advantage Estimation. *arXiv preprint arXiv:1506.02438*, 2015.
- Ramanan Sekar, Oleh Rybkin, Kostas Daniilidis, Pieter Abbeel, Danijar Hafner, and Deepak Pathak. Planning to Explore via Self-Supervised World Models. In *Proceedings of the 37th International Conference on Machine Learning*, pages 8583–8592. PMLR, 2020.
- L. S. Shapley. Stochastic Games. *Proceedings of the National Academy of Sciences*, 39(10): 1095–1100, 1953.

- Pranav Shyam, Wojciech Jaśkowski, and Faustino Gomez. Model-Based Active Exploration. In *Proceedings of the 36th International Conference on Machine Learning*, pages 5779–5788. PMLR, 2019.
- David Silver and Joel Veness. Monte-Carlo planning in large POMDPs. In *Advances in Neural Information Processing Systems*, pages 2164–2172, 2010.
- David Silver, Aja Huang, Chris J. Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, Sander Dieleman, Dominik Grewe, John Nham, Nal Kalchbrenner, Ilya Sutskever, Timothy Lillicrap, Madeleine Leach, Koray Kavukcuoglu, Thore Graepel, and Demis Hassabis. Mastering the game of Go with deep neural networks and tree search. *Nature*, 529(7587):484–489, 2016.
- David Silver, Hado Hasselt, Matteo Hessel, Tom Schaul, Arthur Guez, Tim Harley, Gabriel Dulac-Arnold, David Reichert, Neil Rabinowitz, Andre Barreto, and Thomas Degris. The Predictron: End-To-End Learning and Planning. In *Proceedings of the 34th International Conference on Machine Learning*, pages 3191–3199. PMLR, 2017a.
- David Silver, Julian Schrittwieser, Karen Simonyan, Ioannis Antonoglou, Aja Huang, Arthur Guez, Thomas Hubert, Lucas Baker, Matthew Lai, Adrian Bolton, Yutian Chen, Timothy Lillicrap, Fan Hui, Laurent Sifre, George Van Den Driessche, Thore Graepel, and Demis Hassabis. Mastering the game of Go without human knowledge. *Nature*, 550(7676):354–359, 2017b.
- David Silver, Thomas Hubert, Julian Schrittwieser, Ioannis Antonoglou, Matthew Lai, Arthur Guez, Marc Lanctot, Laurent Sifre, Dharshan Kumaran, Thore Graepel, Timothy Lillicrap, Karen Simonyan, and Demis Hassabis. A general reinforcement learning algorithm that masters chess, shogi, and Go through self-play. *Science*, 362(6419):1140–1144, 2018.
- Matthijs T. J. Spaan. Partially Observable Markov Decision Processes. In Marco Wiering and Martijn van Otterlo, editors, *Reinforcement Learning: State-of-the-Art*, pages 387–414. Springer, Berlin, Heidelberg, 2012.
- Keith Stanovich. *Rationality and the Reflective Mind*. OUP USA, 2011.
- Rolf A. N. Starre, Marco Loog, Elena Congeduti, and Frans A. Oliehoek. An Analysis of Model-Based Reinforcement Learning From Abstracted Observations. *Transactions on Machine Learning Research*, 2023a.
- Rolf A. N. Starre, Marco Loog, and Frans A. Oliehoek. Model-Based Reinforcement Learning with State Abstraction: A Survey. In Toon Calders, Celine Vens, Jeffrey Lijffijt, and Bart Goethals, editors, *Artificial Intelligence and Machine Learning*, pages 133–148, Cham, 2023b. Springer Nature Switzerland.
- Miguel Suau, Jinke He, Mustafa Mert Çelikok, Matthijs Spaan, and Frans Oliehoek. Distributed Influence-Augmented Local Simulators for Parallel MARL in Large Networked Systems. In *Advances in Neural Information Processing Systems*, volume 35, pages 28305–28318, 2022a.

- Miguel Suau, Jinke He, Elena Congeduti, Rolf A. N. Starre, Aleksander Czechowski, and Frans A. Oliehoek. Influence-aware memory architectures for deep reinforcement learning in POMDPs. *Neural Computing and Applications*, 2022b.
- Miguel Suau, Jinke He, Matthijs T. J. Spaan, and Frans Oliehoek. Influence-Augmented Local Simulators: A Scalable Solution for Fast Deep RL in Large Networked Systems. In *Proceedings of the 39th International Conference on Machine Learning*, pages 20604–20624. PMLR, 2022c.
- Richard S. Sutton. Learning to predict by the methods of temporal differences. *Machine Learning*, 3(1):9–44, 1988.
- Richard S. Sutton. Dyna, an integrated architecture for learning, planning, and reacting. *ACM SIGART Bulletin*, 2(4):160–163, 1991.
- Richard S Sutton and Andrew G Barto. *Reinforcement Learning: An Introduction*. MIT press, 2018.
- Richard S. Sutton, Doina Precup, and Satinder Singh. Between MDPs and semi-MDPs: A framework for temporal abstraction in reinforcement learning. *Artificial Intelligence*, 112(1-2):181–211, 1999.
- Aviv Tamar, YI WU, Garrett Thomas, Sergey Levine, and Pieter Abbeel. Value Iteration Networks. In *Advances in Neural Information Processing Systems*, volume 29. Curran Associates, Inc., 2016.
- Jie Tan, Tingnan Zhang, Erwin Coumans, Atil Iscen, Yunfei Bai, Danijar Hafner, Steven Bohez, and Vincent Vanhoucke. Sim-to-Real: Learning Agile Locomotion For Quadruped Robots. In *Robotics: Science and Systems XIV*. Robotics: Science and Systems Foundation, 2018.
- Josh Tobin, Rachel Fong, Alex Ray, Jonas Schneider, Wojciech Zaremba, and Pieter Abbeel. Domain randomization for transferring deep neural networks from simulation to the real world. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 23–30, 2017.
- Emanuel Todorov, Tom Erez, and Yuval Tassa. MuJoCo: A physics engine for model-based control. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 5026–5033. IEEE, 2012.
- Elise van der Pol, Thomas Kipf, Frans A. Oliehoek, and Max Welling. Plannable Approximations to MDP Homomorphisms: Equivariance under Actions. In *Proceedings of the 19th International Conference on Autonomous Agents and MultiAgent Systems, AAMAS '20*, pages 1431–1439, Richland, SC, 2020a. International Foundation for Autonomous Agents and Multiagent Systems.
- Elise van der Pol, Daniel Worrall, Herke van Hoof, Frans Oliehoek, and Max Welling. MDP Homomorphic Networks: Group Symmetries in Reinforcement Learning. In *Advances in Neural Information Processing Systems*, volume 33, pages 4199–4210. Curran Associates, Inc., 2020b.

- Hado P van Hasselt, Matteo Hessel, and John Aslanides. When to use parametric models in reinforcement learning? In *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019.
- William van Melle. MYCIN: A knowledge-based consultation program for infectious disease diagnosis. *International Journal of Man-Machine Studies*, 10(3):313–322, 1978.
- Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is All you Need. In *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017.
- A. Vezhnevets, Simon Osindero, T. Schaul, N. Heess, Max Jaderberg, David Silver, and K. Kavukcuoglu. FeUdal Networks for Hierarchical Reinforcement Learning. *ArXiv*, 2017.
- Felipe A. C. Viana, Christian Gogu, and Tushar Goel. Surrogate modeling: Tricks that endured the test of time and some recent developments. *Structural and Multidisciplinary Optimization*, 64(5):2881–2908, 2021.
- Oriol Vinyals, Igor Babuschkin, Wojciech M. Czarnecki, Michaël Mathieu, Andrew Dudzik, Junyoung Chung, David H. Choi, Richard Powell, Timo Ewalds, Petko Georgiev, Junhyuk Oh, Dan Horgan, Manuel Kroiss, Ivo Danihelka, Aja Huang, Laurent Sifre, Trevor Cai, John P. Agapiou, Max Jaderberg, Alexander S. Vezhnevets, Rémi Leblond, Tobias Pohlen, Valentin Dalibard, David Budden, Yury Sulsky, James Molloy, Tom L. Paine, Caglar Gulcehre, Ziyu Wang, Tobias Pfaff, Yuhuai Wu, Roman Ring, Dani Yogatama, Dario Wünsch, Katrina McKinney, Oliver Smith, Tom Schaul, Timothy Lillicrap, Koray Kavukcuoglu, Demis Hassabis, Chris Apps, and David Silver. Grandmaster level in StarCraft II using multi-agent reinforcement learning. *Nature*, 575(7782):350–354, 2019.
- Marco Wiering and Martijn Van Otterlo, editors. *Reinforcement Learning: State-of-the-Art*, volume 12 of *Adaptation, Learning, and Optimization*. Springer Berlin Heidelberg, Berlin, Heidelberg, 2012.
- Wolfram Wiesemann, Daniel Kuhn, and Berç Rustem. Robust Markov Decision Processes. *Mathematics of Operations Research*, 38(1):153–183, 2013.
- David Wingate, Noah D. Goodman, Daniel M. Roy, Leslie P. Kaelbling, and Joshua B. Tenenbaum. Bayesian policy search with policy priors. In *Proceedings of the Twenty-Second International Joint Conference on Artificial Intelligence - Volume Volume Two*, IJCAI’11, pages 1565–1570, Barcelona, Catalonia, Spain, 2011. AAAI Press.
- Stefan J Witwicki and Edmund H Durfee. Influence-based policy abstraction for weakly-coupled Dec-POMDPs. In *ICAPS 2010 - Proceedings of the 20th International Conference on Automated Planning and Scheduling*, pages 185–192, 2010.
- Markus Wulfmeier, Abbas Abdolmaleki, Roland Hafner, Jost Tobias Springenberg, Michael Neunert, Noah Siegel, Tim Hertweck, Thomas Lampe, Nicolas Heess, and Martin Riedmiller. Compositional Transfer in Hierarchical Reinforcement Learning. *Robotics: Science and Systems XVI*, 2020.

- Peter R. Wurman, Samuel Barrett, Kenta Kawamoto, James MacGlashan, Kaushik Subramanian, Thomas J. Walsh, Roberto Capobianco, Alisa Devlic, Franziska Eckert, Florian Fuchs, Leilani Gilpin, Piyush Khandelwal, Varun Kompella, HaoChih Lin, Patrick MacAlpine, Declan Oller, Takuma Seno, Craig Sherstan, Michael D. Thomure, Houmeh Aghabozorgi, Leon Barrett, Rory Douglas, Dion Whitehead, Peter Dürr, Peter Stone, Michael Spranger, and Hiroaki Kitano. Outracing champion Gran Turismo drivers with deep reinforcement learning. *Nature*, 602(7896):223–228, 2022.
- Linjie Xu, Alexander Dockhorn, and Diego Perez-Liebana. Elastic Monte Carlo Tree Search. *IEEE Transactions on Games*, 15(4):527–537, 2023.
- Nan Ye, Adhiraj Somani, David Hsu, and Wee Sun Lee. DESPOT: Online POMDP Planning with Regularization. *Journal of Artificial Intelligence Research*, 58:231–266, 2017.
- Weirui Ye, Shaohuai Liu, Thanard Kurutach, Pieter Abbeel, and Yang Gao. Mastering Atari Games with Limited Data. In *Advances in Neural Information Processing Systems*, volume 34, pages 25476–25488. Curran Associates, Inc., 2021.
- Kenny Young and Richard S. Sutton. Iterative Option Discovery for Planning, by Planning. *arXiv preprint arXiv:2310.01569*, 2023.
- Tianhe Yu, Garrett Thomas, Lantao Yu, Stefano Ermon, James Y Zou, Sergey Levine, Chelsea Finn, and Tengyu Ma. MOPO: Model-based Offline Policy Optimization. In *Advances in Neural Information Processing Systems*, volume 33, pages 14129–14142. Curran Associates, Inc., 2020.
- Tom Zahavy, Matan Haroush, Nadav Merlis, Daniel J Mankowitz, and Shie Mannor. Learn What Not to Learn: Action Elimination with Deep Reinforcement Learning. In *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018.
- Shengyu Zhu, Ignavier Ng, and Zhitang Chen. Causal Discovery with Reinforcement Learning. In *International Conference on Learning Representations*, 2019.
- Viktor Zobernig, Richard A. Saldanha, Jinke He, Erica van der Sar, Jasper van Doorn, Jia-Chen Hua, Lachlan R. Mason, Aleksander Czechowski, Drago Indjic, Tomasz Kosmala, Alessandro Zocca, Sandjai Bhulai, Jorge Montalvo Arvizu, Claude Klöckl, and John Moriarty. RangL: A Reinforcement Learning Competition Platform. *arXiv preprint arXiv:2208.00003*, 2022.
- Barret Zoph and Quoc Le. Neural Architecture Search with Reinforcement Learning. In *International Conference on Learning Representations*, 2016.

ACKNOWLEDGMENTS

This thesis owes its existence to the wisdom, patience, and mischief of many—some walking among us, others dwelling in books, screens, or imagination.

First and foremost, I would like to express my heartfelt gratitude to my supervisors, Frans Oliehoek and Catholijn Jonker.

Frans — had you not sent that one fateful email asking whether I was interested in research, I might never have stumbled into the world of sequential decision making. That single message rerouted my future. Thank you for the years of guidance, freedom, and support — and for shaping me into a more independent and resilient researcher. One of my most memorable memories is of you, deep in battle with LaTeX at 2 a.m., while I was pondering life, the universe, and absolutely nothing related to our deadline. Your commitment was both inspiring — and a wonderfully guilt-inducing reminder that someone had to be the responsible one.

Catholijn — thank you for warmly welcoming me into the Interactive Intelligence group at the very start of my PhD journey. In those early years, your guidance helped me understand what it truly means to be a researcher — not just in skill, but in mindset and purpose. I'm deeply grateful for your wisdom and for being a role model whose example continues to inspire me.

Second, I would like to thank my committee members — Hendrik, Javier, Leslie, Matthijs, and Mathijs — for their time, thoughtful examination, and valuable feedback.

Third, I want to thank my close collaborators for making this journey more rewarding, insightful, and entertaining.

Miguel — without you, completing my PhD might still have been possible (not really), but it would have been far less fun. As you wrote in your acknowledgements, some of my happiest PhD memories are too of our wild, hilarious discussions about decision making — too fun and too chaotic to ever lead to a publication, but memorable nonetheless. Your, let's say, creative exploration of paper-writing strategies (you know what I mean) has forever shaped the way I think about the reviewing system — though whether that's in a good way or a bad way remains unclear. Jokes aside, I genuinely learned a lot from you — especially about how to present ideas with clarity and flair — and I was inspired by your passion.

Hendrik — thank you for your consistently constructive feedback, not only on our IJCAI paper but also on this very thesis. I truly admire your attention to detail and your sharp editorial eye — both have improved my work immensely.

Michael — thank you for patiently teaching me how to draw meaningful statistical conclusions from experimental data, and for making time to meet and discuss, no matter how busy you were. Your clarity and kindness made a big impact.

Thomas — thank you for showing me the value of structured research planning. Your methodical approach helped me stay grounded when things felt chaotic — which, as you know, was more often than not.

And finally, Joery — thank you for being such a fun collaborator. I'm not entirely sure what I learned from you... but I'm sure it was important.

I would also like to thank all the talented bachelor and master students I had the privilege to work with, as well as their co-supervisors. Watching you grow into thoughtful, curious researchers has been one of my proudest moments at TU Delft. You've given me hope for the future — a future shaped by minds that ask good questions, challenge assumptions, and occasionally even write code that doesn't need my help to debug.

Next, I want to thank my colleagues at TU Delft. First of all, thanks to my colleagues from the INFLUENCE project: Aleks, Elena, Rolf, and Alex. Thank you for being such kind officemates — and my very first batch of friends in the Netherlands. I really miss you all. And a special thanks to the desk by the door — the one that vibrates dramatically every time someone enters or leaves the office. You truly kept me grounded, both literally and figuratively. I'm also grateful to my colleagues at the Interactive Intelligence group — for the human-human interactions. That said, I suspect most of you won't read this thesis... since, let's face it, most of you graduated before me — despite starting after me (sigh).

I'd also like to thank my colleagues at the Sequential Decision-Making group.

Matthijs — thank you for your continuous support throughout my PhD and Postdoc, and for making the group exist in the first place. Without it, I wouldn't have had the chance to work alongside the people I'm about to mention.

Wendelin — thank you for maintaining the RL Mattermost channel, which became my main (if not only) source of papers. I tried to click the like button whenever I saw something useful, but let me say it properly here: thanks for sharing, it truly made a difference.

Yaniv — thank you for sharing your view of the world with me, and for offering so much thoughtful life advice. I'll carry your words with me for a long time. Also, thank you for always taking my joke questions seriously.

To my wonderful officemates — Max, Guyon, Caroline, Laurens, and Thiago — thank you for creating such a pleasant working environment.

To those in other offices — Pascal, Moritz, Andreea, Julia, and Stephan — thank you for the discussions during group meetings, the casual chats, and for unknowingly letting me take the better office. Nothing personal.

Zuzanna — thank you for simply being a kind and lovely person.

Mert — I've enjoyed the ice cream and the malatang. Surely, somewhere else, in some other context, we will share something more.

Oussama - thank you for your dark humor — sometimes it really hurt, but I have to admit, you'd probably do great as a stand-up comedian.

Ariyan — thank you for the mentorship and your kindness. Sometimes your insights really hurt — which I think means they were working.

And Davide — best of luck with your career as a street mathematician. To anyone reading this: do yourself a favor and don't debate with Davide about anything. You already lose when you start.

Thanks also to Ruud, Anita, and Vanessa for their hardware and administrative support. Your support kept everything running smoothly behind the scenes, and I'm truly grateful.

I would also like to thank my Cyber Security neighbours. From Building 28 to Echo, you still haven't gotten rid of me :).

Finally, I've most definitely forgotten someone important. Thank you, [your name]!

接下来，我想要感谢一些我在代尔夫特的朋友们。刘程，感谢你在几乎所有游戏中输给我之后还能有再来一局的勇气。当然，你赢了最后一把大富翁，厉害厉害。孙博，感谢你做一个好人。这个世界已经够复杂了，你的存在让它简单了一点（不是yygq）。文君涵，感谢你从利物浦寄来的明信片。老杨，感谢你的蛋炒饭和清炖羊肉。子龙、Yanbo、Ze Chang，感谢我们一起打的网球。Yun、小韦，感谢我们一起玩的桌游。还有Fenghua, Biyue, Wei, Xueer, Yang, Ruipeng, Shilun, Li, Ding Ding, Longjian, Shiwen。也许再见遥遥无期，但回忆永远在线，服务器永不宕机。另外，感谢在ECAI一起吃喝玩乐的朋友。

我最大的幸运之一，是拥有一群一起长大的朋友。虽然如今我们分散在世界各地，但我们从未真正远离。每一次相聚都太短，每一次重逢都太远，但你们始终是我生命里最牢固的支柱。

还有一些朋友，他们从未真实存在，却曾照亮过我的现实。太中二了，就不提名字了。总有一些，高于其他。光就是一切的意义。

特别感谢我的女朋友，Dr. Xu。读博让我明白一个道理：人终究只能靠自己拯救自己。但有些人，会让你相信自己值得被拯救。感谢你，一直试着理解那个连我自己都无法理解的我。感谢你，让我相信，爱是一种可以连接冰冷的宇宙的力量，让一切未解的困惑变得值得等待。

最后的最后，我想要感谢我的父母和家庭为我所做的一切。他们说，好的童年可以照亮一生。我始终记得那道光从何而来。你们是我永远的归途。

作为结束，我想留一句话送给未来的自己，来自《大鱼海棠》：

“这短短的一生，我们最终都会失去。
你不妨大胆一些，爱一个人，攀一座山，追一个梦。”

也想把一段歌词，送给过去的自己和曾经的梦，来自李宗盛的《爱的代价》：

“还记得年少时的梦吗，
像朵永远不凋零的花，
陪我经过那风吹雨打，
看世事无常，
看沧桑变化。”

何晋科
2025年5月13日

CURRICULUM VITÆ

Jinke HE

1996/03/13 Born in Changzhou, China


EDUCATION


2014-2016	Bachelor of Science in Information and Computing Science <i>Xi'an Jiaotong-Liverpool University</i> First-class Honours
2016-2018	Bachelor of Science in Computer Science <i>University of Liverpool</i> First-class Honours
2018-2019	Master of Science in Computer Science <i>University of Oxford</i> Distinction


WORK EXPERIENCE

2018	Research Intern <i>Latent Logic (acquired by Waymo)</i>
2019-2024	PhD Researcher <i>Delft University of Technology</i>
2024-2025	Postdoctoral Researcher <i>Delft University of Technology and Booking.com</i>

LIST OF PUBLICATIONS

 Included in this thesis.

1. Soft Option Transfer
Jinke He, Maximilian Igl, Matthew Smith, Wendelin Boehmer, Shimon Whiteson
Learning Transferable Skills Workshop at **NeurIPS**, 2019
2. Multitask Soft Option Learning
Maximilian Igl, Andrew Gambardella, **Jinke He**, Nantas Nardelli, N. Siddharth, Wendelin Boehmer, Shimon Whiteson
Conference on Uncertainty in Artificial Intelligence (**UAI**), 2020
3.  Influence-Augmented Online Planning for Complex Environments
Jinke He, Miguel Suau, Frans A. Oliehoek
Conference on Neural Information Processing Systems (**NeurIPS**), 2020
4.  Online Planning in POMDPs with Self-Improving Simulator
Jinke He, Miguel Suau, Hendrik Baier, Michael Kaisers, Frans A. Oliehoek
International Joint Conference on Artificial Intelligence (**IJCAI**), 2022
5. RangL: A Reinforcement Learning Competition Platform
Viktor Zobernig, Richard A. Saldanha, **Jinke He**, Erica van der Sar, Jasper van Doorn, Jia-Chen Hua, Lachlan R. Mason, Aleksander Czechowski, Drago Indjic, Tomasz Kosmala, Alessandro Zocca, Sandjai Bhulai, Jorge Montalvo Arvizu, Claude Klöckl, John Moriarty
arXiv preprint arXiv:2208.00003, 2022
6. Robust Ensemble Adversarial Model-Based Reinforcement Learning
Daniele Foffano, **Jinke He**, Frans A. Oliehoek
Adaptive and Learning Agents Workshop at **AAMAS**, 2022
7. Influence-Augmented Local Simulators: A Scalable Solution for Fast Deep RL in Large Networked Systems
Miguel Suau, **Jinke He**, Matthijs T. J. Spaan, Frans A. Oliehoek
International Conference on Machine Learning (**ICML**), 2022
8. Influence-aware Memory Architectures for Deep Reinforcement Learning
Miguel Suau, **Jinke He**, Elena Congeduti, Rolf A.N. Starre, Aleksander Czechowski, Frans A. Oliehoek
Neural Computing and Applications (**NCAA**), 2022
9. Distributed Influence-Augmented Local Simulators for Parallel MARL in Large Networked Systems
Miguel Suau, **Jinke He**, Mustafa Mert Çelikok, Matthijs T. J. Spaan, Frans A. Oliehoek
Conference on Neural Information Processing Systems (**NeurIPS**), 2022

10. Benchmarking Robustness and Generalization in Multi-Agent Systems: A Case Study on Neural MMO
Yangkun Chen, Joseph Suarez, Junjie Zhang, Chenghui Yu, Bo Wu, Hanmo Chen, Hengman Zhu, Rui Du, Shanliang Qian, Shuai Liu, Weijun Hong, **Jinke He**, Yibing Zhang, Liang Zhao, Clare Zhu, Julian Togelius, Sharada Mohanty, Jiaxin Chen, Xiu Li, Xiaolong Zhu, Phillip Isola
International Conference on Autonomous Agents and Multiagent Systems (**AAMAS**), 2023
11.  What model does MuZero learn?
Jinke He, Thomas M. Moerland, Joery A. de Vries, Frans A. Oliehoek
European Conference on Artificial Intelligence (**ECAI**), 2024
12. Timing the Match: A Deep Reinforcement Learning Approach for Ride-Hailing and Ride-Pooling Services
Yiman Bao, Jie Gao, **Jinke He**, Frans A. Oliehoek, Oded Cats
arXiv preprint arXiv:2503.13200 (under review), 2025
13. Trust-Region Twisted Policy Improvement
Joery A. de Vries, **Jinke He**, Yaniv Oren, Matthijs T.J. Spaan
International Conference on Machine Learning (**ICML**), 2025
14. Bayesian Meta-Reinforcement Learning with Laplace Variational Recurrent Networks
Joery A. de Vries, **Jinke He**, Mathijs M. de Weerd, Matthijs T.J. Spaan
Reinforcement Learning Conference (**RLC**), 2025

LIST OF SUPERVISED STUDENT THESES

1. Learning Models for Planning in Reinforcement Learning
David Moolenaar, Ynze ter Horst, Maarten Lips, Nick Yu, Thijs Molendijk
Bachelor Seminar, 2020, TU Delft
Co-supervised with Frans Oliehoek
2. Q-value Reuse Between State Abstractions for Traffic Light
Emanuel Kuhn
Bachelor Thesis, 2020, TU Delft
Co-supervised with Rolf Starre and Frans Oliehoek
3. Open problems in Model-based Reinforcement Learning
Daniele Foffano
Master Literature Survey, 2021, TU Delft
Co-supervised with Frans Oliehoek
4. Robust Ensemble Adversarial Model-Based Reinforcement Learning
Daniele Foffano
Master Thesis, 2022, TU Delft
Co-supervised with Frans Oliehoek
5. Action Selection in MCTS
Bugra Veysel Yildiz Yildiz, Emil Malmsten, Henwei Zeng, Igor Witkowski
Intelligent Decision Making Project, 2024, TU Delft
Co-supervised with Joery de Vries
6. See Clearly, Act Intelligently: Transformers in Transparent Environments
Omar Elamin
Bachelor Thesis, 2024, TU Delft
Co-supervised with Frans Oliehoek
7. Acting in the Face of Uncertainty: Pessimism in Offline Model-Based Reinforcement Learning
Sten van Wolfswinkel
Bachelor Thesis, 2024, TU Delft
Co-supervised with Frans Oliehoek
8. Generalisation Ability of Proper Value Equivalence Models in Model-Based Reinforcement Learning
Severin Bratus
Bachelor Thesis, 2024, TU Delft
Co-supervised with Frans Oliehoek

9. Understanding the Effects of Discrete Representations in Model-Based Reinforcement Learning
Mihai Mitrea
Bachelor Thesis, 2024, TU Delft
Co-supervised with Frans Oliehoek
10. Optimize the Matching Time Intervals in (Shared) Ride-Hailing Services Using Reinforcement Learning
Yiman Bao
Master Thesis, 2024, TU Delft
Co-supervised with Jie Gao, Frans Oliehoek, and Oded Cats
11. Efficient Planning through LLM-based State Abstractions
Dennis Lentschig
Master Thesis, 2025 (ongoing), TU Delft
Co-supervised with Frans Oliehoek
12. Investigating Improvements into the Exploration Method of MuZero
Francisco Ruas Vaz
Bachelor Thesis, 2025 (ongoing), TU Delft
Co-supervised with Frans Oliehoek
13. Action Sampling Strategies in Sampled MuZero for Continuous Control Tasks
Vaclav Kuboň
Bachelor Thesis, 2025 (ongoing), TU Delft
Co-supervised with Frans Oliehoek
14. Sparse Attention Mechanisms for Robust Short-Term Precision and Long-Term Planning in Atari
Daniel De Dios Allegue
Bachelor Thesis, 2025 (ongoing), TU Delft
Co-supervised with Frans Oliehoek
15. Normalising Flows for Modelling Stochasticity in MuZero
Bogdan Damian
Bachelor Thesis, 2025 (ongoing), TU Delft
Co-supervised with Frans Oliehoek
16. The impact of model learning losses on MuZero in Atari
Daniel Popovici
Bachelor Thesis, 2025 (ongoing), TU Delft
Co-supervised with Frans Oliehoek

SIKS DISSERTATIONS

- 2016 01 Syed Saiden Abbas (RUN), Recognition of Shapes by Humans and Machines
- 02 Michiel Christiaan Meulendijk (UU), Optimizing medication reviews through decision support: prescribing a better pill to swallow
- 03 Maya Sappelli (RUN), Knowledge Work in Context: User Centered Knowledge Worker Support
- 04 Laurens Rietveld (VUA), Publishing and Consuming Linked Data
- 05 Evgeny Sherkhonov (UvA), Expanded Acyclic Queries: Containment and an Application in Explaining Missing Answers
- 06 Michel Wilson (TUD), Robust scheduling in an uncertain environment
- 07 Jeroen de Man (VUA), Measuring and modeling negative emotions for virtual training
- 08 Matje van de Camp (TiU), A Link to the Past: Constructing Historical Social Networks from Unstructured Data
- 09 Archana Nottamkandath (VUA), Trusting Crowdsourced Information on Cultural Artefacts
- 10 George Karafotias (VUA), Parameter Control for Evolutionary Algorithms
- 11 Anne Schuth (UvA), Search Engines that Learn from Their Users
- 12 Max Knobbout (UU), Logics for Modelling and Verifying Normative Multi-Agent Systems
- 13 Nana Baah Gyan (VUA), The Web, Speech Technologies and Rural Development in West Africa - An ICT4D Approach
- 14 Ravi Khadka (UU), Revisiting Legacy Software System Modernization
- 15 Steffen Michels (RUN), Hybrid Probabilistic Logics - Theoretical Aspects, Algorithms and Experiments
- 16 Guangliang Li (UvA), Socially Intelligent Autonomous Agents that Learn from Human Reward
- 17 Berend Weel (VUA), Towards Embodied Evolution of Robot Organisms
- 18 Albert Meroño Peñuela (VUA), Refining Statistical Data on the Web
- 19 Julia Efremova (TU/e), Mining Social Structures from Genealogical Data
- 20 Daan Odijk (UvA), Context & Semantics in News & Web Search
- 21 Alejandro Moreno Céleri (UT), From Traditional to Interactive Playspaces: Automatic Analysis of Player Behavior in the Interactive Tag Playground
- 22 Grace Lewis (VUA), Software Architecture Strategies for Cyber-Foraging Systems
- 23 Fei Cai (UvA), Query Auto Completion in Information Retrieval
- 24 Brend Wanders (UT), Repurposing and Probabilistic Integration of Data; An Iterative and data model independent approach

- 25 Julia Kiseleva (TU/e), Using Contextual Information to Understand Searching and Browsing Behavior
- 26 Dilhan Thilakarathne (VUA), In or Out of Control: Exploring Computational Models to Study the Role of Human Awareness and Control in Behavioural Choices, with Applications in Aviation and Energy Management Domains
- 27 Wen Li (TUD), Understanding Geo-spatial Information on Social Media
- 28 Mingxin Zhang (TUD), Large-scale Agent-based Social Simulation - A study on epidemic prediction and control
- 29 Nicolas Höning (TUD), Peak reduction in decentralised electricity systems - Markets and prices for flexible planning
- 30 Ruud Mattheij (TiU), The Eyes Have It
- 31 Mohammad Khelghati (UT), Deep web content monitoring
- 32 Eelco Vriezekolk (UT), Assessing Telecommunication Service Availability Risks for Crisis Organisations
- 33 Peter Bloem (UvA), Single Sample Statistics, exercises in learning from just one example
- 34 Dennis Schunselaar (TU/e), Configurable Process Trees: Elicitation, Analysis, and Enactment
- 35 Zhaochun Ren (UvA), Monitoring Social Media: Summarization, Classification and Recommendation
- 36 Daphne Karreman (UT), Beyond R2D2: The design of nonverbal interaction behavior optimized for robot-specific morphologies
- 37 Giovanni Sileno (UvA), Aligning Law and Action - a conceptual and computational inquiry
- 38 Andrea Minuto (UT), Materials that Matter - Smart Materials meet Art & Interaction Design
- 39 Merijn Bruijnes (UT), Believable Suspect Agents; Response and Interpersonal Style Selection for an Artificial Suspect
- 40 Christian Detweiler (TUD), Accounting for Values in Design
- 41 Thomas King (TUD), Governing Governance: A Formal Framework for Analysing Institutional Design and Enactment Governance
- 42 Spyros Martzoukos (UvA), Combinatorial and Compositional Aspects of Bilingual Aligned Corpora
- 43 Saskia Koldijk (RUN), Context-Aware Support for Stress Self-Management: From Theory to Practice
- 44 Thibault Sellam (UvA), Automatic Assistants for Database Exploration
- 45 Bram van de Laar (UT), Experiencing Brain-Computer Interface Control
- 46 Jorge Gallego Perez (UT), Robots to Make you Happy
- 47 Christina Weber (UL), Real-time foresight - Preparedness for dynamic innovation networks
- 48 Tanja Buttler (TUD), Collecting Lessons Learned
- 49 Gleb Polevoy (TUD), Participation and Interaction in Projects. A Game-Theoretic Analysis
- 50 Yan Wang (TiU), The Bridge of Dreams: Towards a Method for Operational Performance Alignment in IT-enabled Service Supply Chains

- 2017 01 Jan-Jaap Oerlemans (UL), Investigating Cybercrime
- 02 Sjoerd Timmer (UU), Designing and Understanding Forensic Bayesian Networks using Argumentation
- 03 Daniël Harold Telgen (UU), Grid Manufacturing; A Cyber-Physical Approach with Autonomous Products and Reconfigurable Manufacturing Machines
- 04 Mrunal Gawade (CWI), Multi-core Parallelism in a Column-store
- 05 Mahdiah Shadi (UvA), Collaboration Behavior
- 06 Damir Vandic (EUR), Intelligent Information Systems for Web Product Search
- 07 Roel Bertens (UU), Insight in Information: from Abstract to Anomaly
- 08 Rob Konijn (VUA), Detecting Interesting Differences: Data Mining in Health Insurance Data using Outlier Detection and Subgroup Discovery
- 09 Dong Nguyen (UT), Text as Social and Cultural Data: A Computational Perspective on Variation in Text
- 10 Robby van Delden (UT), (Steering) Interactive Play Behavior
- 11 Florian Kunneman (RUN), Modelling patterns of time and emotion in Twitter #anticipointment
- 12 Sander Leemans (TU/e), Robust Process Mining with Guarantees
- 13 Gijs Huisman (UT), Social Touch Technology - Extending the reach of social touch through haptic technology
- 14 Shoshannah Tekofsky (TiU), You Are Who You Play You Are: Modelling Player Traits from Video Game Behavior
- 15 Peter Berck (RUN), Memory-Based Text Correction
- 16 Aleksandr Chuklin (UvA), Understanding and Modeling Users of Modern Search Engines
- 17 Daniel Dimov (UL), Crowdsourced Online Dispute Resolution
- 18 Ridho Reinanda (UvA), Entity Associations for Search
- 19 Jeroen Vuurens (UT), Proximity of Terms, Texts and Semantic Vectors in Information Retrieval
- 20 Mohammadbashir Sedighi (TUD), Fostering Engagement in Knowledge Sharing: The Role of Perceived Benefits, Costs and Visibility
- 21 Jeroen Linssen (UT), Meta Matters in Interactive Storytelling and Serious Gaming (A Play on Worlds)
- 22 Sara Magliacane (VUA), Logics for causal inference under uncertainty
- 23 David Graus (UvA), Entities of Interest – Discovery in Digital Traces
- 24 Chang Wang (TUD), Use of Affordances for Efficient Robot Learning
- 25 Veruska Zamborlini (VUA), Knowledge Representation for Clinical Guidelines, with applications to Multimorbidity Analysis and Literature Search
- 26 Merel Jung (UT), Socially intelligent robots that understand and respond to human touch
- 27 Michiel Joosse (UT), Investigating Positioning and Gaze Behaviors of Social Robots: People's Preferences, Perceptions and Behaviors
- 28 John Klein (VUA), Architecture Practices for Complex Contexts
- 29 Adel Alhuraibi (TiU), From IT-Business Strategic Alignment to Performance: A Moderated Mediation Model of Social Innovation, and Enterprise Governance of IT"

- 30 Wilma Latuny (TiU), The Power of Facial Expressions
 - 31 Ben Ruijl (UL), Advances in computational methods for QFT calculations
 - 32 Thaer Samar (RUN), Access to and Retrievalability of Content in Web Archives
 - 33 Brigit van Loggem (OU), Towards a Design Rationale for Software Documentation: A Model of Computer-Mediated Activity
 - 34 Maren Scheffel (OU), The Evaluation Framework for Learning Analytics
 - 35 Martine de Vos (VUA), Interpreting natural science spreadsheets
 - 36 Yuanhao Guo (UL), Shape Analysis for Phenotype Characterisation from High-throughput Imaging
 - 37 Alejandro Montes Garcia (TU/e), WiBAF: A Within Browser Adaptation Framework that Enables Control over Privacy
 - 38 Alex Kayal (TUD), Normative Social Applications
 - 39 Sara Ahmadi (RUN), Exploiting properties of the human auditory system and compressive sensing methods to increase noise robustness in ASR
 - 40 Altaf Hussain Abro (VUA), Steer your Mind: Computational Exploration of Human Control in Relation to Emotions, Desires and Social Support For applications in human-aware support systems
 - 41 Adnan Manzoor (VUA), Minding a Healthy Lifestyle: An Exploration of Mental Processes and a Smart Environment to Provide Support for a Healthy Lifestyle
 - 42 Elena Sokolova (RUN), Causal discovery from mixed and missing data with applications on ADHD datasets
 - 43 Maaïke de Boer (RUN), Semantic Mapping in Video Retrieval
 - 44 Garm Lucassen (UU), Understanding User Stories - Computational Linguistics in Agile Requirements Engineering
 - 45 Bas Testerink (UU), Decentralized Runtime Norm Enforcement
 - 46 Jan Schneider (OU), Sensor-based Learning Support
 - 47 Jie Yang (TUD), Crowd Knowledge Creation Acceleration
 - 48 Angel Suarez (OU), Collaborative inquiry-based learning
-
- 2018 01 Han van der Aa (VUA), Comparing and Aligning Process Representations
 - 02 Felix Mannhardt (TU/e), Multi-perspective Process Mining
 - 03 Steven Bosems (UT), Causal Models For Well-Being: Knowledge Modeling, Model-Driven Development of Context-Aware Applications, and Behavior Prediction
 - 04 Jordan Janeiro (TUD), Flexible Coordination Support for Diagnosis Teams in Data-Centric Engineering Tasks
 - 05 Hugo Huurdeman (UvA), Supporting the Complex Dynamics of the Information Seeking Process
 - 06 Dan Ionita (UT), Model-Driven Information Security Risk Assessment of Socio-Technical Systems
 - 07 Jieting Luo (UU), A formal account of opportunism in multi-agent systems
 - 08 Rick Smetsers (RUN), Advances in Model Learning for Software Systems
 - 09 Xu Xie (TUD), Data Assimilation in Discrete Event Simulations
 - 10 Julienka Mollee (VUA), Moving forward: supporting physical activity behavior change through intelligent technology

- 11 Mahdi Sargolzaei (UvA), Enabling Framework for Service-oriented Collaborative Networks
 - 12 Xixi Lu (TU/e), Using behavioral context in process mining
 - 13 Seyed Amin Tabatabaei (VUA), Computing a Sustainable Future
 - 14 Bart Joosten (TiU), Detecting Social Signals with Spatiotemporal Gabor Filters
 - 15 Naser Davarzani (UM), Biomarker discovery in heart failure
 - 16 Jaebok Kim (UT), Automatic recognition of engagement and emotion in a group of children
 - 17 Jianpeng Zhang (TU/e), On Graph Sample Clustering
 - 18 Henriette Nakad (UL), De Notaris en Private Rechtspraak
 - 19 Minh Duc Pham (VUA), Emergent relational schemas for RDF
 - 20 Manxia Liu (RUN), Time and Bayesian Networks
 - 21 Aad Sloatmaker (OU), EMERGO: a generic platform for authoring and playing scenario-based serious games
 - 22 Eric Fernandes de Mello Araújo (VUA), Contagious: Modeling the Spread of Behaviours, Perceptions and Emotions in Social Networks
 - 23 Kim Schouten (EUR), Semantics-driven Aspect-Based Sentiment Analysis
 - 24 Jered Vroon (UT), Responsive Social Positioning Behaviour for Semi-Autonomous Telepresence Robots
 - 25 Riste Gligorov (VUA), Serious Games in Audio-Visual Collections
 - 26 Roelof Anne Jelle de Vries (UT), Theory-Based and Tailor-Made: Motivational Messages for Behavior Change Technology
 - 27 Maikel Leemans (TU/e), Hierarchical Process Mining for Scalable Software Analysis
 - 28 Christian Willemse (UT), Social Touch Technologies: How they feel and how they make you feel
 - 29 Yu Gu (TiU), Emotion Recognition from Mandarin Speech
 - 30 Wouter Beek (VUA), The "K" in "semantic web" stands for "knowledge": scaling semantics to the web
-
- 2019 01 Rob van Eijk (UL), Web privacy measurement in real-time bidding systems. A graph-based approach to RTB system classification
 - 02 Emmanuelle Beauxis Aussalet (CWI, UU), Statistics and Visualizations for Assessing Class Size Uncertainty
 - 03 Eduardo Gonzalez Lopez de Murillas (TU/e), Process Mining on Databases: Extracting Event Data from Real Life Data Sources
 - 04 Ridho Rahmadi (RUN), Finding stable causal structures from clinical data
 - 05 Sebastiaan van Zelst (TU/e), Process Mining with Streaming Data
 - 06 Chris Dijkshoorn (VUA), Nichesourcing for Improving Access to Linked Cultural Heritage Datasets
 - 07 Soude Fazeli (TUD), Recommender Systems in Social Learning Platforms
 - 08 Frits de Nijs (TUD), Resource-constrained Multi-agent Markov Decision Processes
 - 09 Fahimeh Alizadeh Moghaddam (UvA), Self-adaptation for energy efficiency in software systems

- 10 Qing Chuan Ye (EUR), Multi-objective Optimization Methods for Allocation and Prediction
- 11 Yue Zhao (TUD), Learning Analytics Technology to Understand Learner Behavioral Engagement in MOOCs
- 12 Jacqueline Heinerman (VUA), Better Together
- 13 Guanliang Chen (TUD), MOOC Analytics: Learner Modeling and Content Generation
- 14 Daniel Davis (TUD), Large-Scale Learning Analytics: Modeling Learner Behavior & Improving Learning Outcomes in Massive Open Online Courses
- 15 Erwin Walraven (TUD), Planning under Uncertainty in Constrained and Partially Observable Environments
- 16 Guangming Li (TU/e), Process Mining based on Object-Centric Behavioral Constraint (OCBC) Models
- 17 Ali Hurriyetoglu (RUN), Extracting actionable information from microtexts
- 18 Gerard Wagenaar (UU), Artefacts in Agile Team Communication
- 19 Vincent Koeman (TUD), Tools for Developing Cognitive Agents
- 20 Chide Groenouwe (UU), Fostering technically augmented human collective intelligence
- 21 Cong Liu (TU/e), Software Data Analytics: Architectural Model Discovery and Design Pattern Detection
- 22 Martin van den Berg (VUA), Improving IT Decisions with Enterprise Architecture
- 23 Qin Liu (TUD), Intelligent Control Systems: Learning, Interpreting, Verification
- 24 Anca Dumitrache (VUA), Truth in Disagreement - Crowdsourcing Labeled Data for Natural Language Processing
- 25 Emiel van Miltenburg (VUA), Pragmatic factors in (automatic) image description
- 26 Prince Singh (UT), An Integration Platform for Synchromodal Transport
- 27 Alessandra Antonaci (OU), The Gamification Design Process applied to (Massive) Open Online Courses
- 28 Esther Kuindersma (UL), Cleared for take-off: Game-based learning to prepare airline pilots for critical situations
- 29 Daniel Formolo (VUA), Using virtual agents for simulation and training of social skills in safety-critical circumstances
- 30 Vahid Yazdanpanah (UT), Multiagent Industrial Symbiosis Systems
- 31 Milan Jelisavcic (VUA), Alive and Kicking: Baby Steps in Robotics
- 32 Chiara Sironi (UM), Monte-Carlo Tree Search for Artificial General Intelligence in Games
- 33 Anil Yaman (TU/e), Evolution of Biologically Inspired Learning in Artificial Neural Networks
- 34 Negar Ahmadi (TU/e), EEG Microstate and Functional Brain Network Features for Classification of Epilepsy and PNES
- 35 Lisa Facey-Shaw (OU), Gamification with digital badges in learning programming
- 36 Kevin Ackermans (OU), Designing Video-Enhanced Rubrics to Master Complex Skills
- 37 Jian Fang (TUD), Database Acceleration on FPGAs
- 38 Akos Kadar (OU), Learning visually grounded and multilingual representations

- 2020 01 Armon Toubman (UL), Calculated Moves: Generating Air Combat Behaviour
- 02 Marcos de Paula Bueno (UL), Unraveling Temporal Processes using Probabilistic Graphical Models
- 03 Mostafa Deghani (UvA), Learning with Imperfect Supervision for Language Understanding
- 04 Maarten van Gompel (RUN), Context as Linguistic Bridges
- 05 Yulong Pei (TU/e), On local and global structure mining
- 06 Preethu Rose Anish (UT), Stimulation Architectural Thinking during Requirements Elicitation - An Approach and Tool Support
- 07 Wim van der Vegt (OU), Towards a software architecture for reusable game components
- 08 Ali Mirsoleimani (UL), Structured Parallel Programming for Monte Carlo Tree Search
- 09 Myriam Traub (UU), Measuring Tool Bias and Improving Data Quality for Digital Humanities Research
- 10 Alifah Syamsiyah (TU/e), In-database Preprocessing for Process Mining
- 11 Sepideh Mesbah (TUD), Semantic-Enhanced Training Data Augmentation Methods for Long-Tail Entity Recognition Models
- 12 Ward van Breda (VUA), Predictive Modeling in E-Mental Health: Exploring Applicability in Personalised Depression Treatment
- 13 Marco Virgolin (CWI), Design and Application of Gene-pool Optimal Mixing Evolutionary Algorithms for Genetic Programming
- 14 Mark Raasveldt (CWI/UL), Integrating Analytics with Relational Databases
- 15 Konstantinos Georgiadis (OU), Smart CAT: Machine Learning for Configurable Assessments in Serious Games
- 16 Ilona Wilmont (RUN), Cognitive Aspects of Conceptual Modelling
- 17 Daniele Di Mitri (OU), The Multimodal Tutor: Adaptive Feedback from Multimodal Experiences
- 18 Georgios Methenitis (TUD), Agent Interactions & Mechanisms in Markets with Uncertainties: Electricity Markets in Renewable Energy Systems
- 19 Guido van Capelleveen (UT), Industrial Symbiosis Recommender Systems
- 20 Albert Hankel (VUA), Embedding Green ICT Maturity in Organisations
- 21 Karine da Silva Miras de Araujo (VUA), Where is the robot?: Life as it could be
- 22 Maryam Masoud Khamis (RUN), Understanding complex systems implementation through a modeling approach: the case of e-government in Zanzibar
- 23 Rianne Conijn (UT), The Keys to Writing: A writing analytics approach to studying writing processes using keystroke logging
- 24 Lenin da Nóbrega Medeiros (VUA/RUN), How are you feeling, human? Towards emotionally supportive chatbots
- 25 Xin Du (TU/e), The Uncertainty in Exceptional Model Mining
- 26 Krzysztof Leszek Sadowski (UU), GAMBIT: Genetic Algorithm for Model-Based mixed-Integer optimization
- 27 Ekaterina Muravyeva (TUD), Personal data and informed consent in an educational context

-
- 28 Bibeg Limbu (TUD), Multimodal interaction for deliberate practice: Training complex skills with augmented reality
 - 29 Ioan Gabriel Bucur (RUN), Being Bayesian about Causal Inference
 - 30 Bob Zadok Blok (UL), Creatief, Creatiever, Creatiefst
 - 31 Gongjin Lan (VUA), Learning better – From Baby to Better
 - 32 Jason Rhuggenaath (TU/e), Revenue management in online markets: pricing and online advertising
 - 33 Rick Gilsing (TU/e), Supporting service-dominant business model evaluation in the context of business model innovation
 - 34 Anna Bon (UM), Intervention or Collaboration? Redesigning Information and Communication Technologies for Development
 - 35 Siamak Farshidi (UU), Multi-Criteria Decision-Making in Software Production
-
- 2021 01 Francisco Xavier Dos Santos Fonseca (TUD), Location-based Games for Social Interaction in Public Space
 - 02 Rijk Mercur (TUD), Simulating Human Routines: Integrating Social Practice Theory in Agent-Based Models
 - 03 Seyyed Hadi Hashemi (UvA), Modeling Users Interacting with Smart Devices
 - 04 Ioana Jivet (OU), The Dashboard That Loved Me: Designing adaptive learning analytics for self-regulated learning
 - 05 Davide Dell'Anna (UU), Data-Driven Supervision of Autonomous Systems
 - 06 Daniel Davison (UT), "Hey robot, what do you think?" How children learn with a social robot
 - 07 Armel Lefebvre (UU), Research data management for open science
 - 08 Nardie Fanchamps (OU), The Influence of Sense-Reason-Act Programming on Computational Thinking
 - 09 Cristina Zaga (UT), The Design of Robothings. Non-Anthropomorphic and Non-Verbal Robots to Promote Children's Collaboration Through Play
 - 10 Quinten Meertens (UvA), Misclassification Bias in Statistical Learning
 - 11 Anne van Rossum (UL), Nonparametric Bayesian Methods in Robotic Vision
 - 12 Lei Pi (UL), External Knowledge Absorption in Chinese SMEs
 - 13 Bob R. Schadenberg (UT), Robots for Autistic Children: Understanding and Facilitating Predictability for Engagement in Learning
 - 14 Negin Samaeemofrad (UL), Business Incubators: The Impact of Their Support
 - 15 Onat Ege Adali (TU/e), Transformation of Value Propositions into Resource Re-Configurations through the Business Services Paradigm
 - 16 Esam A. H. Ghaleb (UM), Bimodal emotion recognition from audio-visual cues
 - 17 Dario Dotti (UM), Human Behavior Understanding from motion and bodily cues using deep neural networks
 - 18 Remi Wieten (UU), Bridging the Gap Between Informal Sense-Making Tools and Formal Systems - Facilitating the Construction of Bayesian Networks and Argumentation Frameworks
 - 19 Roberto Verdecchia (VUA), Architectural Technical Debt: Identification and Management
 - 20 Masoud Mansoury (TU/e), Understanding and Mitigating Multi-Sided Exposure Bias in Recommender Systems

-
- 21 Pedro Thiago Timbó Holanda (CWI), Progressive Indexes
 - 22 Sihang Qiu (TUD), Conversational Crowdsourcing
 - 23 Hugo Manuel Proença (UL), Robust rules for prediction and description
 - 24 Kaijie Zhu (TU/e), On Efficient Temporal Subgraph Query Processing
 - 25 Eoin Martino Grua (VUA), The Future of E-Health is Mobile: Combining AI and Self-Adaptation to Create Adaptive E-Health Mobile Applications
 - 26 Benno Kruit (CWI/VUA), Reading the Grid: Extending Knowledge Bases from Human-readable Tables
 - 27 Jelte van Waterschoot (UT), Personalized and Personal Conversations: Designing Agents Who Want to Connect With You
 - 28 Christoph Selig (UL), Understanding the Heterogeneity of Corporate Entrepreneurship Programs
-
- 2022 01 Judith van Stegeren (UT), Flavor text generation for role-playing video games
 - 02 Paulo da Costa (TU/e), Data-driven Prognostics and Logistics Optimisation: A Deep Learning Journey
 - 03 Ali el Hassouni (VUA), A Model A Day Keeps The Doctor Away: Reinforcement Learning For Personalized Healthcare
 - 04 Ünal Aksu (UU), A Cross-Organizational Process Mining Framework
 - 05 Shiwei Liu (TU/e), Sparse Neural Network Training with In-Time Over-Parameterization
 - 06 Reza Refaei Afshar (TU/e), Machine Learning for Ad Publishers in Real Time Bidding
 - 07 Sambit Praharaj (OU), Measuring the Unmeasurable? Towards Automatic Co-located Collaboration Analytics
 - 08 Maikel L. van Eck (TU/e), Process Mining for Smart Product Design
 - 09 Oana Andreea Inel (VUA), Understanding Events: A Diversity-driven Human-Machine Approach
 - 10 Felipe Moraes Gomes (TUD), Examining the Effectiveness of Collaborative Search Engines
 - 11 Mirjam de Haas (UT), Staying engaged in child-robot interaction, a quantitative approach to studying preschoolers' engagement with robots and tasks during second-language tutoring
 - 12 Guanyi Chen (UU), Computational Generation of Chinese Noun Phrases
 - 13 Xander Wilcke (VUA), Machine Learning on Multimodal Knowledge Graphs: Opportunities, Challenges, and Methods for Learning on Real-World Heterogeneous and Spatially-Oriented Knowledge
 - 14 Michiel Overeem (UU), Evolution of Low-Code Platforms
 - 15 Jelmer Jan Koorn (UU), Work in Process: Unearthing Meaning using Process Mining
 - 16 Pieter Gijsbers (TU/e), Systems for AutoML Research
 - 17 Laura van der Lubbe (VUA), Empowering vulnerable people with serious games and gamification
 - 18 Paris Mavromoustakos Blom (TiU), Player Affect Modelling and Video Game Personalisation
 - 19 Bilge Yigit Ozkan (UU), Cybersecurity Maturity Assessment and Standardisation

- 20 Fakhra Jabeen (VUA), Dark Side of the Digital Media - Computational Analysis of Negative Human Behaviors on Social Media
 - 21 Seethu Mariyam Christopher (UM), Intelligent Toys for Physical and Cognitive Assessments
 - 22 Alexandra Sierra Rativa (TiU), Virtual Character Design and its potential to foster Empathy, Immersion, and Collaboration Skills in Video Games and Virtual Reality Simulations
 - 23 Ilir Kola (TUD), Enabling Social Situation Awareness in Support Agents
 - 24 Samaneh Heidari (UU), Agents with Social Norms and Values - A framework for agent based social simulations with social norms and personal values
 - 25 Anna L.D. Latour (UL), Optimal decision-making under constraints and uncertainty
 - 26 Anne Dirkson (UL), Knowledge Discovery from Patient Forums: Gaining novel medical insights from patient experiences
 - 27 Christos Athanasiadis (UM), Emotion-aware cross-modal domain adaptation in video sequences
 - 28 Onuralp Ulusoy (UU), Privacy in Collaborative Systems
 - 29 Jan Kolkmeier (UT), From Head Transform to Mind Transplant: Social Interactions in Mixed Reality
 - 30 Dean De Leo (CWI), Analysis of Dynamic Graphs on Sparse Arrays
 - 31 Konstantinos Traganos (TU/e), Tackling Complexity in Smart Manufacturing with Advanced Manufacturing Process Management
 - 32 Cezara Pastrav (UU), Social simulation for socio-ecological systems
 - 33 Brinn Hekkelman (CWI/TUD), Fair Mechanisms for Smart Grid Congestion Management
 - 34 Nimat Ullah (VUA), Mind Your Behaviour: Computational Modelling of Emotion & Desire Regulation for Behaviour Change
 - 35 Mike E.U. Ligthart (VUA), Shaping the Child-Robot Relationship: Interaction Design Patterns for a Sustainable Interaction
-
- 2023 01 Bojan Simoski (VUA), Untangling the Puzzle of Digital Health Interventions
 - 02 Mariana Rachel Dias da Silva (TiU), Grounded or in flight? What our bodies can tell us about the whereabouts of our thoughts
 - 03 Shabnam Najafian (TUD), User Modeling for Privacy-preserving Explanations in Group Recommendations
 - 04 Gineke Wiggers (UL), The Relevance of Impact: bibliometric-enhanced legal information retrieval
 - 05 Anton Bouter (CWI), Optimal Mixing Evolutionary Algorithms for Large-Scale Real-Valued Optimization, Including Real-World Medical Applications
 - 06 António Pereira Barata (UL), Reliable and Fair Machine Learning for Risk Assessment
 - 07 Tianjin Huang (TU/e), The Roles of Adversarial Examples on Trustworthiness of Deep Learning
 - 08 Lu Yin (TU/e), Knowledge Elicitation using Psychometric Learning
 - 09 Xu Wang (VUA), Scientific Dataset Recommendation with Semantic Techniques

- 10 Dennis J.N.J. Soemers (UM), Learning State-Action Features for General Game Playing
 - 11 Fawad Taj (VUA), Towards Motivating Machines: Computational Modeling of the Mechanism of Actions for Effective Digital Health Behavior Change Applications
 - 12 Tessel Bogaard (VUA), Using Metadata to Understand Search Behavior in Digital Libraries
 - 13 Injy Sarhan (UU), Open Information Extraction for Knowledge Representation
 - 14 Selma Čaušević (TUD), Energy resilience through self-organization
 - 15 Alvaro Henrique Chaim Correia (TU/e), Insights on Learning Tractable Probabilistic Graphical Models
 - 16 Peter Blomsma (TiU), Building Embodied Conversational Agents: Observations on human nonverbal behaviour as a resource for the development of artificial characters
 - 17 Meike Nauta (UT), Explainable AI and Interpretable Computer Vision – From Oversight to Insight
 - 18 Gustavo Penha (TUD), Designing and Diagnosing Models for Conversational Search and Recommendation
 - 19 George Aalbers (TiU), Digital Traces of the Mind: Using Smartphones to Capture Signals of Well-Being in Individuals
 - 20 Arkadiy Dushatskiy (TUD), Expensive Optimization with Model-Based Evolutionary Algorithms applied to Medical Image Segmentation using Deep Learning
 - 21 Gerrit Jan de Bruin (UL), Network Analysis Methods for Smart Inspection in the Transport Domain
 - 22 Alireza Shojaiifar (UU), Volitional Cybersecurity
 - 23 Theo Theunissen (UU), Documentation in Continuous Software Development
 - 24 Agathe Balayn (TUD), Practices Towards Hazardous Failure Diagnosis in Machine Learning
 - 25 Jurian Baas (UU), Entity Resolution on Historical Knowledge Graphs
 - 26 Loek Tonnaer (TU/e), Linearly Symmetry-Based Disentangled Representations and their Out-of-Distribution Behaviour
 - 27 Ghada Sokar (TU/e), Learning Continually Under Changing Data Distributions
 - 28 Floris den Hengst (VUA), Learning to Behave: Reinforcement Learning in Human Contexts
 - 29 Tim Draws (TUD), Understanding Viewpoint Biases in Web Search Results
-
- 2024 01 Daphne Miedema (TU/e), On Learning SQL: Disentangling concepts in data systems education
 - 02 Emile van Krieken (VUA), Optimisation in Neurosymbolic Learning Systems
 - 03 Feri Wijayanto (RUN), Automated Model Selection for Rasch and Mediation Analysis
 - 04 Mike Huisman (UL), Understanding Deep Meta-Learning
 - 05 Yiyong Gou (UM), Aerial Robotic Operations: Multi-environment Cooperative Inspection & Construction Crack Autonomous Repair
 - 06 Azqa Nadeem (TUD), Understanding Adversary Behavior via XAI: Leveraging Sequence Clustering to Extract Threat Intelligence

- 07 Parisa Shayan (TiU), Modeling User Behavior in Learning Management Systems
- 08 Xin Zhou (UvA), From Empowering to Motivating: Enhancing Policy Enforcement through Process Design and Incentive Implementation
- 09 Giso Dal (UT), Probabilistic Inference Using Partitioned Bayesian Networks
- 10 Cristina-Iulia Bucur (VUA), Linkflows: Towards Genuine Semantic Publishing in Science
- 11 withdrawn
- 12 Peide Zhu (TUD), Towards Robust Automatic Question Generation For Learning
- 13 Enrico Liscio (TUD), Context-Specific Value Inference via Hybrid Intelligence
- 14 Larissa Capobianco Shimomura (TU/e), On Graph Generating Dependencies and their Applications in Data Profiling
- 15 Ting Liu (VUA), A Gut Feeling: Biomedical Knowledge Graphs for Interrelating the Gut Microbiome and Mental Health
- 16 Arthur Barbosa Câmara (TUD), Designing Search-as-Learning Systems
- 17 Razieh Alidoosti (VUA), Ethics-aware Software Architecture Design
- 18 Laurens Stoop (UU), Data Driven Understanding of Energy-Meteorological Variability and its Impact on Energy System Operations
- 19 Azadeh Mozafari Mehr (TU/e), Multi-perspective Conformance Checking: Identifying and Understanding Patterns of Anomalous Behavior
- 20 Ritsart Anne Plantenga (UL), Omgang met Regels
- 21 Federica Vinella (UU), Crowdsourcing User-Centered Teams
- 22 Zeynep Ozturk Yurt (TU/e), Beyond Routine: Extending BPM for Knowledge-Intensive Processes with Controllable Dynamic Contexts
- 23 Jie Luo (VUA), Lamarck's Revenge: Inheritance of Learned Traits Improves Robot Evolution
- 24 Nirmal Roy (TUD), Exploring the effects of interactive interfaces on user search behaviour
- 25 Alisa Rieger (TUD), Striving for Responsible Opinion Formation in Web Search on Debated Topics
- 26 Tim Gubner (CWI), Adaptively Generating Heterogeneous Execution Strategies using the VOILA Framework
- 27 Lincen Yang (UL), Information-theoretic Partition-based Models for Interpretable Machine Learning
- 28 Leon Helwerda (UL), Grip on Software: Understanding development progress of Scrum sprints and backlogs
- 29 David Wilson Romero Guzman (VUA), The Good, the Efficient and the Inductive Biases: Exploring Efficiency in Deep Learning Through the Use of Inductive Biases
- 30 Vijanti Ramautar (UU), Model-Driven Sustainability Accounting
- 31 Ziyu Li (TUD), On the Utility of Metadata to Optimize Machine Learning Workflows
- 32 Vinicius Stein Dani (UU), The Alpha and Omega of Process Mining
- 33 Siddharth Mehrotra (TUD), Designing for Appropriate Trust in Human-AI interaction

- 34 Robert Deckers (VUA), From Smallest Software Particle to System Specification - MuDForM: Multi-Domain Formalization Method
 - 35 Sicui Zhang (TU/e), Methods of Detecting Clinical Deviations with Process Mining: a fuzzy set approach
 - 36 Thomas Mulder (TU/e), Optimization of Recursive Queries on Graphs
 - 37 James Graham Nevin (UvA), The Ramifications of Data Handling for Computational Models
 - 38 Christos Koutras (TUD), Tabular Schema Matching for Modern Settings
 - 39 Paola Lara Machado (TU/e), The Nexus between Business Models and Operating Models: From Conceptual Understanding to Actionable Guidance
 - 40 Montijn van de Ven (TU/e), Guiding the Definition of Key Performance Indicators for Business Models
 - 41 Georgios Siachamis (TUD), Adaptivity for Streaming Dataflow Engines
 - 42 Emmeke Veltmeijer (VUA), Small Groups, Big Insights: Understanding the Crowd through Expressive Subgroup Analysis
 - 43 Cedric Waterschoot (KNAW Meertens Instituut), The Constructive Conundrum: Computational Approaches to Facilitate Constructive Commenting on Online News Platforms
 - 44 Marcel Schmitz (OU), Towards learning analytics-supported learning design
 - 45 Sara Salimzadeh (TUD), Living in the Age of AI: Understanding Contextual Factors that Shape Human-AI Decision-Making
 - 46 Georgios Stathis (Leiden University), Preventing Disputes: Preventive Logic, Law & Technology
 - 47 Daniel Daza (VUA), Exploiting Subgraphs and Attributes for Representation Learning on Knowledge Graphs
 - 48 Ioannis Petros Samiotis (TUD), Crowd-Assisted Annotation of Classical Music Compositions
-
- 2025 01 Max van Haastrecht (UL), Transdisciplinary Perspectives on Validity: Bridging the Gap Between Design and Implementation for Technology-Enhanced Learning Systems
 - 02 Jurgen van den Hoogen (JADS), Time Series Analysis Using Convolutional Neural Networks
 - 03 Andra-Denis Ionescu (TUD), Feature Discovery for Data-Centric AI
 - 04 Rianne Schouten (TU/e), Exceptional Model Mining for Hierarchical Data
 - 05 Nele Albers (TUD), Psychology-Informed Reinforcement Learning for Situated Virtual Coaching in Smoking Cessation
 - 06 Daniël Vos (TUD), Decision Tree Learning: Algorithms for Robust Prediction and Policy Optimization
 - 07 Ricky Maulana Fajri (TU/e), Towards Safer Active Learning: Dealing with Unwanted Biases, Graph-Structured Data, Adversary, and Data Imbalance
 - 08 Stefan Bloemheuvel (TiU), Spatio-Temporal Analysis Through Graphs: Predictive Modeling and Graph Construction
 - 09 Fadime Kaya (VUA), Decentralized Governance Design - A Model-Based Approach

- 10 Zhao Yang (UL), Enhancing Autonomy and Efficiency in Goal-Conditioned Reinforcement Learning
- 11 Shahin Sharifi Noorian (TUD), From Recognition to Understanding: Enriching Visual Models Through Multi-Modal Semantic Integration
- 12 Lijun Lyu (TUD), Interpretability in Neural Information Retrieval
- 13 Fuda van Diggelen (VUA), Robots Need Some Education: on the complexity of learning in evolutionary robotics
- 14 Gennaro Gala (TU/e), Probabilistic Generative Modeling with Latent Variable Hierarchies
- 15 Michiel van der Meer (UL), Opinion Diversity through Hybrid Intelligence
- 16 Monika Grewal (TU Delft), Deep Learning for Landmark Detection, Segmentation, and Multi-Objective Deformable Registration in Medical Imaging
- 17 Matteo De Carlo (VUA), Real Robot Reproduction: Towards Evolving Robotic Ecosystems
- 18 Anouk Neerincx (UU), Robots That Care: How Social Robots Can Boost Children's Mental Wellbeing
- 19 Fang Hou (UU), Trust in Software Ecosystems
- 20 Alexander Melchior (UU), Modelling for Policy is More Than Policy Modelling (The Useful Application of Agent-Based Modelling in Complex Policy Processes)
- 21 Mandani Ntekouli (UM), Bridging Individual and Group Perspectives in Psychopathology: Computational Modeling Approaches using Ecological Momentary Assessment Data
- 22 Hilde Weerts (TU/e), Decoding Algorithmic Fairness: Towards Interdisciplinary Understanding of Fairness and Discrimination in Algorithmic Decision-Making
- 23 Roderick van der Weerd (VUA), IoT Measurement Knowledge Graphs: Constructing, Working and Learning with IoT Measurement Data as a Knowledge Graph
- 24 Zhong Li (UL), Trustworthy Anomaly Detection for Smart Manufacturing
- 25 Kyana van Eijndhoven (TiU), A Breakdown of Breakdowns: Multi-Level Team Coordination Dynamics under Stressful Conditions
- 26 Tom Pepels (UM), Monte-Carlo Tree Search is Work in Progress
- 27 Danil Provodin (JADS, TU/e), Sequential Decision Making Under Complex Feedback
- 28 Jinke He (TU Delft), Exploring Learned Abstract Models for Efficient Planning and Learning
- 29 Erik van Haeringen (VUA), Mixed Feelings: Simulating Emotion Contagion in Groups
- 30 Myrthe Reuver (VUA), A Puzzle of Perspectives: Interdisciplinary Language Technology for Responsible News Recommendation
- 31 Gebrekirstos Gebreselassie Gebremeskel (RUN), Spotlight on Recommender Systems: Contributions to Selected Components in the Recommendation Pipeline

